Graz University of Technology

Doctoral Thesis

Variational Sparse Bayesian Learning: Centralized and Distributed Processing

Thomas Buchgraber

Faculty of Electrical and Information Engineering
Graz University of Technology, Austria

Advisors:
Dr. Dmitriy Shutin, German Aerospace Center (DLR), Oberpfaffenhofen
Prof. Dr. Gernot Kubin, Graz University of Technology, Austria

Examiners:
Prof. Dr. Gernot Kubin, Graz University of Technology, Austria
Prof. Dr. Jan Larsen, Technical University of Denmark, Lyngby

Graz, April 2013

Abstract

In this thesis we investigate centralized and distributed variants of sparse Bayesian learning (SBL), an effective probabilistic regression method used in machine learning. Since inference in an SBL model is not tractable in closed form, approximations are needed. We focus on the variational Bayesian approximation, as opposed to others used in the literature, for three reasons: First, it is a flexible general framework for approximate Bayesian inference that estimates probability densities including point estimates as a special case. Second, it has guaranteed convergence properties. And third, it is a deterministic approximation concept that is even applicable for high dimensional problems where non-deterministic sampling methods may be prohibitive.

We resolve some inconsistencies in the literature involved in other SBL approximation techniques with regard to a proper Bayesian treatment and the incorporation of a very desired property, namely scale invariance. More specifically, among all discussed methods, we show that the variational Bayesian approximation is the only approach that can be consistently derived from scale-invariant model assumptions.

A novel fast SBL concept based on fixed-point analysis of the variational update expressions is presented as a variational counterpart to fast marginal likelihood maximization, a widely known SBL variant. Within this fast framework, which we denote as fast variational SBL (FV-SBL), we furthermore show how to incorporate an intuitive trade-off parameter that allows to balance sparsity versus accuracy. We compare the performance of all methods using real and synthetic data, where we show that FV-SBL converges several orders of magnitude faster than ordinary variational SBL.

Large-scale sensor networks allow to monitor spatial phenomena with unprecedented resolution. Current research focuses on distributed processing performed directly on the sensor nodes with only local communications and no need for data fusion, i.e., the data is not collected and processed at a single point which would be critical to fail. In the SBL context, our objective is to learn a compact model of the true underlying spatially sampled field function based on noisy sensor measurements in a distributed manner. Finding a sparse representation of the data enables its ef-

ficient handling and processing, which is a desired property especially for wireless sensor network applications due to energy and bandwidth constraints.

We analyze two distributed variants of SBL, which we denote as variational distributed SBL (V-dSBL) and fast variational distributed SBL (FV-dSBL). Both algorithms are based on consensus methods, which perform distributed averaging based on local message passing between neighboring sensors. V-dSBL considers a spatially distributed probabilistic model that is a combination of SBL and consensus propagation. For inference, a variational Bayesian approximation with Gaussian loopy belief propagation as a subroutine is used. The messages involved in this subroutine are directly sent between adjacent sensors and have a communication complexity that is quadratic in the number of model components. In FV-dSBL, the FV-SBL fixed-point solution is collaboratively calculated using any type of consensus algorithm, which results in a communication complexity that is only linear in the number of model components for each message. Furthermore, in FV-dSBL, we can test and add new components to the model. The computational complexity at each sensor for a single FV-dSBL test is quadratic in the number of components, whereas in V-dSBL it is cubic for each single message in Gaussian loopy belief propagation. In real- and synthetic-data experiments, we compare both distributed algorithms to their centralized counterparts, where similar performance is obtained in most of the cases.

Kurzfassung

In dieser Dissertation werden zentralisierte und verteilte Varianten von Sparse Bayesian Learning (SBL) untersucht. Dies ist eine effektive bayessche probabilistische Regressionsmethode des maschinellen Lernens, welche auf dünn besetzten („sparse“) Vektoren beruht. Da bayessche Inferenz in SBL Modellen nicht in geschlossener Form lösbar ist, werden Annäherungsverfahren verwendet. Der Fokus dieser Arbeit liegt dabei auf bayesschen Variationsmethoden. Dies hat die folgenden drei Gründe: Erstens stellen sie einen allgemeinen Rahmen für die bayessche approximative Inferenz dar, welche Wahrscheinlichkeitsdichten annähert und Punktschätzer als Spezialfall beinhaltet. Zweitens ist ihre Konvergenz garantiert. Und drittens sind sie deterministische Approximationsmethoden, welche sogar für hochdimensionale Probleme anwendbar sind, bei denen nicht-deterministische Sampling-Methoden zu viel Ressourcen brauchen würden.

Weiters werden in dieser Arbeit einige Ungereimtheiten in anderen SBL Approximationen aus der Literatur ausgeräumt. Diese Ungereimtheiten beziehen sich auf die Korrektheit der Methoden und die Berücksichtigung einer erwünschten Eigenschaft namens Skaleninvarianz. Wir zeigen, dass die bayessche Variationsmethode die einzige der diskutierten Methoden ist, welche die Modellannahmen für Skaleninvarianz in konsistenter Weise berücksichtigt.

Es wird eine schnelle Variante der bayesschen Variationsmethode für SBL vorgestellt, welche auf einer Fixpunktanalyse der iterativen Update-Gleichungen beruht. Diese Methode, welche wir als FV-SBL bezeichnen, ist das bayessche variationsmethodische Gegenstück zu einer weit bekannten schnellen SBL Methode namens Fast Marginal Likelihood Maximization. Außerdem zeigen wir, wie man in FV-SBL einen Parameter zum Abwägen der Modellkomplexität und Genauigkeit einführen kann. Wir zeigen mit realen und synthetischen Experimenten, dass FV-SBL um mehrere Größenordnungen schneller konvergiert als SBL mit der gewöhnlichen bayesschen Variationsmethode.

Aktuelle Sensornetzwerk-Forschung konzentriert sich auf verteilte Berechnungen, welche direkt auf den Sensorknoten ausgeführt werden und nur lokale Kommunikation benötigen. Dies steigert die Robustheit gegenüber Sensorausfällen. Wir untersu-

chen verteilte SBL Algorithmen, welche Feldfunktionen aus verrauschten Sensormessungen in einem kompakten („sparse“) Modell lernen. Die kompakte Repräsentation der Daten ermöglicht eine effiziente Weiterverarbeitung, welche gerade in Sensornetzwerken wegen vorhandener Energie- und Bandbreitenbegrenzungen sehr erwünscht ist.

Es werden zwei verteilte SBL Algorithmen mit den Bezeichnungen V-dSBL und FV-dSBL vorgestellt. Beide Algorithmen basieren auf Konsensmethoden, welche in verteilter Weise Durchschnittswerte berechnen. V-dSBL verwendet ein räumlich verteiltes bayessches Modell welches aus einer Kombination einer Konsensmethode und SBL besteht. Für die Inferenz wird eine bayessche Variationsmethode mit einer gaußschen Loopy Belief Propagation (LBP) Unterroutine verwendet. Die Nachrichten, welche in der Unterroutine zwischen den Sensoren gesendet werden, haben eine Kommunikationskomplexität welche quadratisch mit der Anzahl der Modellkomponenten wächst. Bei FV-dSBL wird die Fixpunktlösung von FV-SBL in verteilter Weise durch eine Konsensmethode berechnet. Die Kommunikationskomplexität von dieser Methode ist für jede Nachricht nur linear in der Anzahl der Modellkomponenten. Außerdem können bei FV-dSBL neue Komponenten getestet und zum Modell hinzugefügt werden. Die Rechenkomplexität von jedem FV-dSBL Test wächst an jedem Sensor quadratisch mit der aktuellen Komponentenanzahl. Bei V-dSBL hingegen wächst sie kubisch in der Anzahl der aktuellen Komponenten für jede einzelne Nachricht. In realen sowie synthetischen Experimenten werden beide Algorithmen mit ihren zentralisierten Gegenstücken verglichen, wobei in den meisten Fällen vergleichbare Ergebnisse erzielt werden.

Contents

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	(signature)

Acknowledgments

First of all, I would like to thank my advisors Dmitriy Shutin and Prof. Gernot Kubin for their support.

Dmitriy, who left the Lab at an early phase of my research, remote-advised me during all these years. I'm very grateful to your cooperation, helpfulness and endurance. You taught me how to work scientifically and see complicated concepts in a new and simpler way. Thank you for your valuable time.

Gernot, I would like to thank you for many comments concerning fundamental issues of this work. You have the gift to encourage people to improve themselves. This is not only due to the open research environment that you provide at our Lab, but also due to the many fundamental questions that you raise during discussions.

I would like to say a special thanks to my master's thesis advisor Prof. Klaus Witrisal, who encouraged and supported me to start this PhD program. Now, having finished this thesis, Klaus, I am happy to say that I am sure that it was the right decision.

Furthermore, I would like to thank all members of the Signal Processing and Speech Communication Laboratory for their helpfulness and the very relaxed and fruitful atmosphere. I will never forget about the endless discussions that we had during coffee breaks or lunch.

I am very grateful to the *Austrian Science Fund* (FWF) who supported this work within the national research network *Signal and Information Processing in Science and Engineering* (SISE) under awards S10604-N13 and S10610-N13. I am also grateful to all colleagues within the SISE network, and especially to Valentin Schwarz for many valuable discussions about consensus propagation.

At this point, I would like to thank Prof. Vincent Poor, who made my research visit to Princeton University in 2010 possible. Additionally, I would like to thank the *Austrian Marshall Plan Foundation* for financially supporting this visit.

I would like to express my gratitude to Prof. Jan Larsen for taking part in the thesis assessment.

Furthermore, I would like to thank my parents, family and friends for supporting

me in many different important ways. The former also holds for Stefanie's family, which I consider as my family, too.

Finally, I would like to express my deepest gratitude to Stefanie. She walked with me through all the ups and downs during the last years. Without her on my side, I am not sure if these pages would have ever come into existence. Thank you for your love, support and endurance.

Graz, April 2013 Thomas

List of Figures

List of Abbreviations

AC	Average Consensus
Bfs	Basis functions
BN	Bayesian Network
CCS	Concrete Compressive Strength dataset
CP	Consensus Propagation
EM	Expectation Maximization
FMLM	Fast Marginal Likelihood Maximization
FV-dSBL	Fast Variational Distributed Sparse Bayesian Learning
FV-SBL	Fast Variational Sparse Bayesian Learning
i.i.d.	Independent and identically distributed
Kernel-RLS	Kernel recursive least squares algorithm
LBP	Loopy Belief Propagation
MAP	Maximum a posteriori
ML	Maximum Likelihood
MML	Maximum Marginal Likelihood
MML*	MML with implicit hyperparameter updates
MMP-log	Maximum Marginal Posterior over a logarithmic scale
MSE	Mean Square Error
NMSE	Normalized Mean Square Error

NT	Network
pdf	Probability density function
RVM	Relevance Vector Machine
SBL	Sparse Bayesian Learning
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machine
V-dSBL	Variational Distributed Sparse Bayesian Learning
V-SBL	Variational Sparse Bayesian Learning
VB	Variational Bayes
WSN	Wireless Sensor Network

Mathematical Notation

Notation	Description
$\boldsymbol{I}$	Identity matrix of adequate dimension
$\text{diag}(\boldsymbol{x})$	Matrix with the elements of a vector $\boldsymbol{x}$ on the main diagonal
$\text{diag}(\boldsymbol{X})$	Matrix containing only the main diagonal elements of the matrix $\boldsymbol{X}$
$\text{tr}(\boldsymbol{X})$	Trace of the matrix $\boldsymbol{X}$
$[\boldsymbol{B}]_{kl}$ or B_{kl}	kth row and lth column element of the matrix $\boldsymbol{B}$
$[\boldsymbol{b}]_k$ or b_k	If $\boldsymbol{b}$ is a vector, then b_k is its kth element
$[\boldsymbol{B}]_{\bar{k}\bar{l}}$	Matrix obtained by deleting the kth row and lth column from the matrix $\boldsymbol{B}$
$[\boldsymbol{b}]_{\bar{k}}$	Vector obtained by deleting the kth element from the vector $\boldsymbol{b}$
$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian pdf over $\boldsymbol{x}$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\text{Ga}(x\|a, b)$	Gamma pdf over x with parameters a and b, defined as $\text{Ga}(x\|a,b) = \frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx)$, where $\Gamma(\cdot)$ denotes the gamma-function
$\boldsymbol{x}_n$	nth D-dimensional input vectors
t_n	nth scalar target value
N	Number of training examples
K	Number of sensors in the network
$\{\boldsymbol{x}_n, t_n\}_{n=1}^N$	Training dataset
$\psi_m(\boldsymbol{x})$	mth basis function of $\boldsymbol{x}$
M	Number of basis functions
$\boldsymbol{\Phi}$	$N \times M$ design matrix with $[\boldsymbol{\Phi}]_{nm} = \psi_m(\boldsymbol{x}_n)$

$\boldsymbol{\varphi}_m$	mth basis vector, mth column of $\boldsymbol{\Phi}$ defined as $\boldsymbol{\varphi}_m = [\psi_m(\boldsymbol{x}_1), \ldots, \psi_m(\boldsymbol{x}_N)]^T$
$\boldsymbol{\phi}_n^T$	nth row vector of $\boldsymbol{\Phi}$, $\boldsymbol{\phi}_n = [\psi_1(\boldsymbol{x}_n), \ldots, \psi_M(\boldsymbol{x}_n)]^T$
ϵ_n	Additive noise variable, $\epsilon_n \sim \mathcal{N}(\epsilon_n \vert 0, \tau^{-1})$
$\boldsymbol{\epsilon}$	Vector with *i.i.d.* noise variables $\boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_N]^T$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon} \vert \mathbf{0}, \tau^{-1}\boldsymbol{I}) = \prod_{n=1}^{N} \mathcal{N}(\epsilon_n \vert 0, \tau^{-1})$
σ^2	Noise variance
τ	Noise precision (inverse variance), $\tau = \sigma^{-2}$
w_m	Weight parameter for the mth basis function
$\boldsymbol{w}$	Weight vector $\boldsymbol{w} = [w_1, \ldots, w_M]^T$
α_m	Hyperparameter for the mth basis function
$\boldsymbol{\alpha}$	Hyperparameter vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_M]^T$
$\mathrm{KL}(q\Vert p)$	Kullback-Leibler divergence, functional of the distributions $q(\cdot)$ and $p(\cdot)$
$\mathcal{L}(q)$	Variational lower bound as a functional over $q(\cdot)$
$N(k)$	Set of neighbors of sensor k; see definition in Section 4.3.2
$\#N(k)$	Number of neighbors of sensor k; see definition in Section 4.3.2

Chapter 1

Introduction

1.1 Machine learning and regression

Machine learning deals with the design and the analysis of algorithms that make inferences based on data observations. These algorithms, which are said to *learn* from the data, can be categorized into several types. Such types are for instance *supervised learning*, *unsupervised learning* or *reinforcement learning*, to name a few. In this thesis we will consider the first mentioned type, namely *supervised learning*. In *supervised learning*, we are given a set of N *training samples* each consisting of an input/target pair [4]. We define the inputs, with sample index n, as D-dimensional vectors $\boldsymbol{x}_n = [x_{n,1}, \ldots, x_{n,D}]^T \in \mathbb{R}^D$, which can be used to represent almost any sort of data. For instance, $\boldsymbol{x}_n$ could be a gray scale image, where the dimension D is the number of pixels and the values $x_{n,i}$ represent the intensity of pixel i. For each such input $\boldsymbol{x}_n$, we have given an associated target $t_n \in \mathcal{T}$, where $\mathcal{T}$ is the target set. If the target set $\mathcal{T}$ is a finite set, the *supervised learning* task is denoted as *classification* and if it is an infinite set (typically $\mathcal{T} \subseteq \mathbb{R}$), it is denoted as *regression*. For our gray scale image example above, if we consider *classification* of, e.g., pictures with cats and dogs, the target set could be $\mathcal{T} = \{0, 1\}$, where a '0' indicates, e.g., a cat and '1' a dog. For *regression*, we could think of, e.g., estimating the angle from which a picture was taken relatively to the animal. In this case we could define $\mathcal{T} = [-\pi, \pi) \subseteq \mathbb{R}$. Throughout this thesis we will only consider *regression* and more specifically $\mathcal{T} = \mathbb{R}$.

We denote the set of all training samples $\mathcal{D} = \{\boldsymbol{x}_n, t_n\}_{n=1}^N$ as the *training data set*. In *regression*, if we assume $\mathcal{T} = \mathbb{R}$, the fundamental goal is to learn a prediction $y(\boldsymbol{x})$ given an arbitrary input $\boldsymbol{x} \in \mathbb{R}^D$, where $y : \mathbb{R}^D \to \mathbb{R}$. The quality of this prediction is typically estimated by defining an error measure on some unseen test dataset [4]. A fundamental problem in *supervised learning* is *overfitting*. That is, if noise is present in the training data $\mathcal{D}$, a learning algorithm that has a small error

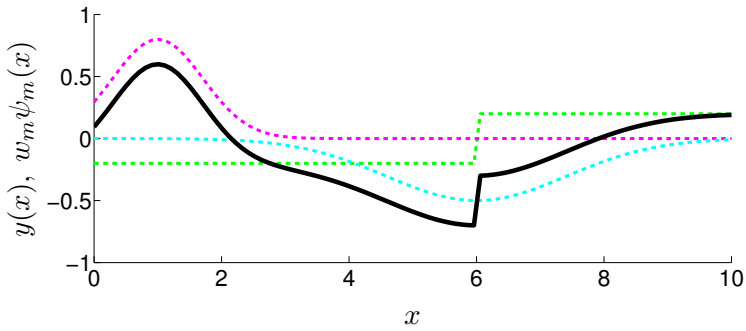


Figure 1.1: Weighted sum of three example basis functions. The dotted curves represent the weighted basis functions $w_m\psi_m(x)$ and the solid black curve shows the superposition.

measure on $\mathcal{D}$ leads to bad predictions on the test dataset, i.e., it does not generalize well for unseen data. A well known method to reduce *overfitting* is regularization [4].

1.1.1 Linear models for regression

To learn a prediction $y(\boldsymbol{x})$, we need a flexible model that can be adjusted to a broad class of datasets $\mathcal{D}$. In this thesis we consider *linear models for regression* [4], which are superpositions of M weighted basis functions $\psi_m(\boldsymbol{x})$, i.e.,

$$y(\boldsymbol{x}) = \sum_{m=1}^{M} w_m \psi_m(\boldsymbol{x}). \tag{1.1}$$

We give an example for $M = 3$ in Figure 1.1. The basis functions are fixed and in general *non-linear*. However, such models are denoted *linear*, since they are linear in the weight parameters w_m, which have to be adjusted (learned). Typcially used basis functions are Fourier-, wavelet-, kernel- or radial basis functions [5, 6, 4, 7]. Note that in the machine learning literature, the terms *basis functions* and, as we will also use later in this thesis, *basis vectors*, differ from the mathematical definition of a basis, which requires linear independency. Throughout this thesis, we will stick to the common definition in machine learning, where *basis functions* and *basis vectors* need not to be linearly independent. Basically, there are almost no restrictions on $\psi_m(\boldsymbol{x})$ and we are free to choose any set of fixed basis functions in (1.1). Note that there are also many other models for the regression functions $y(\boldsymbol{x})$ that are non-linear in the parameters, like, for instance *multi-layer perceptrons* [8]. However, for models which are linear in the weights, learning $w_1, \ldots, w_M$ and thus $y(\boldsymbol{x})$ from the data $\mathcal{D}$

is relatively simple using, e.g., common methods like *regularized least-squares* [9, 4].

1.1.2 Non-parametric regression using kernels

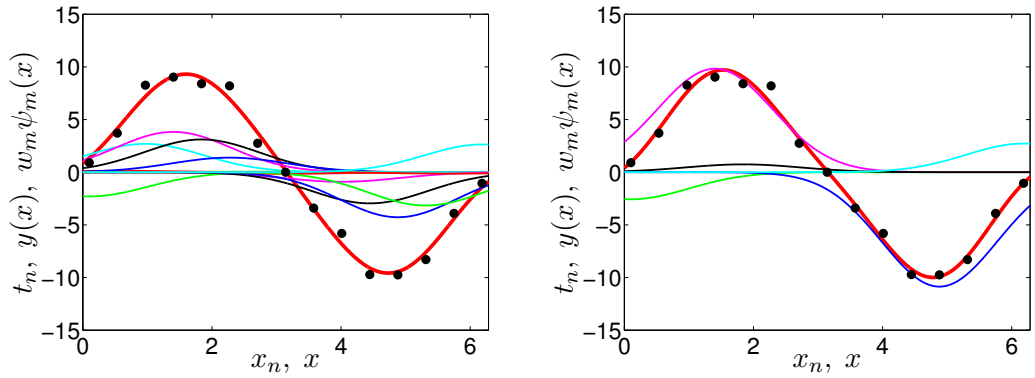
The chosen set of basis functions strongly influences the predictive performance. It is not clear in general how to specify these functions. Furthermore, as the input dimensionality D increases, the required number of basis functions for a reasonable predictive performance grows rapidly and often exponentially [4].

Let us first discuss a flexible data dependent concept of how to define the set of basis functions known as *kernel methods*. Other simple data dependent concepts for defining basis functions can be found for instance in *multivariate adaptive regression spline models* (MARS models); see [10]. Later in Section 1.1.3, we discuss how the required number of basis functions can be reduced.

Kernel methods are widely used in machine learning. A detailed introduction can be found in standard textbooks like [11, 4]. However, in the context of regression, amongst additional restrictions, we can basically interpret kernels as ordinary basis functions ψ_m with an extra parameter $\boldsymbol{x}'_m$, i.e., $\psi_m(\boldsymbol{x}) = \kappa(\boldsymbol{x}, \boldsymbol{x}'_m)$, where κ is the kernel function. The additional restrictions are that a function κ is only denoted as kernel, if it satisfies *Mercer's theorem* [12, 13]. A necessary and sufficient condition for a valid kernel, based on this theorem, is that, for any finite set $\{\boldsymbol{u}_i\}$, the Gram matrix $\boldsymbol{K}$, with elements defined as $K_{ij} = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$, must be positive semidefinite; see also [14, 4]. Furthermore, in kernel methods, the parameters $\boldsymbol{x}'_m$ are defined by the training data $\mathcal{D}$, i.e., $\boldsymbol{x}'_n = \boldsymbol{x}_n$ for $n = 1, \ldots, N$ and we have $M = N$. That is, the basis functions are fully specified by the training data and not fixed in advance like we have, e.g., for Fourier basis functions. We note that in some cases additional basis functions like a constant bias $\psi_0(\boldsymbol{x}) = 1$ may be added as well, where $M > N$. However, once specified, the set of basis functions is fixed and will not be changed during learning. Such methods that are entirely build on the data, are denoted as *non-parametric*[1].

Popular kernels are the polynomial kernel and the Gaussian kernel. The polynomial kernel is defined as $\kappa(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + \tilde{c})^{\tilde{d}}$ with degree $\tilde{d}$ and shift parameter $\tilde{c} > 0$. The Gaussian kernel is defined as $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp\{-\theta_\kappa \|\boldsymbol{x} - \boldsymbol{x}'\|^2\}$ with inverse width parameter θ_κ. Note that such a kernel has the shape of a Gaussian *probability density function* (pdf) but is not normalized, i.e., it does not necessarily integrate to 1. Sums of weighted Gaussian kernels are known to be *universal approx-*

[1] Note that if an additional bias basis function is used, it is still denoted as a *non-parametric* model.



(a) $N = M = 15$ (b) $N = 15,\ M = 5$

Figure 1.2: Kernel regression for noisy *sine* data with (a) a full model and (b) a sparse model. The predicted function is depicted by a thick red solid curve and the thin colored curves show the weighted Gaussian kernels centered at the input data points x_n.

imators for arbitrary continuous real valued functions on a compact set [15, 16, 17][2]. Coarsely speaking, this means that a model based on such kernels can approximate any continuous function up to an arbitrary small error.

1.1.3 Parameter estimation, probabilistic models and sparse regression

For most learning scenarios, if the number of basis functions M in (1.1) is large, usually only a small subset of basis functions is actually required to represent the data. This holds especially for kernel basis functions, introduced in Section 1.1.2, if the training dataset $\mathcal{D}$ is large, since we have $M \geq N$.[3] A kernel regression example with noisy *sine* data is depicted in Figure 1.2(a), where we have used *regularized least-squares* [9, 4] for the weight parameter estimation (learning).[4] A much more compact model for the same data can be seen in Figure 1.2(b), where only 1/3 of the kernels compared to Figure 1.2(a) are used. The problem is, without observing the data, it is not possible to tell in advance which of the basis functions are important.

A famous example of a kernel method that estimates, besides the weight parameters, also the important components, is the *support vector machine* (SVM)

[2]Note that weighted sums of polynomial kernels of fixed degree are no *universal approximators*, however, if the degree is not limited, they are *universal approximators* for arbitrary continuous real valued functions on a compact set as well [16].
[3]If only kernels and no extra basis functions are considered we have $M = N$.
[4]Note that standard *least squares* without regularization typically leads to overfitting [4].

[18, 11, 19]. The less important kernels are simply *switched* off during learning by setting the corresponding weights w_m to zero. According to (1.1), this has the same effect as removing the kernels from the model. In such a case, the model is denoted as *sparse*, which is also why SVMs are denoted as *sparse kernel machines* [4]. Note that SVMs are also defined for classification, however, we only consider the regression type here in our discussion.

On the other hand, probabilistic models have shown to be successful in many areas of machine learning; see, e.g., [18, 20, 21, 4]. A big advantage of such models is that they naturally incorporate the handling of uncertainties. In this way, a learning algorithm can provide for instance a measure about how sure it is about its prediction. Furthermore, uncertainties typically involved in the training data, like the noise in our example from Figure 1.2, can be modelled probabilistically as well.

Probabilistic models basically consist of hidden and observed random variables. An example for hidden random variables in our context would be the weights w_m in (1.1), since we cannot measure them. The targets from the training dataset for instance can be modelled as observed random variables. If we know (or in many cases assume) the probabilistic model that relates these variables to each other, we can make inference about the posterior probability, i.e., the probability of the hidden variable given the observed variables. Inference in such probabilistic models typically requires to apply *Bayes' rule* of probabilities [22], which is why these models are also denoted as Bayesian models.

A Bayesian counterpart to the SVM is the *relevance vector machine* (RVM) [23, 1] that is a special case of *sparse Bayesian learning* (SBL) [1, 24], which we investigate throughout this thesis.[5] However, the models for SBL and RVMs are basically equivalent. The only difference is the used set of basis functions. The more general term SBL is used for models with arbitrary basis functions as in (1.1), whereas RVMs only use kernel basis functions and maybe a constant bias. In other words, SBL is the general term for the probabilistic learning framework and RVMs are the application of this framework using kernels. A direct comparison of RVMs and SVMs, discussed above, typically leads to sparser models and better prediction performances for RVMs [1], although RVMs are computationally more complex than SVMs. RVMs are also denoted as *sparse kernel machines* like SVMs [4]. A fundamental difference of the general SBL model and SVMs is that SBL is a probabilistic model that can be used with arbitrary basis functions, whereas SVMs use deterministic models that

[5] Note that even though we only consider regression throughout this thesis, SBL is also defined for classification [1].

only allow kernel basis functions.

During the last decade, the research of sparse signal representations has received considerable attention in many different research areas [25, 26, 27, 28]. Imposing constraints on the model parameters, in our case $w_1, \ldots, w_M$, is the key to sparse signal modelling [26]. For probabilistic models, like SBL, such constraints can be incorporated via the definition of prior probabilities over such parameters [29].

Finding the optimal sparse representation of the training data, i.e, finding the sparsest representation in the noiseless case or the representation with the smallest error and sparsity for a given error-sparsity trade-off, in general is an combinatorial NP-hard problem [30]. However, in [29] it has been shown that SBL has good properties in approximating the optimal solution. A very interesting feature of SBL is that the parameters which control the sparsity are directly learned from the data itself and additionally act as automatic regularization to avoid overfitting. In most other methods, like in the ℓ_1-*regularized* LASSO [31] for instance, such parameters have to be cross-tuned via several algorithm runs.

Unfortunately, exact inference in probabilistic models is often intractable, i.e., there exists no closed form solution for the posterior. Yet, numerous approximation techniques have been developed; see, e.g., [32, 33, 34]. Note that approximations are also required for SBL, where we will focus on a particular technique, named, *variational Bayesian inference* [34] throughout this thesis. This approximation technique was first applied to the SBL model in [35], where we will develop a much faster implementation and distributed extensions as we will further discuss in Section 1.3.

1.2 Distributed learning in Wireless Sensor Networks

1.2.1 Wireless Sensor Networks

Wireless technologies have become ubiquitous in our everyday life. The continuing advances in microelectronics lead to cheaper, smaller and smarter mobile devices at reduced energy consumption per processing power; a trend that seems not to stop in the foreseeable future. The lack of cabling, which is essential for the desired flexibility, typically makes these devices dependent on some limited energy resource like a battery and thus substantiates the necessity for power aware design at both the hardware and software development stages.

The type of mobile devices that we consider are wireless sensor nodes, which together form a wireless sensor network (WSN) [36, 37, 38, 39]. Each sensor node basically consists of a sensing unit, a transceiver and a processing unit [38]. Note

that the term *sensor node* or simply *sensor*, which is frequently used in the context of WSNs, is a little misleading since it denotes the whole device and not only the sensing unit. However, throughout the thesis we also stick to these commonly used terms.

Sensor networks can be used in many different areas, ranging from military applications like surveillance and target localization [40] to civilian applications like environmental-, health- or structural-monitoring [41, 42, 43] and industrial process control [44]. There are very stringent limitations on the wireless sensor nodes' energy consumption due to their usually long operational life span. In most cases, the nodes are powered by batteries, and typically for large scale networks consisting of cheap devices, recharging or exchanging batteries is too expensive or even impossible, which again points out the necessity for energy awareness to increase the network's life time. Even though energy harvesting from external sources like, e.g., solar, RF, wind or thermal energy [45] can be used in some cases, it is not always applicable and the amount of available power is usually fluctuating and often very small, which makes energy efficiency even more important. Furthermore, due to the energy constraints and long operational life span, the typical transceiver data rate of a sensor is quite low.

Note that throughout the thesis we use the terms *sensor network* and *wireless sensor network* interchangeably. Even though all methods that will be discussed in this thesis can be equivalently applied to *"wired" sensor networks* as well, the development of the methods however is strongly motivated by the constraints that have to be faced within WSNs. Besides the energy and data rate constraints discussed above, in the following we discuss further issues that may arise particularly in WSNs.

Sensor networks are designed to spatially measure and process data from the environment in which they are deployed. Typically, the sensors together form a *wireless ad-hoc network* [38], where only neighboring devices can communicate directly due to their limited communication range. An example network is depicted in Figure 1.3, where we have also shown an exemplary underlying field function that is spatially sampled by the network. Such a field could be any physical measure like, for instance temperature, light intensity or some level of pollution, to name a few.

In many applications, the measured data is collected and processed at a powerful central node in the network, which is denoted as the *fusion center* [46]. Transferring data from each node to the fusion center requires sensors to relay information from distant parts of the network towards the fusion center. This strategy requires *multi-*

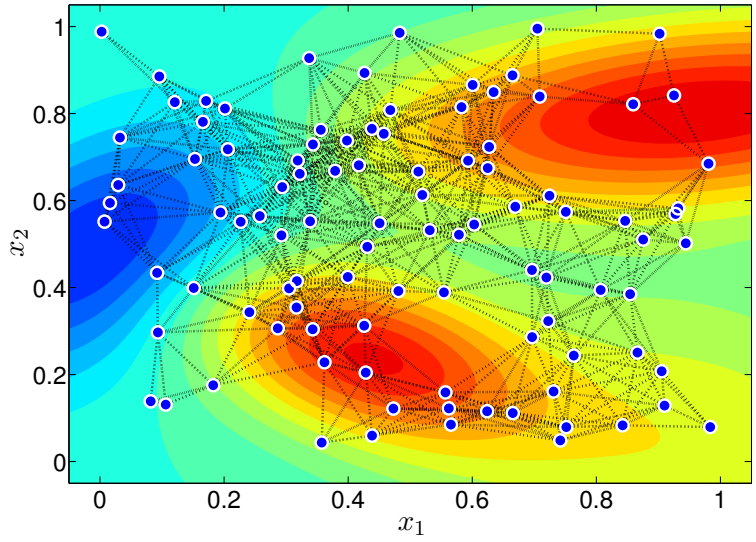


Figure 1.3: Schematic plot of a wireless sensor network (WSN) spatially sampling a 2-dimensional field. Sensors (blue dots) can only communicate along the communication links (dotted lines).

hop communication and appropriate routing protocols need to be implemented [47]. Such protocols should be robust against dynamically changing network topologies, i.e., some connections may suddenly fail or build up from scratch due to changes in the environment. In applications with mobile sensors, such topology changes happen permanently.

Furthermore, sensors may fail due to damage, malfunction or simply an empty battery. If the network is still connected, i.e., information can still be transferred from every working sensor to the fusion center, this may not be a problem in many realistic scenarios, where measurements are highly correlated or even redundant.[6] However, if the *fusion center* fails, the task of the whole network can no longer be performed.

For more robustness and/or less energy consumption, especially for large scale WSNs, *distributed processing* may be an interesting alternative as we discuss in the following.

[6]Note that the correlation of sensor measurements is also an important property for sparse data modelling that we exploit in our distributed SBL variants later in this thesis.

1.2.2 Distributed processing and learning

The fundamental purpose of a sensor network is to make inference about the environment that it is sensing [46]. In the centralized approach that we discussed above, all data needs to be relayed to the fusion center, which may be inefficient, especially for large networks. An interesting alternative is to collaboratively process the data by all sensors within the network. This approach, which is denoted as *in-network processing* or *distributed signal and information processing* [48, 49, 50, 51], offers several advantages, namely:

- Local communication
- Routing protocols are not necessarily needed
- Computational resources can be shared by the nodes
- More robustness, since there is no central point of failure.

For distributed algorithms that collaboratively estimate some parameters of interest through local communication without sharing the sensor measurements, another advantage is *privacy*. That is, a sensor that updates its estimates based on its local measurement and the estimates from its direct neighbors does not see the other sensors' measurements. Such a protocol will be introduced in the context of *consensus algorithms for distributed averaging* later in this thesis.

Past research for distributed parametric models has focused on decentralized estimation or detection if the statistics of the phenomena under observation are assumed to be known [46]. However, in situations where prior knowledge is non-existing or vague, nonparametric methods are desirable [46]. Methods for nonparametric *distributed learning*, including a kernel regression method, are also discussed in [46]. Note that due to the energy and bandwith limitations that have to be faced within WSNs, *distributed learning* is fundamentally different from learning approaches in *parallel computing* [52, Section 2.2].

From a machine learning perspective, as we discussed in Section 1.1, a distributed regression problem for field estimation can be defined by considering the sensor coordinates as inputs $\boldsymbol{x}_k$ and the corresponding measurements as the targets t_k, where $k = 1, \ldots, K$, and K is the number of sensors in the network. Each sensor can be interpreted as a sample from a distributed training dataset $\mathcal{D}$. For example, in the $D = 2$ dimensional network from Figure 1.3, the sensors could collaboratively learn from their measurements t_k to estimate the underlying function. After learning,

each sensor k obtains a similar predictive function $y_k(\boldsymbol{x})$ that is an estimate of the true underlying field function.

1.3 Work contributions

The main contributions of this thesis are twofold.

First, we investigate SBL in a centralized setting, where the whole training data $\mathcal{D}$ is directly accessible by a centralized learning algorithm. Here, we discuss several SBL approximations known in the literature that all lead to the same iterative update equations under slightly different model assumptions. In this context we show that the variational Bayesian (VB) approximation to SBL is conceptionally superior to the discussed alternatives. More specifically, we show that a particular SBL model assumption which naturally incorporates a very desirable property named *scale invariance*, can only be correctly considered in the VB approximation to SBL. Note that one of the other approximation techniques in the literature also leads to the desired update equations under the same model assumptions, but as we show in detail, there are some inconsistencies involved in the derivations. There also exists a fast implementation of SBL in the context of *maximum marginal likelihood*, one of the most popular SBL approximation techniques. It is based on a component-wise decision rule for pruning or keeping a basis function. We show that an equivalent method can also be derived from a fixed-point analysis of the VB update expressions. Based on the slightly different new form, in which the fast decision rule is now given, we additionally show a rather straight forward way of how to incorporate a *sparsity vs. accuracy* trade-off parameter into the SBL framework.

As for the second contribution of this thesis, we propose and investigate two different methods for distributed regression in WSNs based on SBL. Sparsity is of particular interest in WSNs, because finding an effective representation of the data is very important due to the energy and bandwidth constraints discussed above. The two distributed algorithms are based on consensus protocols, which try to find a consensus between all sensors in the network. This makes them robust against changing network topologies without the need for routing. Note that this holds only as long as the network is connected, i.e., there always exists a multi-hop path between any two sensors in the network. We will not consider sensor failures, which remains an open research issue, because, even in centralized SBL, removing data during runtime is problematic. Note that the data on each sensor can be seen as a sample within a distributed dataset. We only investigate the distributed SBL methods

from a machine learning perspective and do not consider implementation aspects like energy consumption and so on. However, we do discuss the computational and communicational complexities involved in the two algorithms. One of the algorithms is based on a combination of VB inference and probabilistic message passing, where the latter is used for inter-sensor communication. The other algorithm can use any distributed averaging protocol to distributedly compute the fixed-points of the fast VB SBL method discussed above. Here, we show a mechanism of how new basis functions can be tested and incorporated into the model if they are relevant. We compare both distributed SBL algorithms to their corresponding centralized counterparts.

1.4 Related Work

In the following, we discuss related work in the field of SBL and distributed learning. To our knowledge, there exist no investigations on distributed SBL methods for field estimation in WSNs.

Let us first start with related work in centralized SBL. In [29], the author is concerned about the NP-hard problem of finding maximally sparse representations in overcomplete dictionaries of basis vectors, where the subtle difference between *sparse regression* vs. *sparse representation* is emphasized. In *sparse regression*, which is mainly the focus of this thesis, sparsity also acts as a regularization to avoid overfitting and to obtain compact, more interpretable models that generalize well on a test dataset. In *sparse representation*, there is no test dataset, and thus the only concern is to find a compact representation of the original data. In other words, overfitting the (training) data is a problem in *sparse regression*, whereas it is, to a certain extent, desired in *sparse representation*, depending on the noise included in the data. The algorithms that we will derive later in this thesis are applicable to both scenarios, where we, however, have more focus on *sparse regression*. That is, we evaluate the performance on a separate test dataset unless stated otherwise. In [29], SBL is analyzed as a potential approximation to the NP-hard problem of finding maximally sparse solutions. As has been shown, the global SBL minimum is always achieved at the maximally sparse solution if no noise is present, i.e., the model can perfectly represent the data. However, if noise is present, which we always consider for the case of *sparse regression* throughout the thesis, such an analysis is much more complicated. This is due to the fact that there exists no optimal sparseness anymore and a *sparsity vs. accuracy* trade-off needs to be specified for such an analysis.

In [53], an ℓ_1-norm SBL method is proposed. Note that this is different from the standard ℓ_2-norm SBL method [1] that we consider. The ℓ_1-norm SBL method is based on the maximization of the *marginal likelihood*, similar to a popular variant of ℓ_2-norm SBL. However, due to additionally required approximations, the overall computational costs increase. It is shown, that ℓ_1-norm SBL leads to sparser solutions than ℓ_2-norm SBL. However, as shown in the simulation results in [53], the resulting predictive error is typically larger. This is an example of the ubiquitous *sparsity vs. accuracy* trade-off in regression that we also discuss in Chapter 3, where we propose a separate trade-off control parameter for our fast VB version of ℓ_2-norm SBL.

In [54, 55], SBL is analyzed from a *Gaussian Process* (see, e.g., [56]) perspective. The counterintuitive behavior of the SBL prediction variance, namely that it is small for input regions with low sample densities, is corrected through an augmentation of the learned model. The presented extensions could be directly incorporated in our centralized fast VB SBL method as well. However, the computation of the modified prediction requires access to all training data samples, which is not directly applicable when the data is distributed as assumed for our distributed SBL approaches. Furthermore, because of the required access to the training data, the augmented model is not really sparse anymore as mentioned in [55].

In the context of RVMs, there are also efforts to adaptively learn the kernel parameters [57]. For a Gaussian kernel, the inverse width parameter θ_κ could be learned together with the SBL parameters for instance. This is not considered in this thesis but constitutes an interesting extension to our methods as well.

Let us now discuss related work in the area of distributed learning. The work [46] can be seen as a keynote introduction to research in the field of *distributed inference and learning* in WSNs; see also the related content in [52]. The discussed distributed kernel regression methods are closely related to our work. One approach, which was proposed in [58], considers the distributed computation of a sparse inverse problem in a WSN. There, a distributed junction tree algorithm is used to efficiently solve the inverse problem. The sparseness, which allows for the efficient distributed compution in this method, comes solely from the network structure and has nothing to do with model sparseness in the sense of SBL. The idea is to use localized kernels, where only sensors in the kernel support regions participate in the partial computations, i.e., where the kernel functions have non-zero values. In other words, this approach exploits the localized kernel structure to obtain localized communication and computations. Note that the neighborhood in the network topology is usually

strongly correlated with the spatial neighborhood of the localized kernels. In this method, the sensors can only give local predictions for their surrounding. Thus, prediction queries have to be relayed to the sensors close to the requested coordinates. Note, that this is different from our approaches, where each sensor finally obtains an overall model of the predictive function and is able to answer any query directly. Another approach discussed in [46] is an distributed incremental subgradient functional optimization algorithm based on [59, 60]. The method can be used with arbitrary kernels. However, it depends on a stable path through the network since it sequentially updates the current state of the predictive function. Thus, a query can only be answered properly by the last sensor in the path. Finally, the method of alternating projections from [61, 62] is discussed. There, sensors can update their current state of the predictive function through neighborhood communication. This is basically similar to our approaches. No path though the network is required, which makes this method robust against topology changes. However, as with the other two discussed methods, no sparsity in the sense of a compact model with a reduced number of kernels is considered.

Another related work that leads to sparse kernel models, is the distributed SVM regression algorithm [63]. Here, the efficient distributed implementation is only achieved through localized kernel functions that separate the optimizions performed at different parts of the network. Our framework, although being computationally more demanding, is more general and can be used with any set of basis functions.

The method [64] also estimates sparse kernel models, where sparsity is obtained through a sequential testing of kernels based on a kernel coherence criterion. This method performs this sequential test along a path of sensors, where the prediction requests have to be relayed to the last sensor in the chain. Furthermore, kernels are only added but never removed from the model, which makes it inflexible in finding a very sparse set of kernels. We will show later in this thesis how to adaptively add or remove basis functions in a non-sequential way bases on a distributed version of our fast VB method for SBL.

The work [65] is concerned about distributed field estimation in sensor networks. However, this is different from our interpretation of *field estimation*. The methods in [65] are not concerned about learning the overall field functions, but only its values at the sensor locations. More specifically, the presented methods try to smooth the noisy measurements distributedly at the sensors according to assumed spatial correlations. The first approach in [65] is based on a distributed consensus like protocol that implicitly projects the sensor measurements onto a lower dimensional

subspace spanned by basis vectors obtained from a set of smooth basis functions. The smoothed sensor measurements are directly estimated distributedly without explicitly learning the weights of the basis functions. The second approach in [65] is based on belief propagation performed on a Gaussian Markov random field within sensor clusters. The random field models the local dependencies between the sensor measurements. Note that in this thesis we are concerned about finding a compact set of basis functions, whereas the methods in [65] only estimate the values directly at the sensor locations.

In [66], a field reconstruction method based on *hybrid shift-invariant spaces* is presented. Simply speaking, a shift-invariant space is the space of functions generated as in (1.1), where identical but shifted generator basis functions are located at a uniform grid. A *hybrid shift-invariant space* is the space of functions generated by a superposition of functions from different shift-invariant spaces, i.e., with different generator functions. A *maximum likelihood* estimator is used for determining the weights w_m in (1.1). This estimator is not sparsity enforcing, i.e., it does not push irrelevant weights towards zero as in SBL. Furthermore, maximum likelihood estimators are known to overfit the data. In our methods, however, we avoid overfitting through automatic regularization caused by the sparsity parameters. An efficient distributed method with different sensor clusters is presented in [66]. This method requires, however, that the generator basis functions have finite support like in some of the previously discussed methods. In [66], spline functions are suggested for this purpose.

1.5 Outline of the thesis and publications

In Chapter 2, we start with a general introduction to variational Bayesian (VB) approximate inference and discuss several different sparse Bayesian learning (SBL) approximations. Here, we clarify the conceptual advantages of the VB approximation to SBL as compared to the other approximations. Furthermore, at the end of Chapter 2, we present *consensus propagation* (CP), a distributed averaging protocol proposed in [67]. We directly show the derivation of CP from a Gaussian *loopy belief propagation* (LBP) perspective. This derivation forms an important basis to understand the later derived distributed SBL methods presented in Chapter 4.

In Chapter 3, we present an alternative derivation of the fast SBL results obtained in the *fast marginal likelihood maximization* method [68]. Here, we show that the same decision rules for keeping or pruning a basis function can be derived from

a VB perspective. This chapter is in part based on our publication [69], where I was involved in the derivation of the algorithms and prepared the simulations. As opposed to [68], we have further extended the fast VB SBL method (FV-SBL) by an intuitive *sparsity vs. accuracy* trade-off parameter. Parts of the experimental analysis for this trade-off, shown in Chapter 3, stem from our publication [70], where my contribution was the idea and preparation of the *sparsity vs. accuracy* trade-off simulations.

In Chapter 4, we present a distributed SBL regression method that is based on a combination of VB inference and Gaussian LBP. The idea of the underlying probabilistic model is based on CP, introduced in Chapter 2. Parts of the content stem from our publication [71], which itself is a further development of our work [72]. My contribution to both publications was the idea and I was heavily involved in writing the main content and preparing the simulations. Additionally, the publication [71] was selected for the *best student paper award* of the conference where it was presented.

In Chapter 5, we develop a distributed SBL regression method using FV-SBL. We therefore extend the FV-SBL method presented in Chapter 3, in which basis functions are tested that are currently in the model, by using a simple keeping or pruning criterion. This extension, however, also allows the testing of new candidate basis functions, which can be added to the model as well. We present a way of how to compute these test criteria distributedly using distributed averaging protocols like CP.

We finally conclude and discuss new research directions in Chapter 6.

Note that we also present an online FV-SBL method in Appendix D, which is based on our publication [73], where I contributed substantially to writing the main content and preparing the simulations. We compare the method to a *kernel recursive least squares* (Kernel-RLS) algorithm [74].

Chapter 2

Background and Preliminaries

In this chapter we discuss the most relevant concepts needed to understand the methods developed later in this thesis.

In Section 2.1, variational Bayesian (VB) inference is introduced. We show that it offers a general framework for deterministic approximate inference, which is opposed to stochastic approximate methods based on sampling. We further show in this section that the well known *expectation maximization* (EM) algorithm can be interpreted as a special case of VB inference.

Next, in Section 2.2 we present sparse Bayesian learning (SBL), a probabilistic Bayesian regression method used in machine learning. Different SBL approximation techniques available in the literature are discussed. However, since there are conceptual inconsistencies involved in most of the methods, we will analyze and clarify these in detail. We show that the VB approach to SBL does not suffer from such inconsistencies and, therefore, we use it as the basis for all later developments in this thesis.

Finally, in Section 2.3, we give a short introduction on consensus propagation (CP), a distributed averaging protocol used in sensor networks. CP is important to understand the derivations of a distributed VB SBL method developed later in Chapter 4. Thus, this section introduces a framework that allows the derivation of a distributed algorithm based on the methods discussed in Section 2.1 and Section 2.2. The derivations of CP are presented as an application of the sum-product algorithm on a loopy Gaussian factor graph, which leads to loopy belief propagation (LBP).

2.1 Variational Bayesian inference

Consider a probabilistic model defined by the joint probability $p(\mathcal{H}, \mathcal{O})$ over the set of *hidden* random variables $\mathcal{H}$ and *observed* random variables $\mathcal{O}$. Typically, from such a model we would like to obtain the posterior $p(\mathcal{H}|\mathcal{O})$, i.e., we would like

to make inference about $\mathcal{H}$ given $\mathcal{O}$. As widely known, the posterior $p(\mathcal{H}|\mathcal{O})$ can be obtained using Bayes' theorem $p(\mathcal{H}|\mathcal{O}) = p(\mathcal{O}|\mathcal{H})p(\mathcal{H})/p(\mathcal{O}) = p(\mathcal{H},\mathcal{O})/p(\mathcal{O})$, but unfortunately the computations are infeasible for many practical models. For continuous random variable models this is due to the intractable[1] integral in the normalization term $p(\mathcal{O}) = \int p(\mathcal{H},\mathcal{O})\, d\mathcal{H}$, which is referred to as the model evidence. High dimensional and complex integrands make numerical integration impractical and especially in such cases alternative approximate inference methods need to be found. For discrete random variable models, although in principle always tractable, the number of hidden states in the marginalization sum can be prohibitively large in many practical models [4]. Note that although the presented concepts hold for continuous and discrete random variables we restrict all analyses in this thesis on the continuous case.

Basically there exist two types of approximation schemes to approximate the intractable posterior $p(\mathcal{H}|\mathcal{O})$. The first scheme is based on stochastic approximation, where sampling methods like Markov chain Monte Carlo [75] are used. These methods could theoretically lead to the exact posterior densities given unlimited computational resources [4]. However, considering limited resources, real world implementations often lead to useless approximations, especially for high dimensional distributions.

The second scheme is based on deterministic approximations, where the intractable posterior $p(\mathcal{H}|\mathcal{O})$ is approximated by a pdf $q(\mathcal{H})$ that typically is of a simpler structure than the posterior. That is,

$$q(\mathcal{H}) \approx p(\mathcal{H}|\mathcal{O}), \tag{2.1}$$

where for simplicity of notation we omit writing the dependence on the observed variables $\mathcal{O}$ in the approximation $q(\mathcal{H})$.[2] The approximation need to be as close as possible to the intractable posterior $p(\mathcal{H}|\mathcal{O})$ according to some distance measure.

Variational Bayesian inference [34, 4, 76], sometimes just termed *Variational Bayes* (VB), is such a deterministic approximation scheme and will be explained in the following. Another example for a deterministic approximation scheme is *expectation propagation* [77].

[1] This means that no closed-form analytical solution exists.

[2] Note that $q(\mathcal{H})$ indirectly depends on $\mathcal{O}$ since it approximates the posterior that obviously depends on $\mathcal{O}$. The compact form of notation $q(\mathcal{H})$ is common in the literature; see, e.g., [34, 4].

2.1.1 Variational lower bound

It is easy to show that the logarithm of the intractable model evidence can be decomposed [4] as

$$\ln p(\mathcal{O}) = \mathcal{L}(q) + \mathrm{KL}(q\|p), \tag{2.2}$$

where we define the variational lower bound functional $\mathcal{L}(q)$ and the Kullback-Leibler divergence $\mathrm{KL}(q\|p)$ as follows:[3]

$$\mathcal{L}(q) = \int q(\mathcal{H}) \ln \left\{ \frac{p(\mathcal{H}, \mathcal{O})}{q(\mathcal{H})} \right\} d\mathcal{H}, \tag{2.3}$$

$$\mathrm{KL}(q\|p) = \int q(\mathcal{H}) \ln \left\{ \frac{q(\mathcal{H})}{p(\mathcal{H}|\mathcal{O})} \right\} d\mathcal{H}. \tag{2.4}$$

From a VB perspective, the best approximating distribution $q^*(\mathcal{H})$ is the one that minimizes the KL-divergence given in (2.4). As by definition of a KL-divergence, $\mathrm{KL}(q\|p) \geq 0$ for all valid densities q and p, the best approximation with $\mathrm{KL}(q^*\|p) = 0$ is achieved if and only if $q^*(\mathcal{H}) = p(\mathcal{H}|\mathcal{O})$. Unfortunately, if $p(\mathcal{H}|\mathcal{O})$ is intractable, we cannot minimize (2.4) directly. The way out of this problem lies in Equation (2.2). Because $\ln p(\mathcal{O})$ is constant with respect to q and $\mathrm{KL}(q\|p) \geq 0$, maximizing $\mathcal{L}(q)$ is equivalent to minimizing $\mathrm{KL}(q\|p)$ with respect to q. This is illustrated in Figure 2.1. The functional $\mathcal{L}(q)$ is typically denoted as the *variational lower bound*, since it is a lower bound on $\ln p(\mathcal{O})$. The obvious advantage of optimizing over $\mathcal{L}(q)$ instead of the KL-divergence, is that it is tractable while it still solves the same problem.

The reason, the whole discussed approach is denoted as *Variational Bayes*, lies in the fact that functionals are optimized here and the mathematical field dealing with such problems is named *calculus of variations*; see, e.g., [78]. Even though in our analysis we will not use typical methods of *calculus of variations* like the *Euler-Lagrange equation*, equivalent results that are based on such analysis can be found for instance in [34].

Minimizing the divergence $\mathrm{KL}(q\|p)$ like in (2.4) is only one approach of obtaining a deterministic approximation of the posterior. *Expectation propagation* [77, 4] for instance aims to minimize the reversed KL-divergence $\mathrm{KL}(p\|q)$. Both KL-divergences $\mathrm{KL}(q\|p)$ and $\mathrm{KL}(p\|q)$ are special cases of the broader alpha family of divergences [79]. It can be shown that VB focuses the propability mass of the approximation q

[3] Note that the arguments q and p in $\mathcal{L}(q)$ and $\mathrm{KL}(q\|p)$ are short form notations for the approximating distribution $q(\mathcal{H})$ and the true posterior $p(\mathcal{H}|\mathcal{O})$ respectively. Thus $\mathcal{L}(\cdot)$ and $\mathrm{KL}(\cdot\|\cdot)$ are both functionals, i.e., they take functions as arguments.

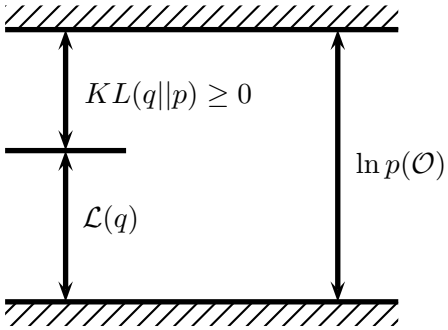


Figure 2.1: Maximizing $\mathcal{L}(q)$ with respect to q is equivalent to minimizing $KL(q\|p)$ because the sum of both terms given in (2.2) does not depend on q and the KL-divergence cannot get negative.

at regions with high probability mass of the true posterior p, whereas *expectation propagation* approximates it over a broader region including areas with very low probability [79, 4]. This property makes VB a very good choice when dealing with multimodal densities as we typically have in SBL which is introduced in Section 2.2.

2.1.2 Mean field approximation

Up to now we have made no assumptions about the form of $q(\mathcal{H})$. For many practical problems with numerous hidden variables, it turns out that it is convenient to separate $\mathcal{H}$ into disjoint subsets $\{\mathcal{H}_1, \ldots, \mathcal{H}_S\}$ and assume that $q(\mathcal{H})$ factorizes as

$$q(\mathcal{H}) = \prod_{i=1}^{S} q_i(\mathcal{H}_i). \tag{2.5}$$

This factorization is called *mean field approximation* if all individual hidden variables fully factorize[4] and *structured mean field approximation* if non-fully factorized subsets of $\mathcal{H}$ are used. This approach has its origin in physics [4]. Inserting (2.5) into the variational lower bound functional (2.3) and separating the terms that depend on a particular factor $q_l(\mathcal{H}_l)$ from the others, denoted as $q_{\bar{l}}(\mathcal{H}_{\bar{l}}) = \prod_{i \neq l} q_i(\mathcal{H}_i)$, which are considered as constants, gives:[5]

$$\mathcal{L}(q) = \mathcal{L}(q_l q_{\bar{l}}) = \int \prod_{i=1}^{S} q_i \left\{ \ln p(\mathcal{H}, \mathcal{O}) - \sum_{j=1}^{S} \ln q_j \right\} d\mathcal{H}$$

[4] In the literature this is also known as singleton or unit subset; see, e.g., [80, Section 1.8.1].
[5] Similar as before, we write q_l and $q_{\bar{l}}$ instead of $q_l(\mathcal{H}_l)$ and $q_{\bar{l}}(\mathcal{H}_{\bar{l}})$ respectively.

$$
\begin{aligned}
&= \int q_l \left\{ \int q_{\bar{l}} \, \ln p(\mathcal{H}, \mathcal{O}) \, d\mathcal{H}_{\bar{l}} \right\} d\mathcal{H}_l - \int q_l \, \ln q_l \, d\mathcal{H}_l + const. \\
&= \int q_l \, \mathbb{E}_{q_{\bar{l}}} \Big\{ \ln p(\mathcal{H}, \mathcal{O}) \Big\} d\mathcal{H}_l - \int q_l \, \ln q_l \, d\mathcal{H}_l + const. \\
&= \int q_l \, \ln \left\{ \frac{\exp\left(\mathbb{E}_{q_{\bar{l}}} \Big\{ \ln p(\mathcal{H}, \mathcal{O}) \Big\} \right)}{q_l} \right\} d\mathcal{H}_l + const. \\
&= -\mathrm{KL}(q_l \| \tilde{p}_l) + const., \qquad (2.6)
\end{aligned}
$$

where $\mathbb{E}_{q_{\bar{l}}}\{\cdot\}$ is the expectation operator with respect to the distribution $q_{\bar{l}}$. We have to assure that $\tilde{p}_l$ is a normalized distribution and thus the maximizing solution of $\mathcal{L}(q)$ with respect to q_l is

$$
q_l^*(\mathcal{H}_l) = \tilde{p}_l(\mathcal{H}_l) = \frac{\exp\left(\mathbb{E}_{q_{\bar{l}}}\{\ln p(\mathcal{H}, \mathcal{O})\} \right)}{\int \exp\left(\mathbb{E}_{q_{\bar{l}}}\{\ln p(\mathcal{H}, \mathcal{O})\} \right) d\mathcal{H}_l}, \qquad (2.7)
$$

since this is the only case the negative KL-divergence term in (2.6) vanishes. All other $q_l \neq \tilde{p}_l$ would lead to a lower bound $\mathcal{L}(q)$ that is less tight. Since in (2.7) we wrote $q_l = \tilde{p}_l$, the optimal solution is assumed to be achievable, which is known as the unconstrained factor update. If we however restrict q_l to a certain family of probability densities $q_l \in \mathcal{Q}_l$ with $\tilde{p}_l \notin \mathcal{Q}$, we need to optimize (2.6) under this constraint. An example for such a case is given later in Section 2.1.3.

Because the expectation in (2.7) depends on all other factors than q_l, i.e., q_i for $i \neq l$, the expression is not explicit and we need to update all factors in any order to converge to a local optimum of $\mathcal{L}(q)$. Also, since this is an iterative method, we need to initialize the factors q_i for $i \neq l$ at the beginning if we start by updating the factor q_l first. Convergence of mean field (or structured mean field) VB inference to a local maximum of $\mathcal{L}(q)$ is guaranteed, since the individual updates (2.6) are concave with respect to each q_l, $l = 1, \ldots, S$. This is because the KL-divergence in general and in particular in (2.6) is convex with respect to both of its arguments [81]. Note that due to the non-convex set of factorized distributions (factorization constraint)[6], *convergence* of *mean field VB* can only be guaranteed to a local maximum of $\mathcal{L}(q)$ in general.

[6]The non-convexity of the set can be shown from the convex combination $\theta \prod_i q_i^A(\mathcal{H}_i) + (1 - \theta) \prod_i q_i^B(\mathcal{H}_i) = q^C(\mathcal{H})$ for $0 \leq \theta \leq 1$, where $q^C(\mathcal{H})$ does not factorize ifself in general, i.e., it is not of the form $\prod_i q_i^C(\mathcal{H}_i)$; cf. [81].

2.1.3 Expectation-Maximization as a special case of mean field variational Bayes

In this section we show that the well known EM algorithm [82, 83, 4] is a special case of variational Bayesian inference using a structured mean field approximation over the hidden variables $\mathcal{H} = \{\boldsymbol{\theta}, \tilde{\mathcal{H}}\}$, where we have separated $\mathcal{H}$ into some parameters of interest $\boldsymbol{\theta}$ and the remaining hidden variables $\tilde{\mathcal{H}}$ [84]. Note that in this VB interpretation of the EM algorithm also the parameters of interest $\boldsymbol{\theta}$ are considered as hidden random variables.

However, before we start with the VB interpretation of the EM, we first start with its common definition to make all further analysis clear. Basically, the EM algorithm is defined for *maximum likelihood* (ML) as well as for *maximum a posteriori* (MAP) estimation of some parameters $\boldsymbol{\theta}$ given $p(\tilde{\mathcal{H}}, \mathcal{O}|\boldsymbol{\theta})$ or $p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta})$ respectively. In the ML version we are looking for $\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta}}\, p(\mathcal{O}|\boldsymbol{\theta})$, where in the MAP version we are interested in $\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \mathrm{argmax}_{\boldsymbol{\theta}}\, p(\boldsymbol{\theta}|\mathcal{O}) = \mathrm{argmax}_{\boldsymbol{\theta}}\, p(\mathcal{O}|\boldsymbol{\theta})p(\boldsymbol{\theta})$. As before, we use $\mathcal{O}$ to denote the set of observed random variables, where in the context of EM this is also known as *incomplete data*. The set $\tilde{\mathcal{H}}$ denotes hidden random variables and the union $\{\tilde{\mathcal{H}}, \mathcal{O}\}$ is termed *complete data* [4]. Since we have no access to the *complete data* because $\tilde{\mathcal{H}}$ is hidden, the EM algorithm is based on computing expectations under the posterior of the hidden variables. Note that in this common form of the EM algorithm, $\boldsymbol{\theta}$ is only considered to be a random variable in the MAP version, where a prior $p(\boldsymbol{\theta})$ needs to be specified.

We summarize the E- and M-steps [4] for both versions of the EM-algorithm below, where the steps need to be iterated until convergence.

EM for ML estimation	**EM for MAP estimation**
Goal: maximize $p(\mathcal{O}\|\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$	**Goal:** maximize $p(\boldsymbol{\theta}\|\mathcal{O})$ w.r.t. $\boldsymbol{\theta}$
E-step:	**E-step:**
Determine: $q(\tilde{\mathcal{H}}) = p(\tilde{\mathcal{H}}\|\mathcal{O}, \hat{\boldsymbol{\theta}}^{\mathrm{old}}_{\mathrm{ML}})$	Determine: $q(\tilde{\mathcal{H}}) = p(\tilde{\mathcal{H}}\|\mathcal{O}, \hat{\boldsymbol{\theta}}^{\mathrm{old}}_{\mathrm{MAP}})$
Define: $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}^{\mathrm{old}}_{\mathrm{ML}}) = \mathbb{E}_{q(\tilde{\mathcal{H}})}\{\ln p(\tilde{\mathcal{H}}, \mathcal{O}\|\boldsymbol{\theta})\}$	Define: $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}^{\mathrm{old}}_{\mathrm{MAP}}) = \mathbb{E}_{q(\tilde{\mathcal{H}})}\{\ln p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta})\}$
M-step:	**M-step:**
Update: $\hat{\boldsymbol{\theta}}^{\mathrm{new}}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta}}\, Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}^{\mathrm{old}}_{\mathrm{ML}})$	Update: $\hat{\boldsymbol{\theta}}^{\mathrm{new}}_{\mathrm{MAP}} = \mathrm{argmax}_{\boldsymbol{\theta}}\, Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}^{\mathrm{old}}_{\mathrm{MAP}})$

We now start with the VB interpretation of the MAP-EM. Since we can interpret the ML-EM as a MAP-EM with flat priors $p(\boldsymbol{\theta}) \propto 1$, which can be easily seen from Bayes' theorem, it is enough to show that MAP-EM is a special case of mean field VB. That is, if the latter can be shown for MAP-EM, it also holds for ML-EM.

Consider a probabilistic model given by the joint distribution $p(\mathcal{H}, \mathcal{O}) = p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta})$, where the hidden variables $\mathcal{H}$ from our VB analysis are now separated into two sets $\boldsymbol{\theta}$ and $\tilde{\mathcal{H}}$ as we have already mentioned above. The variational posterior approximation $q(\mathcal{H})$ is assumed to factorize as

$$q(\mathcal{H}) = q(\tilde{\mathcal{H}})q(\boldsymbol{\theta}). \tag{2.8}$$

From (2.7) we can obtain the optimal unconstrained variational updates for $q(\tilde{\mathcal{H}})$ and $q(\boldsymbol{\theta})$, respectively, as

$$\tilde{p}(\tilde{\mathcal{H}}) = \frac{\exp\left(\mathbb{E}_{q_{\boldsymbol{\theta}}}\{\ln p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta})\}\right)}{\int \exp\left(\mathbb{E}_{q_{\boldsymbol{\theta}}}\{\ln p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta})\}\right) d\tilde{\mathcal{H}}} \tag{2.9}$$

and

$$\tilde{p}(\boldsymbol{\theta}) = \frac{\exp\left(\mathbb{E}_{q_{\tilde{\mathcal{H}}}}\{\ln p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta})\}\right)}{\int \exp\left(\mathbb{E}_{q_{\tilde{\mathcal{H}}}}\{\ln p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta})\}\right) d\boldsymbol{\theta}}. \tag{2.10}$$

By taking the unconstrained solution (2.9) for $q(\tilde{\mathcal{H}})$ while constraining the approximative parameter density $q(\boldsymbol{\theta})$ to a Dirac delta point mass distribution $q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$ centered on $\hat{\boldsymbol{\theta}}$, we will show that the MAP-EM algorithm is a special case of VB inference assuming the factorization (2.8). Using such a point estimate leads to a deterministic interpretation of $q(\boldsymbol{\theta})$, since there is no more uncertainty involved in the outcome if samples were drawn from it.

Let us now start with the unconstrained update $q(\tilde{\mathcal{H}}) = \tilde{p}(\tilde{\mathcal{H}})$ given in (2.9). This update corresponds in part to the E-step of the EM algorithm. From (2.9), we see that the update depends on $q(\boldsymbol{\theta})$ via the expectation, where $q(\boldsymbol{\theta})$ is from the previous iteration since in VB, as discussed in Section 2.1.2, we need to iterate over the factors until convergence. We define $q(\boldsymbol{\theta})$ from the previous iteration as $\delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{\text{old}})$ with center $\hat{\boldsymbol{\theta}}^{\text{old}}$. Note that we need to specify some initial $\hat{\boldsymbol{\theta}}^{\text{old}}$ at the beginning.

When we now consider the expectation term from (2.9)

$$\mathbb{E}_{q_{\boldsymbol{\theta}}}\{\ln p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta})\} = \int \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{\text{old}}) \ln p(\tilde{\mathcal{H}}, \mathcal{O}, \boldsymbol{\theta}) d\boldsymbol{\theta} = \ln p(\tilde{\mathcal{H}}, \mathcal{O}, \hat{\boldsymbol{\theta}}^{\text{old}})$$

we obtain

$$q(\tilde{\mathcal{H}}) = \frac{p(\tilde{\mathcal{H}}, \mathcal{O}, \hat{\boldsymbol{\theta}}^{\text{old}})}{\int p(\tilde{\mathcal{H}}, \mathcal{O}, \hat{\boldsymbol{\theta}}^{\text{old}}) d\tilde{\mathcal{H}}} = p(\tilde{\mathcal{H}}|\mathcal{O}, \hat{\boldsymbol{\theta}}^{\text{old}}), \tag{2.11}$$

which is the hidden posterior given the old parameters. This is a part of the E-step

for both, the ML- and MAP-EM algorithm as shown above.

Since we have constrained the update of $q(\boldsymbol{\theta})$, the result (2.10) cannot be used directly as solution and we need to resort to the optimization problem (2.6) under the Dirac delta constraint. Finding this optimum reduces to the problem of finding the point mass center, which we denote $\hat{\boldsymbol{\theta}}^{\text{new}}$, since this is the only degree of freedom of $q(\boldsymbol{\theta})$. This parameter can be obtained as

$$\begin{aligned}\hat{\boldsymbol{\theta}}^{\text{new}} &= \underset{\hat{\boldsymbol{\theta}}}{\text{argmin}}\ \text{KL}(\ \delta(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})\|\tilde{p}(\boldsymbol{\theta})\ ) = \underset{\hat{\boldsymbol{\theta}}}{\text{argmax}}\ \ln\tilde{p}(\hat{\boldsymbol{\theta}}) \\ &= \underset{\hat{\boldsymbol{\theta}}}{\text{argmax}}\ \mathbb{E}_{q_{\tilde{\mathcal{H}}}}\{\ln p(\tilde{\mathcal{H}},\mathcal{O},\hat{\boldsymbol{\theta}})\}, \end{aligned} \tag{2.12}$$

where we have inserted the Dirac delta distribution into the KL-divergence and solved the integral. Equations (2.11) and (2.12) together form the E- and M-steps of the MAP-EM (and also for the ML-EM algorithm if flat priors $p(\boldsymbol{\theta}) \propto 1$ where used as discussed before). To see the equivalence between both, the E- and M-steps, we can introduce an intermediate function $Q(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}^{\text{old}})$ for the expectation in (2.12) and make it a part of the E-step. Note that this does not change anything from a mathematical point of view but separates the expectation in the E-step from the maximization in the M-step.

2.2 Sparse Bayesian learning

In this section we introduce SBL, a Bayesian supervised learning method for regression, which aims at finding a function that maps arbitrary inputs $\boldsymbol{x} \in \mathbb{R}^D$ to targets $t \in \mathbb{R}$ based on some training data $\mathcal{D} = \{\boldsymbol{x}_n, t_n\}_{n=1}^N$. This function, which we denote $y(\boldsymbol{x}, \boldsymbol{w})$, depends on a set of parameters $\boldsymbol{w}$, where $\boldsymbol{w}$ is a vector. The individual elements of $\boldsymbol{w}$ are only relevant for the shape of the function $y(\boldsymbol{x}, \boldsymbol{w})$ over $\boldsymbol{x}$ if they have non-zero values. The term sparse in SBL refers to finding a model that has many zero elements in $\boldsymbol{w}$, i.e., we would like to find a sparse vector $\boldsymbol{w}$. Such models are computationally cheaper to evaluate and are typically less prone to overfit the training data as compared to complex models [85]. In an SBL model, this sparsity is achieved by defining a special kind of prior over $\boldsymbol{w}$, namely one that leads to a posterior which is highly peaked at zero for such irrelevant components. We will describe this mechanism later in more detail.

The next subsections are organized as follows. Section 2.2.1 defines the SBL model. In Section 2.2.2, we summarize different methods proposed in the literature

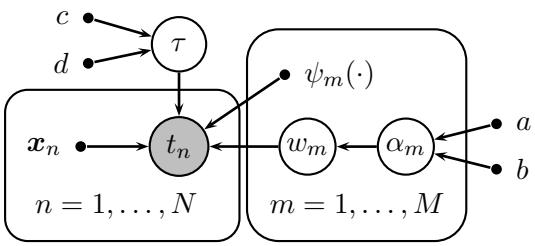


Figure 2.2: A Bayesian network representing the Sparse Bayesian learning model.

to tackle the problem of approximate SBL inference, since there exists no closed form solution for SBL inference. Some of these methods involve inconsistencies in their derivations related to the original model definitions given in [1], which we also present in Section 2.2.1. We discuss these inconsistencies in Section 2.2.3 and resolve why they, despite of their slightly different derivations and model parameterizations, still lead to the same algorithms after all. Based on this discussion, we also give reasons why the focus of this work is on the VB approach to SBL and not on the other presented methods. Finally, in Section 2.2.4 we shortly discuss how sparsity even arises from the presented probabilistic framework.

2.2.1 Model definition

Consider the probabilistic model represented as a Bayesian network (BN) in Figure 2.2. Basically, a BN is a graphical model representing the conditional dependencies between random variables in the form of a directed acyclic graph.[7] In our work, we adhere to the notation given in [4], where we use shaded nodes to denote observed random variables, unshaded nodes to represent hidden random variables and dots to indicate deterministic parameters. Additionally, we find it convenient to also represent deterministic functions using the dot notation. This is not common in the literature but it allows us to include all parts of the SBL model into a single BN.

In such a model, we would like to make inference about the hidden variables given the observations t_n corresponding to inputs $\boldsymbol{x}_n$, $n = 1, \ldots, N$, which are given from the training data $\mathcal{D}$. After learning the pdfs of the hidden variables given the observations, we can make predictions about some t^* given any arbitrary input $\boldsymbol{x}^*$ as will be discussed in Section 2.2.2.

[7] For more information on BNs we refer to standard textbooks on probabilistic inference and machine learning, e.g., [4, 20, 21].

The SBL model depicted in Figure 2.2, represents the joint probability

$$p(\boldsymbol{t}, \tau, \boldsymbol{w}, \boldsymbol{\alpha}) = p(\boldsymbol{t}|\tau, \boldsymbol{w})\, p(\boldsymbol{w}|\boldsymbol{\alpha})\, p(\boldsymbol{\alpha})\, p(\tau), \tag{2.13}$$

where we have used the vector notations $\boldsymbol{t} = [t_1, \ldots, t_N]^T$, $\boldsymbol{w} = [w_1, \ldots, w_M]^T$ and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_M]^T$ to denote targets, weights, and hyperparameters, respectively. The model of the form (2.13) was first proposed in [1] although it originates from [23], where $\boldsymbol{\alpha}$ and τ were modeled as deterministic parameters without priors. The exact meaning of the individual variables will become clear in the next paragraphs, when we define the individual pdfs given in (2.13).

In (2.13), and also in the following pdf notations, we have only included the random variables and not the deterministic parameters for a compact notation. This is common in the literature, since in a fully Bayesian treatment the fixed deterministic parameters are important to define the individual pdfs but have no importance in the general Bayesian analysis. The exact influence of these parameters on the model densities will also be shown in the following.

The first term in (2.13) is defined as $p(\boldsymbol{t}|\tau, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{y}, \tau^{-1}\boldsymbol{I})$, a multivariate Gaussian pdf with mean vector $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{w}$ and covariance matrix $\tau^{-1}\boldsymbol{I}$, where τ is denoted as noise precision (inverse noise variance). The matrix $\boldsymbol{\Phi}$ is called *design matrix* and its elements are given as $\Phi_{nm} = \psi_m(\boldsymbol{x}_n)$, where $\psi_m(\cdot)$ is a basis function from a dictionary of M predefined fixed basis functions and $\boldsymbol{x}_n$ is an input sample from the training data $\mathcal{D}$. Thus, the design matrix $\boldsymbol{\Phi}$ only depends on the dictionary of basis functions evaluated at the inputs from the training data. The elements y_n in $\boldsymbol{y} = [y_1, \ldots, y_N]^T$ are therefore given as

$$y_n = y(\boldsymbol{x}_n, \boldsymbol{w}) = \sum_{m=1}^{M} w_m \psi_m(\boldsymbol{x}_n), \quad n = 1, \ldots, N, \tag{2.14}$$

where the parameters $\boldsymbol{w}$ are the weights for the basis functions, which we further denote as *weight parameters* or simply as *weights*. The function $y(\boldsymbol{x}, \boldsymbol{w})$ used in (2.14) is the regression function that aims to map inputs $\boldsymbol{x}$ to targets t as we mentioned at the beginning of this section. The pdf $p(\boldsymbol{t}|\tau, \boldsymbol{w})$ is denoted as observation likelihood, since it is a likelihood function of the noise precision τ and the weight parameters $\boldsymbol{w}$ for given targets $\boldsymbol{t}$.

Based on the above definition of the observation likelihood, the observed targets $\boldsymbol{t}$ can be interpreted as noisy[8] versions of the regression function $y(\boldsymbol{x}, \boldsymbol{w})$ evaluated at

[8] Here, we use the term *noise* to denote a perturbation which incorporates both, potential model

all inputs $\boldsymbol{x}_n$, i.e.,

$$t_n = y(\boldsymbol{x}_n, \boldsymbol{w}) + \epsilon_n, \tag{2.15}$$

where the $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \tau^{-1})$ are *i.i.d.* Gaussian noise samples.

If no prior is defined over the weights $\boldsymbol{w}$, finding the best function y would mean finding the most likely weights by maximizing the likelihood $p(\boldsymbol{t}|\tau, \boldsymbol{w})$ with respect to $\boldsymbol{w}$. This ML result is well known in the literature and is equivalent to the least squares result which minimizes $\|\boldsymbol{t}-\boldsymbol{y}\|^2$ over $\boldsymbol{w}$. The ML result $\boldsymbol{w}_{\mathrm{ML}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$ is independent of the noise precision and thus there is no need to estimate τ in this case. ML is known to lead to severe overfitting in many practical situations [4]. Overfitting can be avoided by introducing a prior over the weights, which, under some conditions, can also encourage sparse solutions [29]. We use the term *sparse* to denote solutions $\boldsymbol{w}^*$ which have a small number of non-zero elements, i.e., the ℓ_0-quasinorm $\|\boldsymbol{w}^*\|_0$, which gives the number of non-zeros in $\boldsymbol{w}^*$, is small. From (2.14) it can be seen, that weights which are zero have no influence on the model and thus the corresponding basis functions can be pruned.

As the set of basis functions $\psi_m(\cdot)$ can be chosen freely, one can also make them depend on the data. Such methods are called non-parametric and can result in highly flexible models as we already discussed in Chapter 1. The relevance vector machine (RVM) [23, 1] is such a special case of SBL and uses *kernel functions*[9] $\psi_n(\boldsymbol{x}) = \kappa(\boldsymbol{x}, \boldsymbol{x}_n)$ (located on the input data samples) beside a constant bias term, thus resulting in $M = N + 1$ basis functions.

Now, coming back to the SBL model definition, the overall weight prior as shown in Figure 2.2 is of a hierarchical nature. That is, the prior $p(\boldsymbol{w}|\boldsymbol{\alpha})$, which we also denote as *conditional weight prior*, depends on another random variable $\boldsymbol{\alpha}$ that itself has a prior distribution $p(\boldsymbol{\alpha})$, which we denote as *hyper prior*. Note that all distributions used in SBL belong to the exponential family of distributions[10] and all priors are defined as conjugate priors[11]. We define the conditional weight prior as

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{m=1}^{M} \mathcal{N}(w_m|0, \alpha_m^{-1}), \tag{2.16}$$

which is a zero mean Gaussian distribution with precision (inverse variance) param-

mismatches and real measurement noise. Later in Chapter 3, Section 3.3.3 we analyze both effects separately.

[9] For more information on kernels, we refer to standard literature like [11].

[10] For a general definition of exponential family distributions, see standard textbooks, e.g., [4].

[11] If a posterior distribution belongs to the same family of distributions as the prior when Bayes' theorem is used, the prior is called a conjugate prior with respect to the likelihood.

eters α_m for $m = 1, \ldots, M$. We denote $\alpha_1, \ldots, \alpha_M$ as the hyperparameters.

The hyperprior $p(\boldsymbol{\alpha})$ and the prior for the noise precision $p(\tau)$ are defined as

$$p(\boldsymbol{\alpha}) = \prod_{m=1}^{M} \mathrm{Ga}(\alpha_m | a, b) \tag{2.17}$$

and

$$p(\tau) = \mathrm{Ga}(\tau | c, d), \tag{2.18}$$

where $\mathrm{Ga}(z|\eta, \lambda) = \frac{\lambda^\eta}{\Gamma(\eta)} z^{\eta-1} \exp(-\lambda z)$ is a *gamma distribution*[12] over $z \geq 0$ with $\eta > 0$, $\lambda > 0$ and the *gamma function* $\Gamma(\eta) = \int_0^\infty \nu^{\eta-1} \exp(-\nu)\, d\nu$. Note that a *gamma distribution* is a conjugate prior for the precision parameter of a Gaussian likelihood which is the case for both α_m and τ.

Although a gamma distribution $\mathrm{Ga}(z|\eta, \lambda)$ is defined for $\eta > 0$ and $\lambda > 0$, the limit case $\eta \to 0$, $\lambda \to 0$ is of particular interest in this thesis. Setting a and b in (2.17) to zero, leads to a *Jeffreys prior* $p(\alpha_m) \propto 1/\alpha_m$ for the hyperparameters. A Jeffreys prior [87] is an objective *non-informative* prior that is *invariant under reparameterization*; cf. [88, 89, 90]. A *non-informative prior* is a prior which is indented to have as little influence on the posterior distribution as possible [4]. It is thus a good choice if no prior information is available. *Invariance under reparameterization* means that if we consider two likelihoods $p(x|z)$ and $p(x|z')$, which only differ in the parameterization z and z', where z' is a transformation of z defined as $z' = g(z)$ using an invertible function $g(\cdot)$, the priors $p(z)$ and $p(z')$ should lead to the same posterior $p(z|x)$, no matter if directly computed with $p(z)$ or retransformed from $p(z'|x)$ obtained with $p(z')$.[13] Such priors can be objectively derived from the corresponding likelihood function using Jeffreys' rule [87].

The SBL model, as defined in [1], is specified with non-informative hyperparameters and non-informative noise precision, i.e., also $p(\tau)$ is a Jeffreys prior defined as $p(\tau) \propto 1/\tau$ which can be obtained by setting c and d in (2.18) to zero. In [1] it is stated that such priors lead to scale invariance in the SBL model, i.e., predictions are independent of linear scaling of both $\boldsymbol{t}$ and the basis functions $\psi_m(\cdot)$. This is a pleasing property, since the results do not depend on the unit of measurement. Such priors are flat over a logarithmic scale, i.e., $p(\ln \alpha_m) \propto 1$ and $p(\ln \tau) \propto 1$. Intu-

[12] For more information about the *gamma distribution* see, e.g., [86, 4]. Note that the exact definitions are slightly different in the literature. In [4] for instance, $\mathrm{Ga}(z|\eta, \lambda)$ is defined for $z > 0$, whereas in [86] it is defined for $z \geq 0$. Note that throughout the thesis we only consider the latter definition.

[13] Note that for the latter we need to apply the transformation rule for random variables [91].

itively, this means that the prior probability of values between 1 and 10 is the same as between 10 and 100 and so on. We also note that even though the Jeffreys prior is an improper prior, i.e., it does not integrate to 1, it is reasonable and widely used in cases where the posterior has a proper form[14]. In Appendix A.1, we formally derive the scale-invariant priors $p(\boldsymbol{\alpha}) \propto \prod_{m=1}^{M} \frac{1}{\alpha_m}$ and $p(\tau) \propto 1/\tau$ from the scale invariance assumption of the SBL prediction with respect to scaling the targets $\boldsymbol{t}$. The predictive scale invariance with respect to scaling the basis functions $\psi_m(\cdot)$ can be shown in a similar way.

Sparsity within models of the form (1.1), which are linear in the parameters, can be obtained through prior distributions $p(w_m)$ that have flat tails and a sharp peak at zero [29, 92]. Such priors are also denotes as *sparsity-inducing priors*. In SBL, we have not directly specified $p(w_m)$, but it can be obtained through marginalization, i.e., by integrating over the hyperparameters of the joint prior $p(w_m, \alpha_m) = p(w_m|\alpha_m)p(\alpha_m)$. We denote the resulting prior $p(w_m)$ as the *effective weight prior*. Note the difference to the *conditional weight prior* $p(w_m|\alpha_m)$. The effective SBL weight prior can be computed as

$$p(w_m) = \int p(w_m|\alpha_m)p(\alpha_m)d\alpha_m \propto \left(b + \frac{w_m^2}{2}\right)^{-(a+\frac{1}{2})}, \tag{2.19}$$

which has the form of a *Student's-t* distribution; cf. [1]. In Figure 2.3, we show different settings for a and b compared to a Gaussian distribution. Smaller a and b lead to peakier distributions around zero. That is, in the limit case $a = b = 0$, which we consider in SBL, $p(w_m) \propto 1/|w_m|$ is strongly sparsity inducing. However, note that $p(w_m)$ is improper for this case. Later in Section 2.2.4, we further analyze in detail how sparsity is obtained within the SBL model.

The reason that we included the known parameters a, b, c, d in the first place, albeit it turned out that all are considered to be 0, was to show their influence on the model. This general form is important for further analysis presented in the next sections, where we discuss another improper setting widely used in the literature, namely $a = c = 1$ and $b = d = 0$. Furthermore, working with improper priors should be done with care, as they are per definition no valid probability densities. In Appendix B, we show for the particular case of *fast variational SBL* (later derived in Section 3.2), that instead of setting $a = b = 0$ in advance, we can also obtain equivalent results by computing the nontrivial limits $a \to 0$, $b \to 0$ of the general

[14] In SBL, we are more interested in a proper posterior *approximation*, since the true posterior is intractable; see Section 2.2.2.

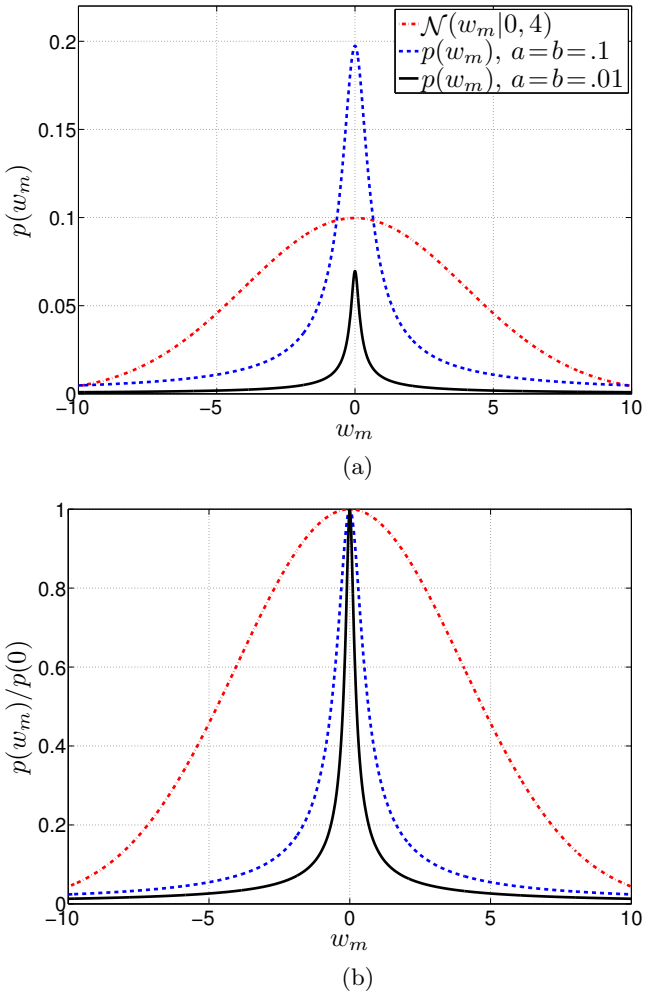


Figure 2.3: (a) Effective SBL weight prior $p(w_m)$ for different settings of a and b compared to a zero mean Gaussian with variance 4. If a and b get smaller, $p(w_m)$ is more peakier at zero. This can be better seen in (b), where all pdfs are scaled such that their respective largest value is 1.

solutions with unspecified a and b.

2.2.2 Approximation methods

Based on the model defined in Section 2.2.1, we are interested in making predictions for a target t^* given an arbitrary input $\boldsymbol{x}^*$ and the training data $\mathcal{D}$. That is, we need to evaluate the distribution[15]

$$p(t^*|\boldsymbol{t}) = \int p(t^*|\tau, \boldsymbol{w}, \boldsymbol{\alpha}) p(\tau, \boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{t})\, d\tau\, d\boldsymbol{w}\, d\boldsymbol{\alpha}. \tag{2.20}$$

Unfortunately, the posterior over all hidden variables,

$$p(\tau, \boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{t}) = \frac{p(\boldsymbol{t}|\tau, \boldsymbol{w}, \boldsymbol{\alpha}) p(\tau, \boldsymbol{w}, \boldsymbol{\alpha})}{p(\boldsymbol{t})}, \tag{2.21}$$

cannot be computed in closed form due to the intractable model evidence $p(\boldsymbol{t})$. Thus, also (2.20) cannot be computed in closed form and we need some effective approximation.

In the following we discuss four common approximation methods proposed in the literature, where all of these methods result in the same sequential update expressions

$$\boxed{\begin{aligned} \boldsymbol{\Sigma}^{[i]} &= \left(\tau^{[i-1]} \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \operatorname{diag}(\boldsymbol{\alpha}^{[i-1]})\right)^{-1}, \\ \boldsymbol{\mu}^{[i]} &= \tau^{[i-1]} \boldsymbol{\Sigma}^{[i]} \boldsymbol{\Phi}^T \boldsymbol{t}, \\ \alpha_m^{[i]} &= \frac{1}{(\mu_m^{[i]})^2 + \Sigma_{mm}^{[i]}}, \quad \forall m, \\ \tau^{[i]} &= \frac{N}{\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}^{[i]}\|^2 + \operatorname{tr}\left(\boldsymbol{\Sigma}^{[i]} \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)}, \end{aligned}} \tag{2.22}$$

although their derivations are different. Later in Section 2.2.3, we explain why this is the case. Note that in (2.22), the iteration index i is defined as $i = 1, 2, \ldots$ and some initialization $\alpha_m^{[0]}$, $\forall m$, and $\tau^{[0]}$ is required[16].

The first two presented methods are based on the fact, that the posterior over all hidden variables given the data can be decomposed as

$$p(\tau, \boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{t}) = p(\boldsymbol{w}|\boldsymbol{t}, \tau, \boldsymbol{\alpha}) p(\tau, \boldsymbol{\alpha}|\boldsymbol{t}), \tag{2.23}$$

[15] Like before, we leave out the known deterministic parts for a compact notation, i.e., also $\boldsymbol{x}^*$.

[16] Note that alternatively we may also change the order of the update indices in (2.22) and start with some initialization $\boldsymbol{\mu}^{[0]}$ and $\boldsymbol{\Sigma}^{[0]}$.

where the first term is the weight posterior given the noise precision and the hyperparameters. We just denote it as the weight posterior. It is a multivariate Gaussian and can be computed as

$$p(\boldsymbol{w}|\boldsymbol{t},\tau,\boldsymbol{\alpha}) = \frac{p(\boldsymbol{t}|\tau,\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})}{p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})} = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}), \tag{2.24}$$

with

$$\boldsymbol{\Sigma} = \left(\tau\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \operatorname{diag}(\boldsymbol{\alpha})\right)^{-1} \tag{2.25}$$

$$\boldsymbol{\mu} = \tau\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t}. \tag{2.26}$$

Equations (2.25) and (2.26) can be simply obtained by comparing the quadratic exponent of the multivariate Gaussian distributions in (2.24)

$$\begin{aligned} p(\boldsymbol{w}|\boldsymbol{t},\tau,\boldsymbol{\alpha}) \;&\propto\; \exp\Big\{ -\frac{\tau}{2}(\boldsymbol{\Phi}\boldsymbol{w}-\boldsymbol{t})^T(\boldsymbol{\Phi}\boldsymbol{w}-\boldsymbol{t}) - \frac{1}{2}\boldsymbol{w}^T\operatorname{diag}(\boldsymbol{\alpha})\boldsymbol{w}\Big\} \\ &\propto\; \exp\Big\{ -\frac{1}{2}\underbrace{\boldsymbol{w}^T\Big(\tau\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \operatorname{diag}(\boldsymbol{\alpha})\Big)\boldsymbol{w}}_{\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{w}} + \underbrace{\tau\boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{t}}_{\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}\Big\}, \end{aligned}$$

where $\mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu},\boldsymbol{\Sigma}) \propto \exp\{\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{\mu})\}$.

The second term in (2.23) is the posterior of the noise precision τ and the hyperparameters $\boldsymbol{\alpha}$ given the targets $\boldsymbol{t}$. The assumed forms of the noise precision prior $p(\tau)$ and the hyperpriors $p(\boldsymbol{\alpha})$ affect the form of $p(\tau,\boldsymbol{\alpha}|\boldsymbol{t})$, which essentially distinguishes the first two presented methods as we will see in the next paragraphs. Both methods are based on an alternating update procedure between the weight posterior (2.24) and the MAP point mass approximation of $p(\tau,\boldsymbol{\alpha}|\boldsymbol{t})$, i.e.,

$$\begin{aligned} \{\tau_{\text{MAP}},\boldsymbol{\alpha}_{\text{MAP}}\} &= \underset{\tau,\boldsymbol{\alpha}}{\operatorname{argmax}}\; p(\tau,\boldsymbol{\alpha}|\boldsymbol{t}) \\ &= \underset{\tau,\boldsymbol{\alpha}}{\operatorname{argmax}}\; p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})p(\tau)p(\boldsymbol{\alpha}). \end{aligned} \tag{2.27}$$

Maximizing the marginal likelihood (MML)

Let us start with the method first proposed in [23], the *maximum marginal likelihood* (MML) approach, which is also called *type-II maximum likelihood* method or *evidence procedure*. The MML method assumes a flat prior for the noise precision and the hyperparameters, i.e., it essentially maximizes $p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})$, which is denoted as the marginal likelihood. In the sense of the general model definition using gamma

priors in Section 2.2.1, this is equivalent to setting $a = c = 1$ and $b = d = 0$, which results in $p(\tau) \propto 1$ and $p(\alpha_m) \propto 1$ for all m. Again, these are improper priors, but not Jeffreys priors [87] as discussed above. Using such priors makes MAP estimation equivalent to ML estimation. Note that for flat priors, the SBL prediction now depends on the scaling of the targets $\boldsymbol{t}$ and the basis functions $\psi_m(\cdot)$. This is not true for scale-invariant Jeffreys priors as stated in [1] and discussed in Section 2.2.1. We further discuss this issue in Section 2.2.3.

The marginal likelihood, can be determined as[17]

$$\begin{aligned} p(\boldsymbol{t}|\tau, \boldsymbol{\alpha}) &= \int p(\boldsymbol{t}|\boldsymbol{w}, \tau) p(\boldsymbol{w}|\boldsymbol{\alpha}) d\boldsymbol{w} \\ &= \int \mathcal{N}(\boldsymbol{t}|\boldsymbol{\Phi}\boldsymbol{w},\ \tau^{-1}\boldsymbol{I})\, \mathcal{N}(\boldsymbol{w}|\boldsymbol{0},\ \mathrm{diag}(\boldsymbol{\alpha})^{-1}) d\boldsymbol{w} \\ &= \mathcal{N}(\boldsymbol{t}|\boldsymbol{0},\ \boldsymbol{C}), \end{aligned} \tag{2.28}$$

where $\boldsymbol{C} = \tau^{-1}\boldsymbol{I} + \boldsymbol{\Phi}\big(\mathrm{diag}(\boldsymbol{\alpha})\big)^{-1}\boldsymbol{\Phi}^T$, which can be simply obtained from standard results for Gaussian marginalization [4]. Its maximum, as mentioned in [1], cannot be obtained in closed form, although update expressions that depend on the previous state of the weight posterior can be derived as

$$\alpha_m = \frac{1}{\mu_m^2 + \Sigma_{mm}}\ , \quad \forall m, \tag{2.29}$$

and

$$\tau = \frac{N}{\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + \mathrm{tr}\left(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)}, \tag{2.30}$$

where μ_m is the mth element of $\boldsymbol{\mu}$ and Σ_{mm} the mth main diagonal element of $\boldsymbol{\Sigma}$ defined in (2.25) and (2.26), respectively. The detailed derivations of (2.29) and (2.30) are given in Appendix A.2. Note that the updates (2.29) and (2.30) are not given in an explicit form since α_m and τ depend on their previous states through $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ from (2.25) and (2.26), respectively. Thus, an iterative update procedure can be defined and if we start with an initialization for $\alpha_m^{[0]}$, $\forall m$, and $\tau^{[0]}$, the method (2.22) directly follows from the MML approach as we wanted to show.

By rearranging the terms in (2.29), one can also define implicit hyperparameter updates in (2.22) that also depend on the previous $\alpha_m^{[i-1]}$, namely

$$\alpha_m^{[i]} = \frac{1 - \alpha_m^{[i-1]}\Sigma_{mm}^{[i]}}{(\mu_m^{[i]})^2}\ , \quad \forall m. \tag{2.31}$$

[17] Note that it has the form of a *Gaussian process* [56, 4].

This updates, as shown in [23, 1], are based on a related idea shown in [93] and typically converge much faster; also see [29] for more comments. Because this rearranged updates have practical relevance for faster implementations but not for our ongoing discussion, we restrict our further analysis on (2.29), although we use the implicit updates for performance comparisons later in this thesis. Furthermore, this heuristic implicit updates have no convergence guarantees as opposed to (2.29), which will be shown in the context of the EM and VB approach later. To distinguish the method with the update (2.29) from the method with the implicit update (2.31), we denote them as MML and MML*, respectively, for later reference.

Updating $\alpha_1, \ldots, \alpha_M$ and τ alternately with $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ as in (2.22) typically leads to sparse solutions in the sense that many α_m diverge towards infinity, which renders the corresponding weight posterior over w_m highly peaked at zero. In the limit case, where $w_m = 0$, the related basis function $\psi_m(\cdot)$ becomes irrelevant as can be seen from (2.14), i.e., it can be pruned from the model.

With the MML approximation, the predictive distribution (2.20) becomes tractable for given estimates τ_{ML} and $\boldsymbol{\alpha}_{\mathrm{ML}}$, and has a Gaussian form

$$\int p(t^*|\tau_{\mathrm{ML}}, \boldsymbol{w}) p(\boldsymbol{w}|\boldsymbol{t}, \tau_{\mathrm{ML}}, \boldsymbol{\alpha}_{\mathrm{ML}})\, d\boldsymbol{w} = \mathcal{N}\left(t^*|y^*, (\tau^*)^{-1}\right), \tag{2.32}$$

where we have inserted the delta approximation and made use of the identity $p(t^*|\tau, \boldsymbol{w}, \boldsymbol{\alpha}) = p(t^*|\tau, \boldsymbol{w})$, since the targets are independent of the hyperparameters given the weights. The latter relation is also directly visible from Figure 2.2, where $\boldsymbol{\alpha}$ is not in the *Markov blanket* [4] of $\boldsymbol{t}$, which also holds for any predicted target t^*. The prediction mean y^* and precision τ^* can be computed as

$$y^* = \boldsymbol{\mu}^T \boldsymbol{\phi}(\boldsymbol{x}^*) \tag{2.33}$$

$$\text{and} \qquad \tau^* = \left(\tau_{\mathrm{ML}}^{-1} + \boldsymbol{\phi}(\boldsymbol{x}^*)^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\boldsymbol{x}^*)\right)^{-1}, \tag{2.34}$$

with $\boldsymbol{\phi}(\boldsymbol{x}^*) = [\psi_1(\boldsymbol{x}^*), \ldots, \psi_M(\boldsymbol{x}^*)]^T$. We note that since (2.32) is Gaussian, the best prediction for the targets given the training data is the mean y^* which is also the mode.

Maximization of the marginal posterior over log-scale (MMP-log)

A maximization method of the marginal posterior over a logarithmic scale, which we denote as MMP-log, is also presented in [1]. It differs from the MML approach by assuming Jeffreys priors ($a = b = c = d = 0$) instead of flat priors and, furthermore,

by maximizing the marginal posterior $p(\tau, \boldsymbol{\alpha}|\boldsymbol{t})$ over a logarithmic scale instead of a linear scale. Note that, over a logarithmic scale, the priors are flat, as we already mentioned in Section 2.2.1, and so the influence of the prior vanishes again, namely

$$\begin{aligned}\{[\ln\tau]_{\mathrm{MAP}}, [\ln\boldsymbol{\alpha}]_{\mathrm{MAP}}\} &= \underset{\ln\tau,\,\ln\boldsymbol{\alpha}}{\operatorname{argmax}}\; p(\boldsymbol{t}|\ln\tau, \ln\boldsymbol{\alpha})p(\ln\tau)p(\ln\boldsymbol{\alpha}) \\ &= \underset{\ln\tau,\,\ln\boldsymbol{\alpha}}{\operatorname{argmax}}\; p(\boldsymbol{t}|\ln\tau, \ln\boldsymbol{\alpha}).\end{aligned} \tag{2.35}$$

After transforming the log-MAP estimators back to linear scale, it is easy to see that the resulting update expressions are equivalent to MML. This seems counterintuitive, since we have assumed different priors $p(\tau)$ and $p(\boldsymbol{\alpha})$ for MML and MMP-log. Indeed, there is a fundamental problem involved in this approach which will be discussed in Section 2.2.3.

Finally we note, that since the updates are equivalent, all results from the MML method can be equivalently applied to MMP-log.

Expectation maximization (EM-SBL)

If we treat the weights as the hidden variables, we can perform point estimation for τ and $\boldsymbol{\alpha}$ using the MAP-EM framework, which was presented to be a special case of variational Bayes in Section 2.1.3. As we show in the following, the resulting EM updates are only equivalent to the updates (2.22) if we consider flat priors $p(\tau)$ and $p(\alpha_m)$, $\forall m$.

In the E-step, we simply update the weight posterior $p(\boldsymbol{w}|\boldsymbol{t}, \tau^{\mathrm{old}}, \boldsymbol{\alpha}^{\mathrm{old}})$ given the previous estimates τ^{old} and $\boldsymbol{\alpha}^{\mathrm{old}}$ and consider the expectation

$$\mathbb{E}_{p(\boldsymbol{w}|\boldsymbol{t},\tau^{\mathrm{old}},\boldsymbol{\alpha}^{\mathrm{old}})}\{\ln p(\boldsymbol{t}, \tau, \boldsymbol{w}, \boldsymbol{\alpha})\} \tag{2.36}$$

to be maximized over $\boldsymbol{\alpha}$ and τ in the M-step. The maximization in the M-step leads to the updates

$$\alpha_m^{\mathrm{new}} = \begin{cases} \frac{2a-1}{\mu_m^2+\Sigma_{mm}+2b}, & a > 1/2 \\ 0 & 0 \le a \le \frac{1}{2} \end{cases}, \quad \forall m. \tag{2.37}$$

and

$$\tau^{\mathrm{new}} = \begin{cases} \frac{N+2c-2}{\|\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{\mu}\|^2+\mathrm{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi})+2d}, & c > 1-\frac{N}{2} \text{ and } c \ge 0 \\ 0, & \text{otherwise} \end{cases}, \tag{2.38}$$

where the detailed derivations can be found in Appendix A.3.

From (2.37) we see that the MAP-EM approach to SBL with $0 \le a \le \frac{1}{2}$ fails

for sparse estimation, since all α_m get stuck at $\alpha_m = 0$, independent of the weight posterior and thus also independent of the observations $\boldsymbol{t}$. It is easy to see, that (2.37) reduces to the known hyperparameter update in (2.22) only for $a = 1$ and $b = 0$, i.e., for a flat hyperprior. Obviously, for Jeffreys priors, as $a = b = 0$, EM-SBL fails for sparse estimation as discussed before.

From (2.38), we see that the update is only equivalent to our known update in (2.22) for $c = 1$ and $d = 0$, which corresponds to a flat prior $p(\tau)$. Note that since N is typically much larger than 2, the first solution $\tau^{\text{new}} > 0$ in (2.38) can be practically achieved for any $c \geq 0$.

We must note that our results (2.37) and (2.38) differ from the EM-SBL solutions presented in [1, Section A.3, Eq. (48) and (50)][18], which in our opinion are flawed.

Since it turns out that the hyper- and noise-precision-priors must be flat to obtain the same updates as in (2.22), the MAP-EM reduced to an ML-EM procedure for this case and thus is closely related to the MML approach.

After convergence, predictions for targets t^* can be equivalently made as before, using (2.33) and (2.34).

Variational Bayesian approximation (V-SBL)

Compared to all previous approaches, the VB approximation is the only method not using point estimates, i.e., the posterior is approximated by pdfs different from point mass distributions. This approach, denoted as V-SBL, was presented in [35], but together with the EM-SBL method did not get as much attention in the literature compared to the MML and MMP-log method. We discuss in Section 2.2.3 why this may be the case, although compared to the other presented methods, it is conceptually the most profound one.

Given the joint distribution from (2.13), in consistency with our notation in Section 2.1 we can define the set of hidden variables as $\mathcal{H} = \{\tau, \boldsymbol{w}, \boldsymbol{\alpha}\}$ and the observed variables as $\mathcal{O} = \{\boldsymbol{t}\}$. We further assume a *structured mean field approximation* over the approximate posterior as

$$q(\mathcal{H}) = q(\tau)q(\boldsymbol{w})q(\boldsymbol{\alpha}) = q(\tau)q(\boldsymbol{w}) \prod_{m=1}^{M} q(\alpha_m), \tag{2.39}$$

where $q(\boldsymbol{\alpha}) = \prod_{m=1}^{M} q(\alpha_m)$ automatically follows from the independence assumption

[18] Note that in [1, Equation (50)], the transformation $(\sigma^2)^{\text{old}} \sum_i \gamma_i = \text{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi})$, shown in [1, Section A.2.2], must be used for direct comparison. We further note that $\tau \triangleq \beta = \sigma^{-2}$ in [1].

(pdf factorization) between the $\boldsymbol{w}$ and $\boldsymbol{\alpha}$ posterior approximations. By applying the optimal unconstrained update given in (2.7) to each factor from (2.39), we obtain closed form solutions $q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$, $q(\alpha_m) = \text{Ga}(\alpha_m|\hat{a}_m, \hat{b}_m)$, and $q(\tau) = \text{Ga}(\tau|\hat{c}, \hat{d})$, with the parameters

$$\hat{\boldsymbol{\Sigma}} = \left(\hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \text{diag}(\hat{\boldsymbol{\alpha}})\right)^{-1}, \quad \hat{\boldsymbol{\mu}} = \hat{\tau}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{t} \tag{2.40}$$

$$\hat{a}_m = a + 1/2, \quad \hat{b}_m = b + (\hat{\mu}_m^2 + \hat{\Sigma}_{mm})/2, \tag{2.41}$$

$$\hat{c} = c + \frac{N}{2}, \text{ and } \quad \hat{d} = d + \frac{\|\boldsymbol{t} - \boldsymbol{\Phi}\hat{\boldsymbol{\mu}}\|^2 + \text{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\Phi})}{2}, \tag{2.42}$$

where $\hat{\tau} = \mathbb{E}_{q(\tau)}\{\tau\} = \hat{c}/\hat{d}$, $\hat{\alpha}_m = \mathbb{E}_{q(\alpha_m)}\{\alpha_m\} = \hat{a}_m/\hat{b}_m$, $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_1, \ldots, \hat{\alpha}_M]^T$, $\hat{\mu}_m$ is the mth element of the mean vector $\hat{\boldsymbol{\mu}}$, and $\hat{\Sigma}_{mm}$ is the mth element on the main diagonal of the covariance matrix $\hat{\boldsymbol{\Sigma}}$. For a detailed derivation we refer to [35].

The variational lower bound $\mathcal{L}[q(\mathcal{H})]$ (see Section 2.1.1) can be locally maximized, by updating the distributions $q(\boldsymbol{w})$, $q(\boldsymbol{\alpha})$ and $q(\tau)$ in any arbitrary order until convergence, namely by updating the equations (2.40)-(2.42) in any order. Note that we can directly work with $\hat{\alpha}_m$ instead of $\hat{a}_m$ and $\hat{b}_m$ because $\hat{a}_m$ is a constant and $\hat{\alpha}_m$ only depends on the other variables through $\hat{b}_m$. The same holds for $\hat{\tau}$ in combination with $\hat{c}$ and $\hat{d}$, where both $\hat{\alpha}_m$ and $\hat{\tau}$ are sufficient to fully specify the distributions $q(\alpha_m)$ and $q(\tau)$, respectively. When we rewrite (2.41) and (2.42) in terms of the expectations $\hat{\alpha}_m$ and $\hat{\tau}$ as specified above, we obtain

$$\hat{\alpha}_m = \frac{1 + 2a}{\hat{\mu}_m^2 + \hat{\Sigma}_{mm} + 2b}, \quad \forall m \tag{2.43}$$

and

$$\hat{\tau} = \frac{N + 2c}{\|\boldsymbol{t} - \boldsymbol{\Phi}\hat{\boldsymbol{\mu}}\|^2 + \text{tr}\left(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right) + 2d}, \tag{2.44}$$

where we would like to stress the differences in the numerators compared to the EM-SBL solutions in (2.37) and (2.38). It is easy to see, that the updates are only equivalent to the results presented in (2.22) if we set $a = b = c = d = 0$, i.e., if we consider Jeffreys priors. Another difference to the previous methods is that we now consider the expectations $\hat{\alpha}_m$ and $\hat{\tau}$ instead of point estimators at the modes.

Now, since we have a factorized posterior approximation $q(\mathcal{H})$, predictions based on (2.20) result in

$$\hat{p}(t^*|\boldsymbol{t}) = \int p(t^*|\tau, \boldsymbol{w}) q(\tau) q(\boldsymbol{w})\, d\tau\, d\boldsymbol{w}, \tag{2.45}$$

where we have integrated out $\boldsymbol{\alpha}$. Unfortunately, as mentioned in [35], the integration

Name of the method	Type of prior	Uses point-estimation
MML	Flat	Yes
MMP-log	Jeffreys	Yes
EM-SBL	Flat	Yes
V-SBL	Jeffreys	No

Table 2.1: Properties of the four approximations: *maximizing the marginal likelihood* (MML), *maximizing the marginal posterior over log-scale* (MMP-log), *expectation maximization SBL* (EM-SBL) and *variational SBL* (V-SBL).

over both τ and $\boldsymbol{w}$ is intractable. However, according to [35], it can be reasonably approximated by

$$\int p(t^*|\hat{\tau}, \boldsymbol{w}) q(\boldsymbol{w})\, d\boldsymbol{w} = \mathcal{N}(t^*|y^*, \tau^*), \tag{2.46}$$

with y^* and τ^* specified as in (2.33) and (2.34), where we use $\hat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{\mu}}$ and $\hat{\tau}$ instead of $\boldsymbol{\Sigma}, \boldsymbol{\mu}$ and τ_{ML} respectively; cf. [35]. Note that when the noise precision τ is known, i.e., we do not have a $q(\tau)$ in (2.39), then (2.46) directly follows from (2.45) for $\hat{\tau} = \tau$.

2.2.3 Comparison and analysis of the discussed methods

In this section we want to compare and analyze the four previously presented methods used for SBL approximate inference. They all lead to the same iterative updates (2.22) under certain assumptions. However, there are also some inconsistencies involved which we will discuss in detail.

Since SBL optimization is multimodal, we are not guaranteed to find a global optimum. However, empirically, in many situations these local optima typically lead to very sparse solutions with good predictive performance. A more detailed analysis of SBL local optima in the MML context can be found in [29].

Table 2.1 summarizes the main properties of the methods. *Type of prior* refers to the used parameterization for the noise precision and hyperparameter priors defined by (2.18) and (2.17) to obtain update expressions of the form (2.22) for each method. From this perspective, the only two methods that are consistent with the SBL model defined in Section 2.2.1 and [1], which uses scale-invariant Jeffreys priors, are MMP-log and V-SBL. We further see from Table 2.1 that V-SBL is the only method not using point mass approximations, i.e., it is the only approach with a Bayesian treatment that approximates the posterior with a pdf different to a point mass mode estimator.

Convergence of all four methods given in Table 2.1 is guaranteed. This automatically follows from the general convergence properties of EM and VB and the fact that all methods lead to equivalent updates under their respective settings. Note that for MML*, which is based on the implicit updates (2.31), convergence has not been proven to our knowledge.

Now let us analyze MMP-log, which is consistent with the SBL model definition using Jeffreys priors. We want to find out why it obtains equivalent results as MML, even though MML is not consistent with our scale-invariant definition of the SBL model, since it uses flat priors. The main difference of MMP-log related to MML, is that the maximization of the marginal noise precision and hyperparameter posterior is performed over a logarithmic scale instead of a linear scale. As pointed out in [94, 54], for a variable-transformation using an invertible function $g(\cdot)$, the MAP estimate $[\alpha]_{\text{MAP}}$ in general is not equal to $g^{-1}([g(\alpha)]_{\text{MAP}})$, where $g^{-1}(\cdot)$ is the inverse function of $g(\cdot)$. That is, transforming the random variables to log-scale does not solve the same problem and thus the MMP-log approach is misleading. The reason why MMP-log nevertheless leads to the same results as MML, is that the influence of the Jeffreys prior vanishes over log-scale, i.e., it becomes an ML approach instead, and the relation $[\alpha]_{\text{ML}} = g^{-1}([g(\alpha)]_{\text{ML}})$ always holds for ML. Note the fundamental difference between the transformation of a random variable vs. the transformation of a likelihood parameter.

In the following we investigate the difference between the EM and the VB approach. EM-SBL uses point estimates τ and $\boldsymbol{\alpha}$, whereas V-SBL can be expressed in terms of the expectations $\hat{\tau}$ and $\hat{\boldsymbol{\alpha}}$.[19] By noting that the mode of the gamma pdf $\text{Ga}(\alpha_m|\hat{a}_m, \hat{b}_m)$ is $(\hat{a}_m - 1)/\hat{b}_m$ for $\hat{a}_m > 1$ and zero for $0 \leq \hat{a}_m \leq 1$, whereas its expectation is $\hat{a}_m/\hat{b}_m$, it is clear that both estimates lead to the same result for flat and Jeffreys priors, respectively. Based on the discussion about EM and VB in Section 2.1.3, this can be seen from (2.41), where the mode estimate over $q(\alpha_m)$ using $a = 1$ and $b = 0$ in (2.37) equals the expectation for $a = b = 0$ in (2.43). Similarly, the same equivalence holds for the noise precision τ when we compute the point estimator (2.38) for $d = 1$ and $c = 0$ and the expectation estimator (2.44) for $c = d = 0$ from (2.42).

To summarize, the equivalence of EM-SBL and V-SBL for their respective priors seems to be a coincidence. It is only based on the mode and expectation relations of the gamma distribution. That is, the difference between the mode and the expectation estimator corrects the effect of the different prior assumptions, namely for the

[19] This is not true for VB in general, but holds for V-SBL.

flat and Jeffreys prior, respectively.

There are further methods for approximate SBL inference in the literature. Many are closely related to the presented ones. The approach [95] for instance uses Jeffreys hyperpriors with the EM algorithm, where the hyperparameters are considered as hidden variables while the weights and noise precision are estimated in the M-step. Note that this is different from EM-SBL, where the weights are considered as hidden variables. Nonetheless, it is also just a special case of VB, although the update expressions turn out to be slightly different for this modified usage of the EM. Another alternative method is presented, e.g., in [96]. In this work, the negative log-marginal likelihood function is minimized via an auxiliary upperbounding function. This procedure, however, still optimizes the marginal likelihood and produces point estimators rather then densities, i.e., it has the same conceptual weaknesses as discussed for MML, namely that it does not consider a scale-invariant model.

Finally, it should be mentioned that as we have shown in this section, all four discussed methods, which are based on different approximation techniques, lead to the same implementations under different model assumptions. However, as we have shown, the V-SBL approximation is the only method that is able to properly consider a scale-invariant SBL model, which is defined with the Jeffreys prior settings $a = b = c = d = 0$ as pointed out in [1].

Why variational sparse Bayesian learning?

Throughout this thesis we investigate the VB approach to SBL for many reasons. Although some reasons were already discussed before and given in Table 2.1, let us summarize the main ones as follows:

- VB is more general than EM, where the latter is included as a special case (see Section 2.1.3).
- VB approximates the posterior of hidden variables given the data by a density which is located around regions with large probability mass. That is, it uses a Bayesian approximation that carries information beyond the posterior mode.
- Convergence of VB is always guaranteed, since the lower bound is convex with respect to each approximation factor [4].
- VB allows to use different prior distributions including scale-invariant Jeffreys priors, which have desired properties for SBL as mentioned in [1] and discussed in Section 2.2.1.

- VB is a general deterministic approximate inference framework that can be applied to many different models. This also includes extensions to the SBL model as we will show in Chapter 4.

The EM-SBL and the VB-SBL methods did not get as much attention in the literature compared to MML and MMP-log. We think that this mostly has practical reasons. For MML/MMP-log [1], a faster method with implicit hyperparameter updates (i.e., MML* with update (2.31)) is defined, although, compared to (2.29), these updates have no proven convergence guarantees to our knowledge. Furthermore, in [68], a very fast but greedy algorithm was proposed to further speed up MML's convergence. It computes a maximizer of the marginal likelihood with respect to each single α_m in closed form, but still assumes flat priors and thus is not consistent with the SBL model definition based on scale-invariant priors (see Section 2.2.1). In Chapter 3, we develop an equivalent fast method for V-SBL based on fixed-point iterations, where our derivations are consistent with scale-invariant priors as discussed in Section 2.2.1.

2.2.4 How sparse Bayesian learning leads to sparsity

First, consider the fundamental sparse recovery problem [29]

$$\boldsymbol{w}_0^{\lambda} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \, \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{t}\|_2^2 + \lambda \|\boldsymbol{w}\|_0, \quad \lambda > 0, \tag{2.47}$$

where λ is an *accuracy of fit* vs. *sparsity* trade-off parameter, and $\|\cdot\|_0$ is the ℓ_0-quasinorm which counts the number of non-zeros of a vector argument. Note that in general it is not clear how to set λ in the presence of model mismatches and/or data noise to obtain a certain *sparsity* and/or accuracy of fit, although the optimization problem (2.47) can be interpreted as the augmented Lagrangian form of both problems

$$\boldsymbol{w}_0^{\tilde{M}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \, \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{t}\|_2^2, \quad \text{s.t.} \quad \|\boldsymbol{w}\|_0 \leq \tilde{M}, \quad \tilde{M} > 0 \tag{2.48}$$

and

$$\boldsymbol{w}_0^{\eta} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \, \|\boldsymbol{w}\|_0, \quad \text{s.t.} \quad \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{t}\|_2^2 \leq \eta, \quad \eta > 0, \tag{2.49}$$

which allow to specify these parameters, respectively. For the noiseless case without model mismatches, when perfect reconstruction is possible, i.e., $\boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{t}$, we obtain

the *exact sparse recovery problem* as $\lambda \to 0$ [97][20], thus there is no trade-off parameter in this case. Both, the *exact sparse recovery problem* and (2.47), although theoretically important, are combinatorial NP-hard problems, which makes them computationally impractical for many scenarios. Alternatively, relaxations to the ℓ_0-quasinorm have been defined to reduce this computational burden. The tightest convex approximation to the ℓ_0 quasinorm is the ℓ_1 norm, which results in widely used methods called LASSO [31] or equivalently Basis Pursuit[21] [25]. This convex approximation, although leading to a unique global optimum, is not as sparse compared to non-convex approximations closer to the ℓ_0 quasinorm like, e.g., $\|\boldsymbol{w}\|_p$ with $p < 1$ [97, 98]. Roughly speaking, the better we approximate the ℓ_0 quasinorm, the more sparse our solution may be, but this is at the expense of getting more and more local optima, which again increases the computational burden when we are seeking for the best solution. So, basically we are interested in a method that can be computed quite efficiently while at the same time has a small number of local optima. It turns out that SBL is a very useful tool to trade-off these issues by having few local optima which typically correspond to very sparse solutions. That means, in practical situations, we can only locally optimize SBL to obtain good solutions. Furthermore, as pointed out in [29], for the noiseless case, the SBL global optimum is always equal to the maximally sparse solution, which does not hold for LASSO. In the following we only consider the noise and/or model mismatch case (2.47) and relaxations thereof, i.e., we do not consider *exact sparse recovery problems*.

Before we show how MAP estimation in the SBL model relates to the structure of the problem (2.47), we note that all the above mentioned sparse recovery problems can be rewritten as MAP estimation problems using sparsity inducing prior distributions over $\boldsymbol{w}$. That is

$$\boldsymbol{w}_{\text{MAP}} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}), \tag{2.50}$$

with a Gaussian likelihood

$$p(\boldsymbol{t}|\boldsymbol{w}) \propto \exp\left\{-\frac{1}{2\lambda}\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{t}\|_2^2\right\} \tag{2.51}$$

and a factorial prior

$$p(\boldsymbol{w}) \propto \exp\left\{-\frac{1}{2}\sum_{m=1}^{M} g(w_m)\right\}. \tag{2.52}$$

[20] Note that, as mentioned in [97], the limit must be taken outside the minimization.
[21] Sometimes also called Basis Pursuit Denoising when noise and/or model mismatches are present.

A prior of the form (2.52) is termed a sparse prior if $g(\sqrt{\cdot})$ is a concave and non-decreasing function on the interval $[0, \infty)$ [92, 99, 97]. In this sense, the equivalent prior for (2.47), defined as

$$p_0(\boldsymbol{w}) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{w}\|_0\right\} = \exp\left\{-\frac{1}{2}\sum_{m=1}^{M}\mathcal{I}(w_m \neq 0)\right\}, \tag{2.53}$$

obviously is a sparse prior[22], where we have used the indicator function defined as

$$\mathcal{I}(w_m \neq 0) = \begin{cases} 0 & w_m = 0 \\ 1 & w_m \neq 0 \end{cases} . \tag{2.54}$$

Let us now return to the SBL model, where for the following discussion we consider the noise precision τ to be known. To obtain the effective weight prior density $p(\boldsymbol{w})$ for SBL, we need to integrate over the hyperparameters of the joint hierarchical prior $p(\boldsymbol{w}, \boldsymbol{\alpha}) = p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})$. The resulting density is given as $p(\boldsymbol{w}) = \prod_{m=1}^{M} p(w_m)$, a product of *Student's-t* distributions $p(w_m)$ as pointed out in [1], which we already defined in (2.19). For $a = b = 0$, this prior is of the form $p(w_m) \propto 1/|w_m|$, which is highly peaked at zero[23]. Exact inference based on the effective weight prior is intractable, as pointed out in Section 2.2.2, because we cannot compute the normalization $p(\boldsymbol{t})$. However, as the MAP estimator has no normalization requirements, we can find the equivalent optimization problem as

$$\begin{aligned} \boldsymbol{w}_{\text{MAP}} &= \underset{\boldsymbol{w}}{\text{argmax}} \ \ln\left\{p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w})\right\} \\ &= \underset{\boldsymbol{w}}{\text{argmin}} \ \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \frac{2}{\tau}\sum_{m=1}^{M}\ln|w_m|. \end{aligned} \tag{2.55}$$

When we define the relation $\lambda = 2/\tau$, we obtain $g(w_m) = \ln|w_m|$ from (2.51) and (2.52), which again corresponds to a sparsity inducing prior. Although sparsity inducing, the exact SBL MAP optimization is practically useless, because whenever any weight becomes zero, a global minimum of (2.55) is achieved independently of the data fitting term.[24] That means, also $\boldsymbol{w}_{\text{MAP}} = \boldsymbol{0}$ is a global optimum. From this perspective, the only reason why one can obtain useful results from SBL, is

[22] Note that this prior is improper.

[23] Note that this improper prior is sometimes denoted as a Jeffreys prior in the literature, which is not correct, since the Jeffreys prior for a linear function of the Gaussian mean is flat.

[24] Note that these problems also remain for flat hyperpriors with $a = 1$ and $b = 0$, where $p(w_m) \propto 1/|w|^3$.

because of the more suitable posterior approximations made for inference, i.e., such approximations that incorporate information beyond the mode.

In this sense we support the comment given in [54, Section 2.2.2] that one should rather speak of "Sparse Approximate Bayesian" instead of "Sparse Bayesian". However, throughout this work we retain using the common "SBL" terminology to refer to the results obtained from approximate inference.

Note that from a Bayesian perspective, we should not claim that exact inference in the SBL model is useless, it is only intractable and the MAP estimation given in (2.55) is what makes no sense. When we however consider posterior information beyond the mode, things get decidedly different.

Later in Chapter 6, Section 6.2, we discuss an idea about how the MAP approach could possibly lead to usefull results.

From the optimization (2.55), we can only get a notion about how SBL may lead to sparsity after approximations were made, especially when we optimize over the hyperparameters. However, in a recent work [97], an analysis of Type-I methods (optimization over $\boldsymbol{w}$) vs. Type-II methods (optimization over $\boldsymbol{\alpha}$) is presented. It is shown that one can transform both problems to the respective other space, which allows for investigating the effective equivalent SBL prior in weight space for the approximative Type-II methods presented in Section 2.2.2. It turns out, that the resulting effective weight prior is sparsity inducing, but in general non-factorial, i.e., not of the form (2.52). For further details on this issue we refer the reader to [100, 97].

2.3 Consensus propagation

Complementary to the discussions in the previous sections, we now present the main properties of consensus propagation (CP) [67], a method for distributed averaging in networks based on message passing[25]. Amongst other distributed averaging methods like [103, 104], CP can be described in terms of loopy belief propagation (LBP), which is an approximate probabilistic inference algorithm for loopy graphs. Since CP has a Bayesian interpretation, it offers nice opportunities to be combined with the SBL model as we will show in Chapter 4.

[25] Note that we only consider standard loopy belief propagation [101, 4] here in this section and not any variational type of message passing [102] which our previous analysis in Section 2.1 may suggest.

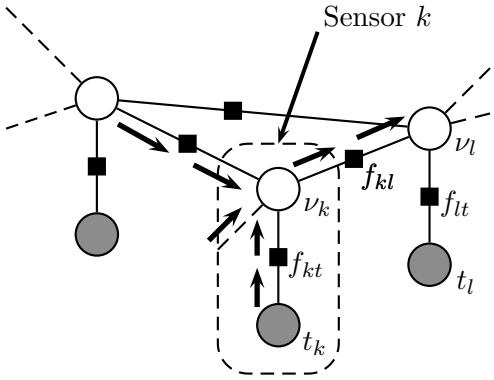


Figure 2.4: A factor graph representation of the *consensus propagation* (CP) model. Only local messages between neighboring nodes are necessary for inference using *loopy belief propagation* (LBP).

2.3.1 Model definition

We have already considered Bayesian networks as probabilistic directed acyclic graphs in Section 2.2.1. Now let us discuss CP in terms of factor graphs [105, 4], which are undirected bipartite graphs that consist of nodes, which represent random variables, and factors (black boxes) that represent functions of the variables they are connected to. Figure 2.4 shows the CP factor graph, which we assume to be connected, i.e., there is at least one path connecting each node in the graph. It can be interpreted as being overlayed on a sensor network, where each sensor k has some observation t_k that is connected to a hidden variable ν_k via a factor $f_{kt}(\nu_k, \tilde{t}_k) = \exp\left\{-\frac{1}{2}(\tilde{t}_k - \nu_k)^2\right\}$. The hidden variables of different sensors k and l are connected via the factor $f_{kl}(\nu_k, \nu_l) = \exp\left\{-\frac{\beta}{2}(\nu_k - \nu_l)^2\right\}$, which is called *coupling factor* and β is denoted as the *coupling parameter*. All factors represent unnormalized Gaussian distributions and thus, large β values in the factor f_{kl} increase the probability that hidden variables at neighboring sensors have similar values.[26] That is, for large β, a strong consensus is achieved.

Inference using LBP on the factor graph in Figure 2.4 leads to approximate Gaussian marginal posterior distributions $\tilde{p}(\nu_k|t_1, \ldots, t_K)$ with mean values close to the desired average $\frac{1}{K}\sum_{k=1}^{K} t_k$ at each sensor. The messages in LBP are only sent between neighboring nodes. Thus, the algorithm can be implemented distributedly on a sensor network.

[26]Note that factors need not to be normalized. Normalization can be also done at a later stage during message passing.

2.3.2 Gaussian loopy belief propagation

By applying the sum-product algorithm [4], which is a generalization of belief propagation [101], on the loopy factorgraph presented in Figure 2.4, an LBP algorithm can be derived. To obtain an approximation of the marginal probability $p(\nu_k|t_1,\dots,t_K)$ the following messages need to be computed and passed through the network until convergence:

$$m_{\tilde{t}_k\to f_{kt}} = \delta(\tilde{t}_k - t_k)\ , \quad k=1,\dots,K, \tag{2.56}$$

$$\begin{aligned} m_{f_{kt}\to\nu_k} &= \frac{1}{Z}\int m_{\tilde{t}_k\to f_{kt}} f_{kt}(\nu_k,\tilde{t}_k)\ d\tilde{t}_k = \frac{1}{Z} f_{kt}(\nu_k,t_k) \\ &= \mathcal{N}(\nu_k|t_k,1)\ , \quad k=1,\dots,K, \end{aligned} \tag{2.57}$$

$$m_{\nu_k\to f_{kl}} = \frac{1}{Z} m_{f_{kt}\to\nu_k} \prod_{u\in N(k)\backslash l} m_{f_{uk}\to\nu_k}, \tag{2.58}$$

$$\begin{aligned} m_{f_{kl}\to\nu_l} &= \frac{1}{Z}\int m_{\nu_k\to f_{kl}} f_{kl}(\nu_k,\nu_l)\ d\nu_k \\ &= \mathcal{N}(\nu_l|\hat{\nu}_{kl},\hat{\theta}_{kl}^{-1})\ , \quad \forall l\in N(k),\ k=1,\dots,K, \end{aligned} \tag{2.59}$$

where the variables Z are appropriate normalization constants, which are not necessary in general, but allow for a Gaussian interpretation[27], $N(k)$ denotes the set of neighbors of sensor k and $N(k)\backslash l$ the set of neighbors of k except the l-th sensor. The mean $\hat{\nu}_{kl}$ and precision $\hat{\theta}_{kl}$ in (2.59) can be computed as

$$\hat{\nu}_{kl} = \frac{t_k + \sum_{u\in N(k)\backslash l}\hat{\theta}_{uk}\hat{\nu}_{uk}}{1+\sum_{u\in N(k)\backslash l}\hat{\theta}_{uk}} \tag{2.60}$$

$$\text{and} \qquad \hat{\theta}_{kl} = \frac{1+\sum_{u\in N(k)\backslash l}\hat{\theta}_{uk}}{1+\beta^{-1}\left(1+\sum_{u\in N(k)\backslash l}\hat{\theta}_{uk}\right)}, \tag{2.61}$$

which are functions of the incoming mean an precision messages at sensor k. Thus, the message passing part of LBP can be fully described by (2.60) and (2.61). After some initialization, e.g., $\hat{\nu}_{kl}^{[0]} = \hat{\theta}_{kl}^{[0]} = 0$, the messages (2.60) and (2.61) have to be send through the network until convergence. To obtain the approximative marginal posterior at node ν_k after convergence, the normalized product of the incoming

[27] Using Z as a general symbol is a convenient notation for a normalization constant. Note that it may have different values, depending on where the normalization is applied.

messages needs to be computed by

$$\tilde{p}(\nu_k|t_1,\ldots,t_K) = \frac{1}{Z}\prod_{u\in N(k)} m_{f_{uk}\to\nu_k} = \mathcal{N}(\nu_k|\hat{\nu}_k,\hat{\theta}_k^{-1}) \tag{2.62}$$

with

$$\hat{\nu}_k = \frac{t_k + \sum_{u\in N(k)}\hat{\theta}_{uk}\hat{\nu}_{uk}}{1+\sum_{u\in N(k)}\hat{\theta}_{uk}} \tag{2.63}$$

$$\text{and} \qquad \hat{\theta}_k = 1 + \sum_{u\in N(k)}\hat{\theta}_{uk}, \tag{2.64}$$

where $Z = \int \prod_{u\in N(k)} m_{f_{uk}\to\nu_k} d\nu_k$. If the network is connected and $\beta \to \infty$, the mean (2.63) converges exactly to the average $\frac{1}{K}\sum_{k=1}^{K} t_k$ [67]. That is, each node in the network obtains the average value of all observations t_k in the network. A large value β leads to very slow convergence but good average estimates. On the other hand, a small β leads to fast convergence but induces errors in the estimates. Thus, for practical applications, a trade-off between accuracy and convergence time needs to be found. Convergence of CP is guaranteed for all $\beta > 0$ [67].

We must note that for LBP, convergence is not guaranteed in general. For Gaussian LBP, however, there exist sufficient conditions for convergence [106, 107].

Finally we note, that the graph topology and the order in which messages are sent also influences the convergence speed. For more discussions on this topic we refer to [67]. However, from a practical point of view, the easiest way to implement CP is to transfer the messages synchronously, although accurate synchronization in WSN is a great challenge [108]. For an alternative update scheme used for CP, see, e.g., [109], which is based on an aloha-like protocol.

Chapter 3

Fast Variational Sparse Bayesian Learning

In Section 2.2.3, we discussed different approximations used for SBL inference. We showed the advantages of VB inference over other existing methods, especially in terms of its usability with non-informative Jeffreys priors and its Bayesian density approximation rather than point estimation. Although conceptually superior, up to now there was no fast method like [68] defined for variational SBL, which may be one of the reasons for its lower popularity compared to MML. In this chapter we show how to derive such a fast method from fixed point analysis of the variational update expressions and further show its equivalence to the results obtained in [68]. We denote this method as *fast variational sparse Bayesian learning* (FV-SBL). Additionally, we also show that the resulting pruning criteria can be interpreted as comparing a basis function's signal to noise ratio (SNR) with a 0dB threshold. By changing this threshold to values larger than 0dB, one obtains an intuitive parameter for trading-off *quality of fit* vs. *sparsity*, which we analyze in different experiments.

3.1 Introduction

Consider the canonical model

$$\boldsymbol{t} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{\epsilon} \tag{3.1}$$

as defined in Section 2.2.1, where we now represent $\boldsymbol{\Phi} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M]$ by its M column vectors $\boldsymbol{\varphi}_m \in \mathbb{R}^N$, which we denote as basis vectors.

In the MML approach to SBL [1], as described in Section 2.2.2, the sparsity parameters $\boldsymbol{\alpha}$ are estimated by maximizing the marginal likelihood $p(\boldsymbol{t}|\tau, \boldsymbol{\alpha}) = \int p(\boldsymbol{t}|\tau, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w}$. Unfortunately, MML is known to converge rather slowly and the computational complexity of the algorithm scales as $O(M^3)$ [68, 1][1], which makes

This chapter is partly based on our publications [69] and [70].

[1] Also MML*, which uses implicit hyperparameter updates (2.31) that lead to faster overall con-

its application impractical for a large number of basis functions M. In [68] an alternative learning scheme was proposed to alleviate this drawback. Specifically, for a Gaussian prior $p(\boldsymbol{w}|\boldsymbol{\alpha}) \propto \exp(-\sum_m \alpha_m |w_m|^2/2)$ the maximum of the marginal likelihood function with respect to a single sparsity parameter α_m, assuming the sparsity parameters of the other basis functions are fixed, can be evaluated in closed form.[2]

An alternative approach to SBL is based on approximating the posterior $p(\boldsymbol{w}, \tau, \boldsymbol{\alpha}|\boldsymbol{t})$ with a pdf $q(\boldsymbol{w}, \tau, \boldsymbol{\alpha}) = q(\boldsymbol{w})q(\tau)q(\boldsymbol{\alpha})$ [35] so as to maximize the variational lower bound on $\ln p(\boldsymbol{t})$; see Section 2.2.2. There are several advantages of the variational approach to SBL as compared to others, as we discussed in detail in Section 2.2.3. Unfortunately, the variational approach to SBL [35] also suffers from the same computational complexity as MML and is prone to a slow convergence rate. Also, the pdfs $q(\boldsymbol{\alpha})$ and $q(\tau)$ are estimated so as to approximate the true posterior pdfs, thus obscuring the structure of the marginal likelihood that was exploited in [68] to accelerate the convergence rate of the learning scheme.

One possible strategy to improve the convergence rate of variational inference is to reduce coupling between the estimated random variables (see, e.g., [110] and [111]). In this chapter we propose an alternative approach that in some sense imitates fast marginal likelihood maximization (FMLM) [68].

Specifically, we consider the maximization of the variational lower bound with respect to a single factor $q(\alpha_m)$. As we will demonstrate, it then becomes possible to analytically compute the stationary points of the repeated updates of $q(\alpha_m)$ and $q(\boldsymbol{w})$ that maximize the bound and thus accelerate convergence. In [112] this was done both for a Gaussian prior $p(w_m|\alpha_m)$ and a Laplace prior $p(w_m|\alpha_m) \propto \exp(-\alpha_m|w_m|)$ when $q(\boldsymbol{w}) = \prod_m q(w_m)$, i.e., when the correlations between the elements of $\boldsymbol{w}$ are ignored. Here we present an extension of these results for the Gaussian prior case when $q(\boldsymbol{w})$ does not factorize, i.e., $q(\boldsymbol{w}) \neq \prod_m q(w_m)$. We derive the closed form expressions for the stationary points of the variational updates of $q(\alpha_m)$ and determine conditions that ensure convergence to these stationary points. We demonstrate that the convergence condition for each $q(\alpha_m)$ has a simple and intuitive interpretation in terms of the component signal to noise ratio (SNR), which provides further insight into the performance of SBL and eventually allows one to improve it. Moreover, we show that this convergence condition coincides with the

vergence, has a computational complexity of $O(M^3)$ over the iterations due to the unchanged weight updates (2.25) and (2.26).

[2] Fast suboptimal solutions to SBL with Laplace prior $p(\boldsymbol{w}|\boldsymbol{\alpha}) \propto \exp(-\sum_m \alpha_m |w_m|)$ have also been proposed [28].

condition in [68], which determines the maximum of the marginal likelihood function with respect to a single sparsity parameter.

3.2 Variational fixed-point analysis

Essentially, the variational update expressions (2.40)–(2.42), obtained in Section 2.2.2, provide the estimates of the parameters of the corresponding approximating pdfs. Due to the convexity of the variational lower bound in the approximating factors $q(\boldsymbol{w})$, $q(\tau)$, $q(\alpha_m)$, $m = 1, \ldots, M$, we can update these factors in any order [4]; furthermore, a group of factors can be updated successively while keeping the other factors fixed.[3]

3.2.1 General a and b parameters

We now start to study the expression for the mean $\hat{\alpha}_m$ of $q(\alpha_m)$ of some basis m for arbitrary $a \geq 0$ and $b \geq 0$.[4] Later in Section 3.2.2, we restrict ourselves to $a = b = 0$, since the expressions turn out to get quite involved and difficult to analyze. This simpler case, representing non-informative Jeffreys hyperpriors, is anyway the desired case for SBL, since it leads to scale invariance as discussed in Section 2.2.1. In Appendix B we formally show that if we compute the limits $a \to 0$, $b \to 0$ of the general fixed point solutions (discussed in the following), we obtain the same results as if we set $a = b = 0$ before computing the fixed points, which will be presented in Section 3.2.2. By computing these limits, we can first follow a proper justified Bayesian analysis and only in the end determine the effect of approaching the improper prior limit.

From (2.41) it follows that

$$\hat{\alpha}_m = \mathbb{E}_{q(\alpha_m)}\{\alpha_m\} = \frac{\hat{a}_m}{\hat{b}_m} = \frac{2a+1}{2b + \boldsymbol{e}_m^T(\hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^T + \hat{\boldsymbol{\Sigma}})\boldsymbol{e}_m}, \tag{3.2}$$

where $\boldsymbol{e}_m = [0, \ldots, 0, 1, 0, \ldots, 0]^T$ is a vector of all zeros but a 1 at the mth position. Let us now assume that $q(\boldsymbol{w})$ and $q(\alpha_m)$ are successively updated while keeping

[3] Note, however, that the order in which the factors are updated is important since different update orderings might lead to different local optima of the variational lower bound. We will return to this issue later in the text.

[4] Notice that since a and b are constants, $\hat{a}_m$ from (2.41) is also a constant and only $\hat{b}_m$ depends on the parameters of the other factors. With $\hat{\alpha}_m = \hat{a}_m/\hat{b}_m$, there is a direct connection between $\hat{b}_m$ and $\hat{\alpha}_m$. Thus, it makes sense to study the stationary point of the variational update expression in terms of $\hat{\alpha}_m$, rather than in terms of $\hat{a}_m$ and $\hat{b}_m$. This is also more intuitive, because (2.40) directly depends on $\hat{\alpha}_m$.

$q(\tau)$ and $q(\alpha_l)$ for $l \neq m$, fixed. This will generate a sequence of estimates $\{\hat{\alpha}_m^{[i]} = \hat{a}_m^{[i]}/\hat{b}_m^{[i]}\}_{i=1}^{I}$, with each element in the sequence computed according to (3.2). Our goal is to compute the stationary point $\hat{\alpha}_m^{[\infty]}$ of this sequence as $I \to \infty$.

With $\hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^T = \hat{\tau}^2\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{t}\boldsymbol{t}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}}^T$ we consider the influence of a single sparsity parameter $\hat{\alpha}_m$ on the right hand side of (3.2). By noting that $\mathrm{diag}(\hat{\boldsymbol{\alpha}}) = \sum_m \hat{\alpha}_m \boldsymbol{e}_m \boldsymbol{e}_m^T$, we rewrite $\hat{\boldsymbol{\Sigma}}$ as

$$\begin{aligned}\hat{\boldsymbol{\Sigma}} &= \left(\hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \hat{\alpha}_m\boldsymbol{e}_m\boldsymbol{e}_m^T + \sum_{l\neq m}\hat{\alpha}_l\boldsymbol{e}_l\boldsymbol{e}_l^T\right)^{-1} \\ &= \bar{\boldsymbol{\Sigma}}_m - \frac{\bar{\boldsymbol{\Sigma}}_m\boldsymbol{e}_m\boldsymbol{e}_m^T\bar{\boldsymbol{\Sigma}}_m}{\hat{\alpha}_m^{-1} + \boldsymbol{e}_m^T\bar{\boldsymbol{\Sigma}}_m\boldsymbol{e}_m},\end{aligned} \tag{3.3}$$

where the latter expression was obtained using the matrix inversion lemma [113] and defining

$$\bar{\boldsymbol{\Sigma}}_m = \left(\hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sum_{l\neq m}\hat{\alpha}_l\boldsymbol{e}_l\boldsymbol{e}_l^T\right)^{-1}. \tag{3.4}$$

Finally we define

$$\varsigma_m = \boldsymbol{e}_m^T\bar{\boldsymbol{\Sigma}}_m\boldsymbol{e}_m \quad \text{and} \quad \rho_m = \hat{\tau}\boldsymbol{e}_m^T\bar{\boldsymbol{\Sigma}}_m\boldsymbol{\Phi}^T\boldsymbol{t}. \tag{3.5}$$

Now, by substituting (3.3) into (3.2) and using the definitions (3.5) we obtain

$$\hat{\alpha}_m = \frac{(1+2a)(1+\hat{\alpha}_m\varsigma_m)^2}{\rho_m^2 + \varsigma_m + \hat{\alpha}_m\varsigma_m^2 + 2b(1+\hat{\alpha}_m\varsigma_m)^2}. \tag{3.6}$$

Expression (3.6) is a modified version of (3.2) that is now an implicit equation for $\hat{\alpha}_m$. Solving (3.6) explicitly for $\hat{\alpha}_m$ naturally leads to the desired stationary points $\hat{\alpha}_m^{[\infty]}$. As the equations for the stationary points get quite involved, we present these in Appendix B, Equations (B.1)-(B.3). Further stability analysis of these general terms is very complex and may not give much insight. Thus, we restrict ourselves on the simpler case of Jeffreys priors, i.e., $a = b = 0$, in the following.

3.2.2 Restriction to $a = b = 0$

In this section we compute the stationary points $\hat{\alpha}_m^{[\infty]}$ for improper Jeffreys priors with $a = b = 0$. In Appendix B we show that the same results can also be obtained by computing the limits $a \to 0$ and $b \to 0$ of the general solutions discussed in Section 3.2.1 and solved in Appendix B.

By setting $a = b = 0$, Equation (3.6) simply becomes

$$\hat{\alpha}_m = \frac{(1 + \hat{\alpha}_m \varsigma_m)^2}{\rho_m^2 + \varsigma_m + \hat{\alpha}_m \varsigma_m^2}. \tag{3.7}$$

Observe that both, (3.6) and (3.7) can be seen as nonlinear maps $\hat{\alpha}_m^{[i+1]} = F(\hat{\alpha}_m^{[i]})$ that at iteration i map $\hat{\alpha}_m^{[i]}$ to $\hat{\alpha}_m^{[i+1]}$. Naturally, the stationary points of these maps are equivalent to the desired (possibly multiple) stationary points $\hat{\alpha}_m^{[\infty]}$.

The following theorem provides analytical expressions for the stationary points of the map $\hat{\alpha}_m^{[i+1]} = F(\hat{\alpha}_m^{[i]})$ given in (3.7).

Theorem 1. *Assuming $a = b = 0$ and the initial condition $0 \leq \alpha_m^{[0]} < \infty$, the iterations of the nonlinear map*

$$\hat{\alpha}_m^{[i+1]} = F(\hat{\alpha}_m^{[i]}) = \frac{(1 + \hat{\alpha}_m^{[i]} \varsigma_m)^2}{\rho_m^2 + \varsigma_m + \hat{\alpha}_m^{[i]} \varsigma_m^2}, \tag{3.8}$$

where ρ_m and ς_m are defined in (3.5)*, converge as $i \to \infty$ to*

$$\hat{\alpha}_m^{[\infty]} = \begin{cases} (\rho_m^2 - \varsigma_m)^{-1}, & \rho_m^2 > \varsigma_m \\ \infty, & \rho_m^2 \leq \varsigma_m. \end{cases} \tag{3.9}$$

Proof. The proof is separated into the following three parts:

- We first compute the stationary points of the map (3.8).
- Then, we analyze the linear stability of the resulting finite stationary point.
- Finally, we show that for all initializations $0 \leq \alpha_m^{[0]} < \infty$ a unique stationary point is reached, depending on ρ_m and ς_m.

We begin by computing the stationary points of the map $\hat{\alpha}_m^{[i+1]} = F(\hat{\alpha}_m^{[i]})$. By inspecting (3.7) we observe that $\hat{\alpha}_m^{[\infty]} = \infty$ is a stationary point. The other solution is found by solving $\hat{\alpha}_m^* = F(\hat{\alpha}_m^*)$ with respect to $\hat{\alpha}_m^*$. After straightforward algebraic manipulations we obtain the second stationary point at

$$\hat{\alpha}_m^{[\infty]} = \hat{\alpha}_m^* = (\rho_m^2 - \varsigma_m)^{-1}. \tag{3.10}$$

We now investigate the linear stability of (3.10) by analyzing the map (3.8) in the vicinity of $\hat{\alpha}_m^* = (\rho_m^2 - \varsigma_m)^{-1}$. It is known that a stationary point of a map is linearly stable if the eigenvalues of the Jacobian of the map evaluated at this stationary point

are all within the unit circle (see, e.g., [114, Section 10.1] for one-dimensional maps). Thus, we compute

$$\left.\frac{\mathrm{d}F(\hat{\alpha}_m)}{\mathrm{d}\hat{\alpha}_m}\right|_{\hat{\alpha}_m=\hat{\alpha}_m^*} = -\frac{\varsigma_m(\varsigma_m - 2\rho_m^2)}{\rho_m^4}. \tag{3.11}$$

Now it can be shown[5] that $\left|\frac{\varsigma_m(\varsigma_m-2\rho_m^2)}{\rho_m^4}\right| < 1$ when

$$\rho_m^2 > \varsigma_m, \tag{3.12}$$

or

$$\rho_m^2 < \varsigma_m < (1+\sqrt{2})\rho_m^2. \tag{3.13}$$

Observe that the condition (3.13) suggests that the stationary point (3.10) might become negative. However, a negative value of $\alpha_m^{[i]}$ cannot be reached for $\alpha_m^{[0]} \geq 0$ since the map (3.8) can be shown to be positive for ς_m and ρ_m^2 satisfying (3.13).[6] Thus, $\hat{\alpha}_m^{[\infty]} = \hat{\alpha}_m^*$ is a finite stable positive stationary point when $\rho_m^2 > \varsigma_m$.

To see that this stationary point is reached for any $\alpha_m^{[0]}$ satisfying $0 \leq \alpha_m^{[0]} < \infty$ with $\rho_m^2 > \varsigma_m$, we further need to analyze the original map $F(\hat{\alpha}_m^{[i]})$ given in (3.8), which is a rational function of $\hat{\alpha}_m^{[i]}$. It is important to notice that the map is continuous for $\hat{\alpha}_m^{[i]} \geq 0$, since the only discontinuous point, corresponding to the root of the denominator of $F(\hat{\alpha}_m^{[i]})$, is at $-(\rho_m^2+\varsigma_m)/\varsigma_m^2$, which is always negative.

Now consider the cobweb diagrams [114] shown in Figure 3.1(a) and 3.1(b), where we illustrate iterations starting at $\hat{\alpha}_m^{[0]} > \hat{\alpha}_m^*$ and $0 \leq \hat{\alpha}_m^{[0]} < \hat{\alpha}_m^*$, respectively. We see, that at each iteration, $\hat{\alpha}_m^{[i]}$ gets closer to $\hat{\alpha}_m^*$ from the respective side of initialization if $\rho_m^2 > \varsigma_m$. To show that this is true for all $0 \leq \alpha_m^{[0]} < \infty$, it is sufficient to show that the following conditions hold:

$$F(\hat{\alpha}_m) \geq \hat{\alpha}_m^* \quad \text{for} \quad \hat{\alpha}_m > \hat{\alpha}_m^*, \tag{3.14}$$

$$F(\hat{\alpha}_m) < \hat{\alpha}_m \quad \text{for} \quad \hat{\alpha}_m > \hat{\alpha}_m^*, \tag{3.15}$$

$$F(\hat{\alpha}_m) \leq \hat{\alpha}_m^* \quad \text{for} \quad 0 \leq \hat{\alpha}_m < \hat{\alpha}_m^*, \tag{3.16}$$

$$F(\hat{\alpha}_m) > \hat{\alpha}_m \quad \text{for} \quad 0 \leq \hat{\alpha}_m < \hat{\alpha}_m^*. \tag{3.17}$$

The area of possible functions $F(\hat{\alpha}_m)$ that satisfy these conditions are illustrated in Figure 3.2.

[5] Assuming $\varsigma_m > 0$ according to definition (3.5), where all $\hat{\alpha}_l \geq 0$ for $l \neq m$.

[6] Since for $\hat{\alpha}_m^{[i]} \geq 0$, the map (3.8) is positive for any ρ_m^2 and $\varsigma_m \geq 0$, this also holds for (3.12).

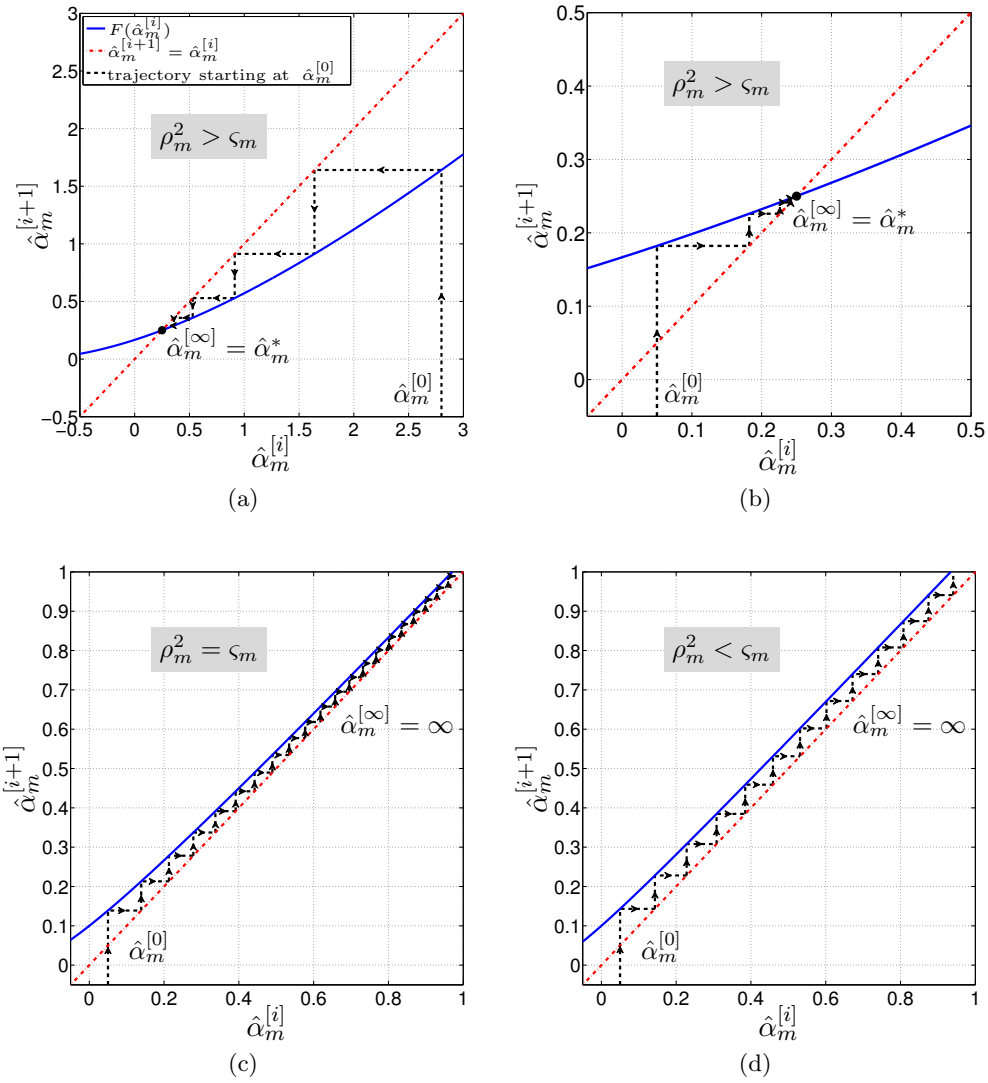


Figure 3.1: Cobweb diagram to illustrate the iterations to the stationary point $\hat{\alpha}_m^{[\infty]}$ dependent on the relations between ρ_m^2 and ς_m for different initializations $\hat{\alpha}_m^{[0]}$. The situations (a) and (b) illustrate a stable finite stationary point, whereas in (c) and (d) the iterations diverge towards infinity.

By noting that for $\hat{\alpha}_m \geq 0$, $F(\hat{\alpha}_m)$ is continuous and

$$\frac{\mathrm{d}F(\hat{\alpha}_m)}{\mathrm{d}\hat{\alpha}_m} = \frac{\varsigma_m(\hat{\alpha}_m\varsigma_m + 1)\left(\hat{\alpha}_m\varsigma_m^2 + 2\rho_m^2 + \varsigma_m\right)}{\left(\hat{\alpha}_m\varsigma_m^2 + \rho_m^2 + \varsigma_m\right)^2} > 0, \tag{3.18}$$

i.e., we have a positive slope for all $\hat{\alpha}_m \geq 0$, it is easy to see that conditions (3.14)

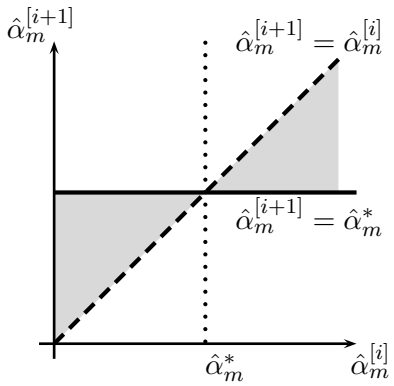


Figure 3.2: For $\rho_m^2 > \varsigma_m$ and any initial $\alpha_m^{[0]}$ satisfying $0 \leq \alpha_m^{[0]} < \infty$, the iterations of the map (3.8) converge steadily from the side of the initialization to the stationary point $\hat{\alpha}_m^*$, if the function $F(\hat{\alpha}_m)$ lies within the shaded area defined by (3.14)-(3.17).

and (3.16) are always satisfied, since by definition $F(\hat{\alpha}_m) = \hat{\alpha}_m$ only if $\hat{\alpha}_m = \hat{\alpha}_m^*$ or $\hat{\alpha}_m = \infty$.

Furthermore, the continuity of $F(\hat{\alpha}_m)$ for $\hat{\alpha}_m \geq 0$ together with the fact that we only have a single finite stationary point, namely $\hat{\alpha}_m^*$, lets us conclude that (3.15) and (3.17) are also satisfied. This is because, otherwise, there should exist another finite stationary point $F(\hat{\alpha}_m^{**}) = \hat{\alpha}_m^{**}$, which, however, is not the case.

Finally, both remaining situations $\rho_m^2 = \varsigma_m$ and $\rho_m^2 < \varsigma_m$, as depicted in Figure 3.1(c) and 3.1(d), respectively, lead to $\hat{\alpha}_m^{[\infty]} = \infty$, i.e., the iterations simply diverge. This is clear since for $\rho_m^2 = \varsigma_m$ we have a unique stationary point at $\hat{\alpha}_m^* = \infty$ and for $\rho_m^2 < \varsigma_m$ it is straightforward to show that for any $\hat{\alpha}_m \geq 0$, $F(\hat{\alpha}_m) > \hat{\alpha}_m$, i.e., the value for $\hat{\alpha}_m^{[i]}$ increases at every iteration i.

This ends the proof. □

Corollary 1. *Assume that $a = b = 0$ and (3.12) is satisfied for some basis vector $\boldsymbol{\varphi}_m$. Then the following is true: (i) the repeated updates of $q(\boldsymbol{w})$ and $q(\alpha_m)$ from (2.40) and (2.41), respectively, maximize the variational lower bound[7] and (ii) $q(\alpha_m)$ converges to $q(\alpha_m) = \mathrm{Ga}(\alpha_m|1/2, (\rho_m^2 - \varsigma_m)/2)$.*

Expression (3.10) together with the pruning condition (3.12) allow one to assess the impact of the mth basis vector $\boldsymbol{\varphi}_m$ in the matrix $\boldsymbol{\Phi}$ on the variational lower bound by computing (3.9): a finite value of $\hat{\alpha}_m^{[\infty]}$ instructs us to keep the mth component

[7] Note that the maximization is meant locally here.

since it should increase the bound, while an infinite value of $\hat{\alpha}_m^{[\infty]}$ indicates that the basis vector m is superfluous. In this way all M basis vectors can be processed sequentially in a round-robin fashion.

3.2.3 Equivalence between fast variational sparse Bayesian learning and fast marginal likelihood maximization

Theorem 2. *Consider a basis vector $\boldsymbol{\varphi}_m$ and assume that the pdfs $q(\alpha_l)$, $l \neq m$, and $q(\tau)$ are fixed. Let $\hat{\alpha}_m^{[\infty]}$ be the mean of $q(\alpha_m)$ obtained by repeatedly updating* ad infinitum *$q(\boldsymbol{w})$ and $q(\alpha_m)$ from* (2.40) *and* (2.41)*, respectively, where we assume $a = b = 0$. Then, $\hat{\alpha}_m^{[\infty]}$ given by* (3.9) *is a maximizer of the marginal likelihood function $p(\boldsymbol{t}|\alpha_m, \hat{\boldsymbol{\alpha}}_{\bar{m}}, \hat{\tau})$ with respect to the sparsity parameter α_m.*

Proof. It has been shown [68] that the maximum of $\ln p(\boldsymbol{t}|\alpha_m, \hat{\boldsymbol{\alpha}}_{\bar{m}}, \hat{\tau})$ with respect to α_m is obtained at

$$\hat{\alpha}_m = \begin{cases} s_m^2(q_m^2 - s_m)^{-1}, & q_m^2 > s_m, \\ \infty, & q_m^2 \leq s_m, \end{cases} \tag{3.19}$$

where $s_m = \boldsymbol{\varphi}_m^T \boldsymbol{C}_{\bar{m}}^{-1} \boldsymbol{\varphi}_m$, $q_m = \boldsymbol{\varphi}_m^T \boldsymbol{C}_{\bar{m}}^{-1} \boldsymbol{t}$, and $\boldsymbol{C}_{\bar{m}} = \hat{\tau}^{-1}\boldsymbol{I} + \sum_{l \neq m} \hat{\alpha}_l^{-1} \boldsymbol{\varphi}_l \boldsymbol{\varphi}_l^T$.

Using the matrix inversion lemma [113] applied to $\boldsymbol{C}_{\bar{m}}^{-1}$ it is easy to show that $\varsigma_m = s_m^{-1}$ and $\rho_m^2 = q_m^2/s_m^2$ (details are given in Appendix A.4). Thus, (i) the condition for keeping a basis function $q_m^2 > s_m$ in (3.19) is equivalent to the condition $\rho_m^2 > \varsigma_m$ in (3.12), and (ii) the value of $\hat{\alpha}_m$ in (3.19) and that of $\hat{\alpha}_m^{[\infty]}$ in (3.9) coincide. □

This means that, as we have already discussed in Section 2.2.2 for MML and V-SBL, also the corresponding fast methods FMLM and FV-SBL lead to the same pruning conditions under their respective model assumptions. More specifically, FMLM assumes flat hyperpriors $p(\alpha_m) \propto 1$, whereas FV-SBL assumes Jeffreys hyperpriors $p(\alpha_m) \propto 1/\alpha_m$. That is, only the derivation of FV-SBL is in accordance with a scale-invariant model as we discussed in Section 2.2.1. Note that the updates for the noise precision in FMLM and FV-SBL are equivalent to the updates in MML and V-SBL, respectively.

3.2.4 Analysis of the pruning condition

Since the pruning condition (3.12) is a key to model sparsity, let us study it in greater detail. From (3.5) it follows that ρ_m is the mean weight of the basis $\boldsymbol{\varphi}_m$ and

ς_m is the estimated variance of this weight obtained when $\hat{\alpha}_m = 0$ and $\hat{\alpha}_l$, $l \neq m$, are fixed. Notice that the ratio ρ_m^2/ς_m can be recognized as an estimate of the mth component signal-to-noise ratio (SNR) for $\hat{\alpha}_m = 0$, i.e., when the mth basis function is unregularized. Furthermore, according to (3.12) the basis vector $\boldsymbol{\varphi}_m$ is retained in the model provided $\text{SNR}_m = \rho_m^2/\varsigma_m > 1$, i.e., when the component's SNR is above 0dB; otherwise, the component is pruned.

This simple interpretation of the pruning condition can be used to generalize (3.12) to any desired SNR above 0dB. More specifically, given a certain desired $\text{SNR}'_m \geq 1$, the pruning condition (3.12) can be empirically adjusted as

$$\rho_m^2 > \varsigma_m \times \text{SNR}'_m, \tag{3.20}$$

which allows for the removal of the mth component when the SNR satisfies $\text{SNR}_m \leq \text{SNR}'_m$. Note, however, that the adjustment (3.20) might potentially decrease the variational lower bound since it will remove basis functions with finite sparsity parameters.

Despite the empirical nature of (3.20), it can be theoretically justified as a statistical composite hypothesis test. For more details see [115].

3.3 Implementation aspects and simulation results

3.3.1 Efficient computation of the test parameters ς_m and ρ_m

Consider the matrix $\bar{\boldsymbol{\Sigma}}_m$ in (3.4) in an alternative (permuted) form, where $\boldsymbol{\varphi}_m$ is shifted to the last column of $\boldsymbol{\Phi}$:

$$\begin{bmatrix} \hat{\tau}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\Phi}_{\bar{m}} + \text{diag}(\hat{\boldsymbol{\alpha}}_{\bar{m}}) & \hat{\tau}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\varphi}_m \\ \hat{\tau}\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}} & \hat{\tau}\boldsymbol{\varphi}_m^T\boldsymbol{\varphi}_m \end{bmatrix}^{-1}. \tag{3.21}$$

Here $\boldsymbol{\Phi}_{\bar{m}}$ is a matrix obtained from $\boldsymbol{\Phi}$ by removing the mth basis $\boldsymbol{\varphi}_m$ and $\hat{\boldsymbol{\alpha}}_{\bar{m}} = [\hat{\boldsymbol{\alpha}}]_{\bar{m}}$ is the vector $\hat{\boldsymbol{\alpha}}$ without the element $\hat{\alpha}_m$. Using a standard result for block matrix inversion [116, 113], the expression (3.21) can be re-written as

$$\begin{bmatrix} \left(\hat{\boldsymbol{\Sigma}}_{\bar{m}}^{-1} - \hat{\tau}\frac{\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\varphi}_m\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}}{\boldsymbol{\varphi}_m^T\boldsymbol{\varphi}_m}\right)^{-1} & -\hat{\tau}\hat{\boldsymbol{\Sigma}}_{\bar{m}}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\varphi}_m\gamma_m^{-1} \\ -\hat{\tau}\gamma_m^{-1}\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}\hat{\boldsymbol{\Sigma}}_{\bar{m}} & \gamma_m^{-1} \end{bmatrix}, \tag{3.22}$$

where $\gamma_m = \hat{\tau}\boldsymbol{\varphi}_m^T\boldsymbol{\varphi}_m - \hat{\tau}^2\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}\hat{\boldsymbol{\Sigma}}_{\bar{m}}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\varphi}_m$ and $\hat{\boldsymbol{\Sigma}}_{\bar{m}} = (\hat{\tau}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\Phi}_{\bar{m}} + \text{diag}(\hat{\boldsymbol{\alpha}}_{\bar{m}}))^{-1} = \left[\hat{\boldsymbol{\Sigma}} - \frac{\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_m\boldsymbol{e}_m^T\hat{\boldsymbol{\Sigma}}}{\boldsymbol{e}_m^T\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_m}\right]_{\bar{m}\bar{m}}$. Note that the notation $[\boldsymbol{B}]_{\bar{m}\bar{m}}$ denotes a matrix obtained after

removing the mth row and column of $\boldsymbol{B}$.

Using (3.22), ρ_m and ς_m in (3.5) can now be computed as

$$\begin{aligned} \varsigma_m &= (\hat{\tau}\boldsymbol{\varphi}_m^T\boldsymbol{\varphi}_m - \hat{\tau}^2\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}\hat{\boldsymbol{\Sigma}}_{\bar{m}}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\varphi}_m)^{-1} \\ \text{and} \quad \rho_m &= \hat{\tau}\varsigma_m\boldsymbol{\varphi}_m^T\boldsymbol{t} - \hat{\tau}^2\varsigma_m\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}\hat{\boldsymbol{\Sigma}}_{\bar{m}}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{t}. \end{aligned} \tag{3.23}$$

Notice that when the basis $\boldsymbol{\varphi}_m$ is retained in the model, the matrix $\hat{\boldsymbol{\Sigma}}$ can be efficiently updated with

$$\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}} - \frac{\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_m\boldsymbol{e}_m^T\hat{\boldsymbol{\Sigma}}}{(\hat{\alpha}_m^{[\infty]} - \hat{\alpha}_m^{\text{old}})^{-1} + \boldsymbol{e}_m^T\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_m}, \tag{3.24}$$

where $\hat{\alpha}_m^{[\infty]} = (\rho_m^2 - \varsigma_m)^{-1}$ and $\hat{\alpha}_m^{\text{old}}$ is a previous estimate of the mean of $q(\alpha_m)$.

3.3.2 Algorithm summary

We summarize the main steps of the proposed fast variational SBL scheme in Algorithm 3.1. It should be mentioned that updating $q(\tau)$, which leads to a new value $\hat{\tau}$, requires the covariance matrix $\hat{\boldsymbol{\Sigma}}$ to be recomputed, an $O(M^3)$ operation. The recomputation of $\hat{\boldsymbol{\Sigma}}$ after some $\hat{\alpha}_m$ has changed is a rank-1 update and can be implemented much more efficiently with complexity of $O(M^2)$ as given in (3.24). Notice that for both the FMLM algorithm [68] and the fast variational SBL algorithm the relevance of the vectors $\boldsymbol{\varphi}_m$ can theoretically be processed in any order. However, the exact order in which the basis vectors are processed matters since different update protocols can lead to different local optima. In the following simulations we first update components with large values of $\hat{\alpha}_m$, i.e., those basis functions that are least well aligned with the measurement $\boldsymbol{t}$. This should potentially reduce the dimensionality of the model already at early iterations.

For algorithm initialization we use the following simple procedure. First, the initial value for the mean $\hat{\tau}$ of the noise pdf $q(\tau)$ is chosen. Then, the parameters of $q(\boldsymbol{w})$ are initialized as $\hat{\boldsymbol{\Sigma}} = (\hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \tilde{\alpha}_0\boldsymbol{I})^{-1}$ and $\hat{\boldsymbol{\mu}} = \hat{\tau}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{t}$ for some small value of $\tilde{\alpha}_0$; finally, parameters of $q(\alpha_m)$, $m = 1, \ldots, M$, are found using (2.41). Such initialization provides an initial ranking of the components $\boldsymbol{\varphi}_m$ based on the values of $\hat{\boldsymbol{\alpha}}$, which is then used to define a sequence of basis function updates used by the FMLM and the fast variational SBL algorithms as mentioned above. Notice that FV-SBL can also start with an empty model and add basis functions sequentially, each time testing whether a new basis function should be added to the model or

Algorithm 3.1

Initialize $q(\boldsymbol{w})$, $q(\boldsymbol{\alpha})$, $q(\tau)$
while Not converged **do**
  for $m \in \{1, \ldots, M\}$ **do**
    Compute: ς_m and ρ_m from (3.23)
    if $\rho_m^2 > \varsigma_m$ **then**
      $\hat{\alpha}_m^{[\infty]} = 1/(\rho_m^2 - \varsigma_m)$, update $\hat{\boldsymbol{\Sigma}}$ from (3.24)
    else
      $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}}_{\bar{m}}$, $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}]_{\bar{m}}$, $M = M - 1$;
    end if
  end for
  Compute $\hat{\boldsymbol{\mu}}$ from (2.40), $q(\tau)$ from (2.42) and recompute $\hat{\boldsymbol{\Sigma}}$ from (2.40)
  Check for convergence
end while

not. Such a scheme is discussed later in Chapter 5 (see also [117]).

3.3.3 Simulation results

Comparison of different SBL approaches

In this section we compare the proposed fast variational SBL algorithm (FV-SBL) with the MML* algorithm [1] that uses the implicit update (2.31), the variational SBL algorithm (V-SBL) [35], and parameter-expanded variational Bayesian inference (PX-VB) [111]. Note that according to Theorem 2, FV-SBL and FMLM [68] are equivalent methods. However, we also analyze the results for FV-SBL with adjusted pruning as discussed in Section 3.2.4.

In the first experiment we consider a sparse vector estimation problem with random $\boldsymbol{\Phi}$ and a fixed number of nonzero weights[8]. In the second experiment we test the algorithms on the Concrete Compressive Strength (CCS) dataset [118] from the UCI Machine Learning Repository [119], which is a multivariate regression data set with 8 attributes and 1030 instances.

MML*, V-SBL and PX-VB methods require one to specify a pruning threshold for the sparsity parameters $\hat{\boldsymbol{\alpha}}$; specifically, when some $\hat{\alpha}_m$ exceeds the pruning threshold, the corresponding basis function is removed from the model. In all simulations we set this threshold to 10^{12}, as suggested in [1]. Obviously the estimated sparsity depends on a particular choice of this threshold. In contrast, for FV-SBL and

[8]The first experimental setup is not directly related to regression, but rather falls into another important area of SBL applications, namely sparse basis selection, see, e.g., [29]. Sparse basis selection on random matrices is also frequently used in compressed sensing [26].

FMLM methods the divergence of sparsity parameters is detected using closed form pruning conditions.

For all methods the same convergence criteria is used: the algorithm stops (i) when the number of basis functions between two consecutive iterations has stabilized and (ii) when the ℓ^2-norm of the difference between the values of hyperparameters at two consecutive iterations is less than 10^{-3}. Note that (i) is necessary to properly evaluate the norm in (ii). In the first experiment we also interrupt the algorithms when the number of iterations exceeds 10^4. Also, to simplify the analysis of the simulation results we assume the noise precision $\hat{\tau}$ to be known and fixed in all simulations, i.e., we also do not estimate $q(\tau)$ in FV-SBL.

We now begin with the discussion of the synthetic data experiment. To generate data for the sparse vector estimation we construct a random design matrix $\boldsymbol{\Phi}_{\text{gen}} \in \mathbb{R}^{N \times N}$ with $N = 100$ by drawing N^2 samples independently from a standard Gaussian distribution $\mathcal{N}(0, 1)$; the target vector $\boldsymbol{t}$ is then generated according to $\boldsymbol{t} = \boldsymbol{\Phi}_{\text{gen}} \boldsymbol{w}_{\text{gen}} + \boldsymbol{\epsilon}_{\text{gen}}$ with a weight vector $\boldsymbol{w}_{\text{gen}}$ having only 5 nonzero elements equal to 1 at random locations. White Gaussian noise is added via $\boldsymbol{\epsilon}_{\text{gen}}$ such that specific SNR levels are achieved. Our goal is to estimate $\boldsymbol{w}_{\text{gen}}$ out of the noisy observations $\boldsymbol{t}$ when the same basis vectors as in the data generation are used, i.e., we have $\boldsymbol{\Phi} = \boldsymbol{\Phi}_{\text{gen}}$. We also test the performance of FV-SBL with adjusted pruning condition (3.20) (FV-SBL, adj.) by simulating an oracle estimator that knows the true signal SNR; the true SNR is then used as the SNR'_m adjustment in (3.20). The corresponding simulation results are summarized in Figure 3.3, where the *normalized mean square error* (NMSE) is the prediction MSE normalized by $\boldsymbol{t}^T \boldsymbol{t}/N$. The performance is directly evaluated at the training samples $\boldsymbol{t}$. Thus, this experimental setup is more of a *sparse representation* kind as opposed to *sparse regression*, since we use no separate test dataset; cf. Section 1.4.

It can be seen from the plots in Figures 3.3(a) and 3.3(b) that FV-SBL with adjusted pruning condition is able to estimate the true sparsity of the signal by achieving the smallest normalized mean-square error (NMSE) almost over the whole tested SNR range. V-SBL as well as PX-VB, although they perform well in terms of the reached NMSE, do not produce sparse estimates. The reason for the V-SBL algorithm is that we had to abort the simulation in all averaging runs, since the required number of necessary iterations for the algorithm to terminate is enormous for a pruning threshold of 10^{12}. Note that the hyperparameters diverge linearly in V-SBL and for much smaller pruning thresholds similar results as for MML* are expected. However, for a fair comparison we had to use the same thresholds for

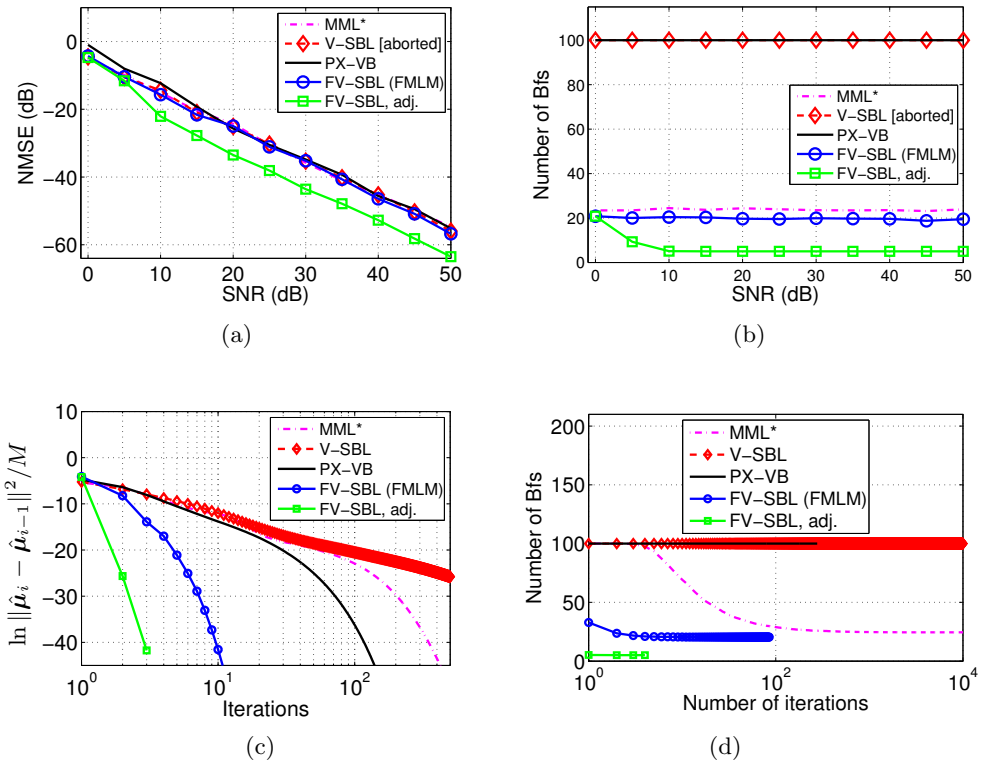


Figure 3.3: Random vector estimation. Results are averaged over 50 independent noise realizations. (a) Normalized mean square error (NMSE) versus the SNR; (b) the estimated number of components versus the SNR; (c) the convergence of the estimated weights versus the number of iterations for SNR=10dB; (d) the estimated number of basis functions (Bfs) versus the number of iterations for SNR=10dB.

MML*, V-SBL and PX-VB. The reason PX-VB does not produce sparse estimates, although it converges faster than V-SBL (see Fig. 3.3(c)), is that it quickly converges to a non-sparse local optimum of the variational objective function. FV-SBL without the adjustment (or equivalently FMLM) perform similar to MML*, V-SBL and PX-VB methods in terms of NMSE and slightly better in terms of the number of estimated components. In Figures 3.3(c) and 3.3(d) we demonstrate the convergence properties of the algorithms for an SNR of 10dB. Observe that FV-SBL clearly outperforms other estimation schemes; FV-SBL with adjusted pruning criterion converges rapidly (in roughly 3-4 iterations). PX-VB converges faster than MML* and V-SBL, but does not lead to a sparse estimate with the used pruning threshold as discussed before.

Now, let us investigate the algorithms on the CCS dataset. We normalized the

	MML*	V-SBL **[aborted]**	PX-VB	FV-SBL (FMLM)	FV-SBL ($\text{SNR}'_m = 10\text{dB}$)
# iterations	1774	(10^5)	294	13	**6**
NMSE (dB)	−15.74	(−16.06)	**−16.96**	−15.56	−14.41
#basis functions	66	(722)	722	55	**31**

Table 3.1: Performance results for Concrete Compressive Strength data. See text for details.

data to zero mean and unit variance; 70% of the data were picked at random for training and the remaining set was used for testing. The design matrix $\boldsymbol{\Phi}$ includes a bias term $\boldsymbol{\varphi}_0 = [1, \ldots, 1]^T$ and N Gaussian kernels $\boldsymbol{\varphi}_n = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}_n), \ldots, \kappa(\boldsymbol{x}_N, \boldsymbol{x}_n)]^T$ centered at the measurement samples, where $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\{-\theta_\kappa \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\}$. The variance of the additive noise $\hat{\tau}^{-1}$ as well as the Gaussian kernel parameter θ_κ were optimized by validating different settings on the test set with the FV-SBL algorithm; these parameters were then fixed at the estimated values $\hat{\tau}^{-1} = 0.1$ and $\theta_\kappa = 0.115$ for all compared methods. For the FV-SBL method with the adjusted pruning criterion we use $\text{SNR}'_m = 10\text{dB}$ for all components. Note that the $10dB$ assumption is somewhat arbitrary and, as we see later in Figure 3.8, is a particular choice for balancing *accuracy of fit* versus *sparsity*. Additionally, the assumption that the SNR'_m are equal for all $m = 1, \ldots, M$ components might also be incorrect for the CCS dataset. The corresponding performance results are summarized in Table 3.1.

Here again the FV-SBL approach outperforms the other methods. It achieves very sparse results with only 55 basis functions in 13 iterations, only marginally losing in NMSE as compared to MML*, V-SBL and PX-VB. The latter method achieves the best NMSE and is second to both FV-SBL schemes in terms of convergence rate; however, it does not lead to a sparse solution. Observe that the V-SBL algorithm is interrupted after 10^5 iterations. Although the sparsity parameters α_m continue to diverge, the rate of divergence is very low, which prevents them from reaching the used pruning threshold within our specified limit of iterations. Notice also that FV-SBL with the adjusted pruning condition leads to the sparsest estimator and converges very rapidly, but nonetheless loses in the achieved NMSE. As previously mentioned, the trade-off between NMSE and sparsity depends on how the parameters SNR'_m are set. In the following we analyze the influence of these parameters, set equally for each component, in more detail and on different datasets.

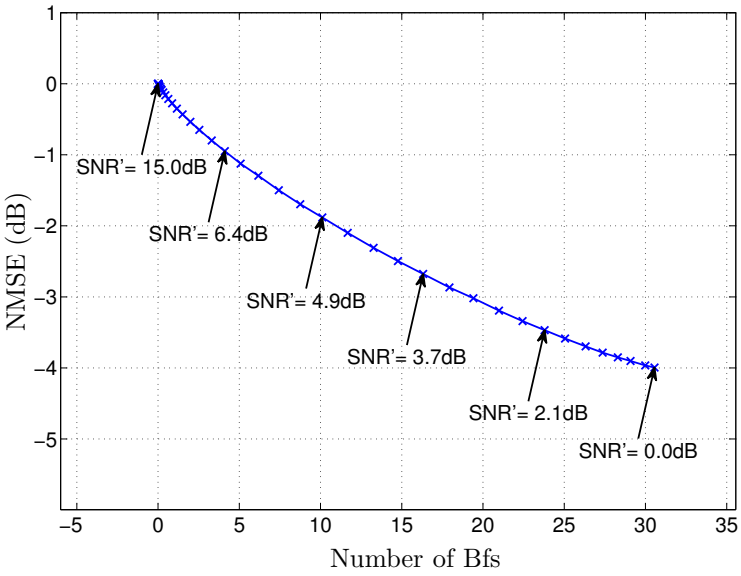
Investigations on the FV-SBL adjusted pruning condition

We now analyze the trade-off between *quality of fit* and *sparsity* for FV-SBL based on different settings for the adjusted threshhold SNR'_m from (3.20). We consider the homogeneous case, where $\mathrm{SNR}'_m = \mathrm{SNR}'$ for all m and use four different datasets. Namely, two realizations of the random basis experiment with 5 and 10 active components out of 100, a one-dimensional *sinc* function and the CCS dataset. For all four simulations, the noise precision τ was automatically estimated as $\hat{\tau} = \hat{c}/\hat{d}$ using the update (2.42) in a sequel amongst the other variational updates.

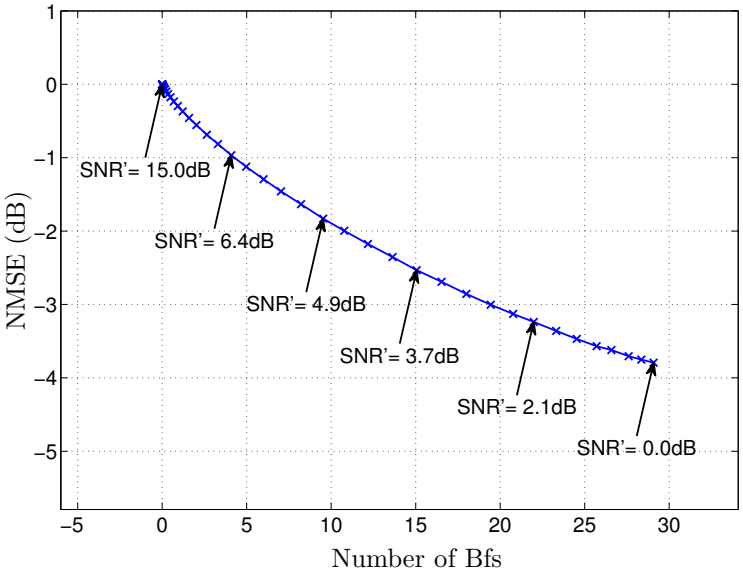
Let us now start with the random basis experiment, where we generate the data equivalently as before from $\boldsymbol{t} = \boldsymbol{\Phi}_{\mathrm{gen}}\boldsymbol{w}_{\mathrm{gen}} + \boldsymbol{\epsilon}_{\mathrm{gen}}$ with a sparse vector $\boldsymbol{w}_{\mathrm{gen}}$. More specifically, we generate two datasets $\boldsymbol{w}_{\mathrm{gen}}$ having 5 or 10 elements respectively with value 1 at random locations, where all other elements are zero. The matrix $\boldsymbol{\Phi}_{\mathrm{gen}}$, like before, is a random matrix of size 100×100 and is sampled from a standard Gaussian distribution. The elements of $\boldsymbol{\epsilon}_{\mathrm{gen}}$ are sampled independently from a zero-mean Gaussian distribution with the variance set to achieve an SNR of 10dB. For both datasets we analyze two situations, one with a model match, i.e., $\boldsymbol{\Phi} = \boldsymbol{\Phi}_{\mathrm{gen}}$ and one with a model mismatch, i.e., $\boldsymbol{\Phi} \neq \boldsymbol{\Phi}_{\mathrm{gen}}$. Note that, like above, we evaluate the prediction performance on the training samples $\boldsymbol{t}$; cf. *sparse representation* in Section 1.4.

First consider the case when we have a model mismatch. Since we have a design matrix $\boldsymbol{\Phi}$ different from the data generating matrix $\boldsymbol{\Phi}_{\mathrm{gen}}$, where $\boldsymbol{\Phi}$ is also sampled from a standard Gaussian, the error term $\boldsymbol{\epsilon}$ in (3.1) not only models the noise $\boldsymbol{\epsilon}_{\mathrm{gen}}$ in the data but also the model errors occurring from the differences of $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}_{\mathrm{gen}}$. The averaged results are shown in Figure 3.4, where we can clearly see that SNR' is a suitable design parameter to trade-off *quality of fit* vs. *sparsity* in FV-SBL models. We must note that there exists no optimal setting for SNR' without specifying this trade-off. That is, when we increase sparsity we sacrifice the *quality of fit* in terms of NMSE and vice versa.

Now let us consider the *model match* case. Here we have the same matrix for estimation as for data generation, i.e., $\boldsymbol{\Phi} = \boldsymbol{\Phi}_{\mathrm{gen}}$. The averaged results are shown in Figure 3.5, where we clearly see a difference compared to the model mismatch case. We now have a clear optimal setting for SNR' with values around the data SNR of 10dB, where the true sparsity, respectively for the 5 and 10 active component experiments, is revealed in both cases. For SNR' values smaller than 10dB, the number of basis components as well as the NMSE increases. For values larger then 10dB, the NMSE is also increasing while the number of components stays approximately



(a) 5 active components out of 100



(b) 10 active components out of 100

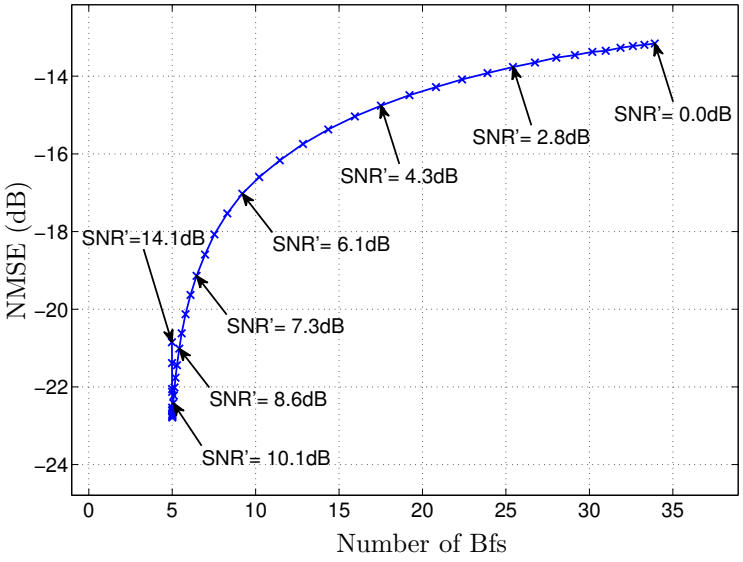
Figure 3.4: *Quality of fit* vs. *sparsity* for different SNR$'$ settings in a sparse vector estimation problem with model mismatch, i.e., $\mathbf{\Phi} \neq \mathbf{\Phi}_{\text{gen}}$. The data was generated with an SNR of 10dB and the results were averaged over 10000 noise realizations.

the same. Thus we have a clear optimum in this case. Note that for the *FV-SBL adj.* results presented previously in Figure 3.3(a) and 3.3(b) for different SNR, we assumed $\text{SNR}' = \text{SNR}$, i.e., we assumed to know the true SNR of the data. From the results presented now here in Figure 3.5 we see that this is indeed the optimal value if we have a *model match*, even though in many cases we do not know the true SNR.

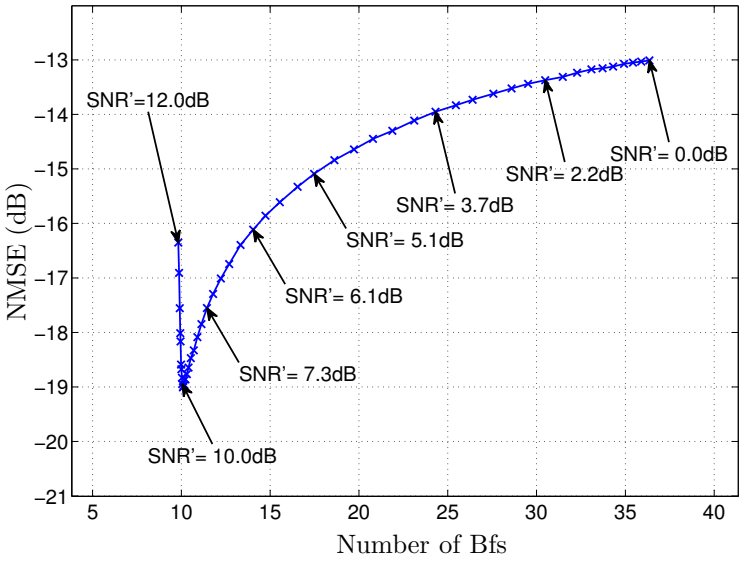
Figure 3.6 shows the averaged *quality of fit* vs. *sparsity* trade-off of the *sinc* data experiment for different SNR' values, where we used a constan bias basis function and Gaussian kernels with parameter $\theta_\kappa = 0.2174$ and $N = 100$ samples. Additive zero mean white Gaussian noise was added to the *sinc function samples* such that an SNR of 10dB was achieved. The inputs x where randomly placed in an interval $[-10, 10]$ drawn from a uniform distribution; cf. samples in Figure 3.7. Since we use a constant bias and Gaussians kernels as basis functions, we clearly have a *model mismatch* in this case. From Figure 3.6 we see that by adjusting the SNR' value, we can directly trade-off the *quality of fit* vs. the *sparsity* equivalently as in the random basis mismatch case. Also, we cannot determine an optimal SNR' value without selecting a *quality of fit* vs. *sparsity* trade-off that specifies what is more important for us. In Figure 3.7 we show *sinc* regression examples for different SNR' values.

In our final experiment, depicted in Figure 3.8, we show the *quality of fit* vs. *sparsity* trade-off for the CCS dataset which consists of $N = 1030$ samples. The results are obtained using 50-fold cross-validation. The data was first normalized and bias plus Gaussian kernels ($\theta_\kappa = 0.115$) were used as before. Thus, in this scenario we also have a *model mismatch*. Compared to the *Sinc* and *random basis model mismatch* experiment (see Figure 3.4 and Figure 3.6) the curve looks still similar but now shows a slightly sharper upwards turn. This may intuitively suggest that an SNR' in the region around 10dB to 14dB seems to be a better choice compared to other values. But again, without specifying the desired trade-off we cannot prefer any SNR' (except for some small jitter variations of the points, which we neglect here).

Considering all the presented experiments, we should keep in mind that with $\text{SNR}' = 0\text{dB}$ the algorithm is equivalent to unmodified FV-SBL and FMLM. We see from our different *quality of fit* vs. *sparsity* trade-off curves, that we have always obtained the smallest NMSE for $\text{SNR}' = 0\text{dB}$ when we have a *model mismatch*. On the other hand, when we have a *model match*, as shown in Figures 3.5, the worst NMSE was achieved over the plotted SNR' region in this case, because the SNR of the data was different from 0dB. That is, in the *model match* situation we have



(a) 5 active components out of 100



(b) 10 active components out of 100

Figure 3.5: *Quality of fit* vs. *sparsity* for different SNR′ settings in a sparse vector estimation problem with model match, i.e., $\mathbf{\Phi} = \mathbf{\Phi}_{\text{gen}}$. The data was generated with an SNR of 10dB and the results were averaged over 10000 noise realizations.

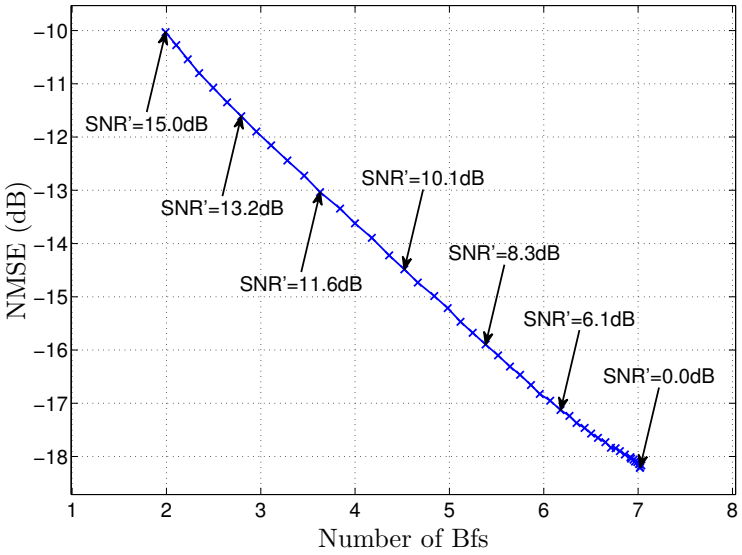


Figure 3.6: *Quality of fit* vs. *sparsity* for *sinc* data with different SNR′ settings. Data was generated with an SNR of 10dB and the results were averaged over 20000 noise realizations.

a clear optimum for SNR′ and do not have to trade-off *quality of fit* vs. *sparsity*, which we need for cases of *model mismatch*.

3.4 Discussion

In this chapter we have developed a fast variational sparse Bayesian learning (FV-SBL) algorithm. The algorithm is based on stationary point computations of the variational sparse Bayesian learning (V-SBL) update expressions when non-informative hyperpriors are used. This method, which dramatically increases the convergence speed, leads to equivalent pruning conditions as in fast marginal likelihood maximization (FMLM) [68]. It has been shown that for the case of non-informative hyperpriors, the mean of the sparsity parameter pdf that maximizes the variational lower bound also maximizes the marginal likelihood function with respect to this sparsity parameter. But, as opposed to FMLM, the form of the pruning conditions of FV-SBL reveals it as a comparison of a weight-component's SNR (when the influence of the component's hyperparameter is removed) with a 0dB threshold.

This perspective on the pruning condition brings up the idea of modifying the 0dB threshold to larger values. As it turns out, the modified threshold, which we denoted as SNR′, is an intuitive parameter for trading off *quality of fit* vs. *sparsity*.

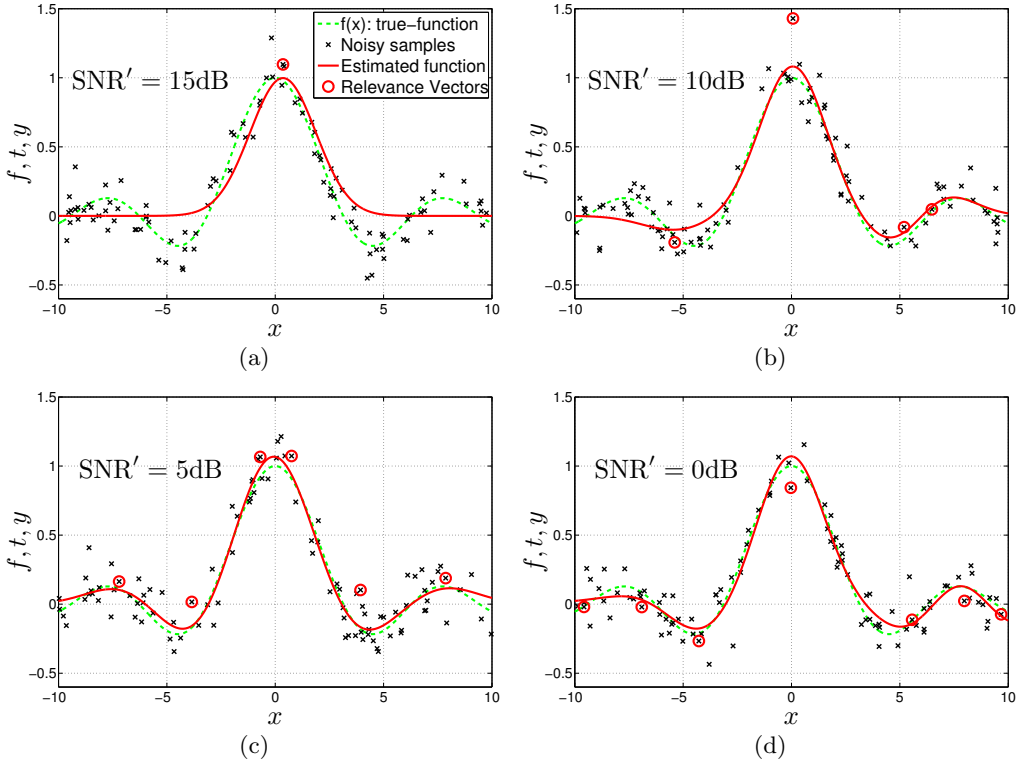


Figure 3.7: *Sinc* regression examples for different SNR′ values. The circle marked samples are denoted *relevance vectors* (RVs) [1]. Except for those kernel basis functions corresponding to RVs, all other kernel basis functions, which are centered on the inputs x_n of the samples, were pruned from the model.

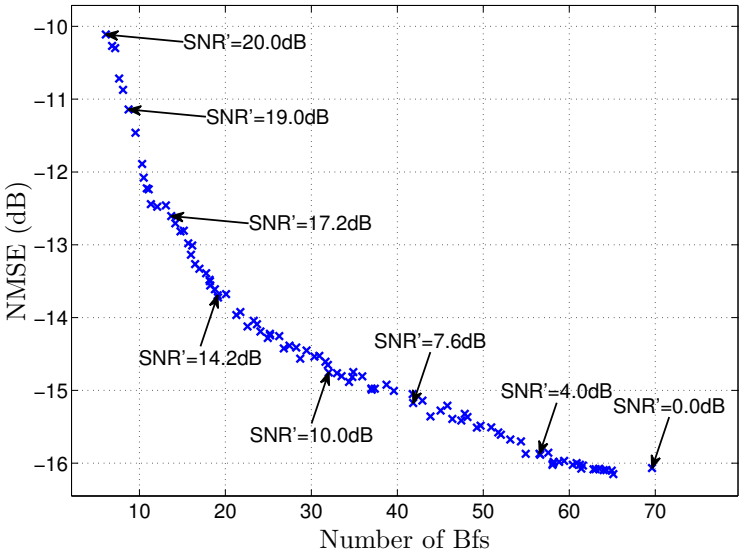


Figure 3.8: *Quality of fit* vs. *sparsity* for the CCS dataset with different SNR′ settings. Results were obtained using 50-fold cross-validation.

In our simulations we showed, that when we plot this trade-off over different settings of SNR′, the resulting curves look fundamentally different for cases with a *model match* compared to a *model mismatch*. That is, if the same basis components that have generated the data in the first place are available to the algorithm or not.

This analysis led to the general discussion in which situation we can prefer certain *quality of fit* and *sparsity* results amongst others. The simple answer is that, when we want to maximize both terms, we can only prefer results that are larger in both terms or in one term while the other is equivalent. In all other cases we need to specify a trade-off objective function that somehow specifies which term is more important to us, *quality of fit* or *sparsity*. We showed in our experiments that the case of a *model match* was the only situation that allowed us to improve both terms at the same time.

Chapter 4

Variational Distributed Sparse Bayesian Learning

In this chapter we discuss a variational distributed sparse Bayesian learning (V-dSBL) regression algorithm that is based on the idea of consensus propagation (CP) which we discussed in Section 2.3. It can be used for collaborative sparse estimation of spatial functions in wireless sensor networks (WSNs). The sensor measurements are modeled as a weighted superposition of basis functions equivalent to centralized SBL (see Section 2.2). When kernels are used, the algorithm can be seen as a distributed version of the relevance vector machine [23, 68]. The proposed method is based on a combination of variational inference and loopy belief propagation (LBP)[1], where data is only communicated between neighboring nodes without the need for a fusion center. We show that for tree structured networks, under certain parameterization, V-dSBL coincides with centralized V-SBL. For general loopy networks, V-dSBL and V-SBL are differend. However, simulations show that V-dSBL requires less hyperparameter updates as compared to V-SBL. That is, it converges faster over the variational inference iterations.

4.1 Bayesian model definition

Consider a sensor network with K sensors, where each sensor $k = 1, \ldots, K$ observes a scalar measurement t_k, which we also denote as the target signals. We assume that the measurements $t_1, \ldots, t_K$ are noisy observations of some unknown static field function $f(\boldsymbol{x})$, spatially sampled at the sensor positions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$, where $\boldsymbol{x}_k \in \mathbb{R}^D$

This chapter is partly based on our publications [71] and [72].

[1] Note that we use LBP as a subroutine of variational inference. The method is not related to *variational message passing* as in [102].

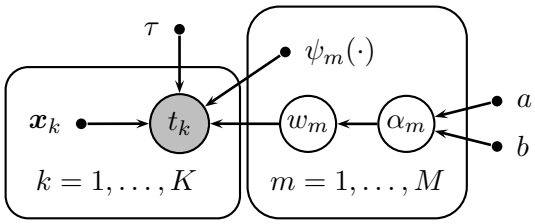


Figure 4.1: A Bayesian network representing the centralized SBL model with known noise precision τ.

and D is the coordinate system dimension (e.g., $D = 2$ for planar deployment)[2]. Since we assume a static field function, the set $\mathcal{D} = \{\boldsymbol{x}_k, t_k\}_{k=1}^K$ can be considered as constant distributed training data with inputs $\boldsymbol{x}_k$ and targets t_k only available to sensor k.

However, before we show how to learn from the training data $\mathcal{D}$ in a distributed fashion, we first assume $\mathcal{D}$ to be available at some central unit and return to our distributed considerations later in this chapter. Let us therefore first consider the centralized SBL model depicted as a Bayesian network in Figure 4.1. For simplicity, we assume the noise precision τ to be known. Note that it is also possible to estimate the noise precision in a distributed manner. We give some comments on how this could be done in Section 4.5.

The targets are modeled as

$$t_k = \sum_{m=1}^{M} w_m \psi_m(\boldsymbol{x}_k) + \epsilon_k, \tag{4.1}$$

a superpositions of M weighted basis functions $\psi_m(\cdot)$ with additive perturbation $\epsilon_k \sim \mathcal{N}(\epsilon_k|0, \tau^{-1})$. Remember from our discussion in Chapter 3 that since we do not know the underlying true function $f(\boldsymbol{x})$, and our model (4.1) may not be rich enough to perfectly represent any f, the perturbation term ϵ_k can be seen not only as including sensor measurement noise, but also any model mismatches.

As in Section 2.2, we define the hierarchical prior on the weights and hyperparameters as $p(w_m|\alpha_m)p(\alpha_m)$ with $p(w_m|\alpha_m) = \mathcal{N}(w_m|0, \alpha_m^{-1})$ and $p(\alpha_m) = \mathrm{Ga}(\alpha_m|a, b) \propto \alpha_m^{a-1} e^{-b\alpha_m}$.

[2]Note that although we use $\boldsymbol{x}_k$ as sensor coordinates here, it could also be any other kind of input vector available at sensor k.

4.2 Variational approximation

We now derive the V-SBL algorithm with known noise precision τ. To this end we define the VB approximation of the intractable SBL posterior $p(\boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{t})$ as

$$q(\boldsymbol{w}, \boldsymbol{\alpha}) = q(\boldsymbol{w})q(\boldsymbol{\alpha}) \;\approx\; p(\boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{t}) = \frac{p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(\boldsymbol{t})}, \tag{4.2}$$

where we again use the vector notations $\boldsymbol{w} = [w_1, \ldots, w_M]^T$, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_M]^T$ and $\boldsymbol{t} = [t_1, \ldots, t_K]^T$ for the weights, hyperparameters and targets, respectively. The factorization of $q(\boldsymbol{w})$ and $q(\boldsymbol{\alpha})$ in (4.2), known as structured mean field approximation, forms the only relaxation to the problem. By maximizing the variational lower bound [4] (cf. Section 2.1.1)

$$\mathcal{L}[q(\boldsymbol{w}, \boldsymbol{\alpha})] = \int q(\boldsymbol{w}, \boldsymbol{\alpha}) \ln \frac{p(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{t})}{q(\boldsymbol{w}, \boldsymbol{\alpha})} \, d\boldsymbol{w} d\boldsymbol{\alpha}, \tag{4.3}$$

we obtain the optimal unconstrained updates as

$$q^*(\boldsymbol{w}) = \frac{\exp\left(\mathbb{E}_{q(\boldsymbol{\alpha})}\{\ln[\, p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})\,]\}\right)}{\int \exp\left(\mathbb{E}_{q(\boldsymbol{\alpha})}\{\ln[\, p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})\,]\}\right) d\boldsymbol{w}} \tag{4.4}$$

and

$$q^*(\boldsymbol{\alpha}) = \frac{\exp\left(\mathbb{E}_{q(\boldsymbol{w})}\{\ln[\, p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})\,]\}\right)}{\int \exp\left(\mathbb{E}_{q(\boldsymbol{w})}\{\ln[\, p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})\,]\}\right)} \tag{4.5}$$

By solving (4.4) and (4.5), we obtain $q^*(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$ and $q^*(\boldsymbol{\alpha}) = \prod_{m=1}^{M} \mathrm{Ga}(\alpha_m|\hat{a}_m, \hat{b}_m)$ (cf. (2.40) and (2.41) in Section 2.2.2), with

$$\hat{\boldsymbol{\Sigma}} = \left(\hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathrm{diag}(\hat{\boldsymbol{\alpha}})\right)^{-1}, \quad \hat{\boldsymbol{\mu}} = \hat{\tau}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{t}, \tag{4.6}$$

and

$$\hat{a}_m = a + 1/2, \quad \hat{b}_m = b + (\hat{\mu}_m^2 + \hat{\Sigma}_{mm})/2, \tag{4.7}$$

where $\hat{\alpha}_m = \mathbb{E}_{q(\alpha_m)}\{\alpha_m\} = \hat{a}_m/\hat{b}_m$, $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_1, \ldots, \hat{\alpha}_M]^T$, $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K]^T$, $\boldsymbol{\phi}_k = [\psi_1(\boldsymbol{x}_k), \ldots, \psi_M(\boldsymbol{x}_k)]^T$, $\hat{\mu}_m$ is the mth element of the mean vector $\hat{\boldsymbol{\mu}}$, and $\hat{\Sigma}_{mm}$ is the mth element on the main diagonal of the covariance matrix $\hat{\boldsymbol{\Sigma}}$.

Consecutive updates of (4.6) and (4.7) lead to the desired posterior approximation (4.2). Components of the density $q(\boldsymbol{w})$, which have large probability mass around zero have no influence on the model (4.1), as we also discussed in Sec-

tion 2.2.2. These components, which correspond to irrelevant basis functions, can be equivalently detected by large elements of $\hat{\boldsymbol{\alpha}}$. Practically, one defines a large threshold θ_{th} and considers all $\hat{\alpha}_m > \theta_{\text{th}}$, for $m = 1, \ldots, M$, as infinite, where the corresponding basis functions can be pruned from the model.

4.3 Distributed sparse Bayesian learning

Let us now extend the centralized SBL model in such a way that we can derive the V-dSBL algorithm step by step. The fundamental problem is how to obtain a distributed versions of the variational updates (4.4) and (4.5).

First, consider the update expression (4.5). If we knew $q(\boldsymbol{w})$ at each sensor in the network, i.e., if we knew its parameters $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$, we could compute $\hat{\boldsymbol{\alpha}}$, which fully specifies $q(\boldsymbol{\alpha})$, from (4.7). This is because $\hat{a}_m$ is constant and $\hat{b}_m = \hat{a}_m/\hat{\alpha}_m$. That means, updating $q(\boldsymbol{\alpha})$ is only a local computation at each sensor in the network if $q(\boldsymbol{w})$ is known everywhere.

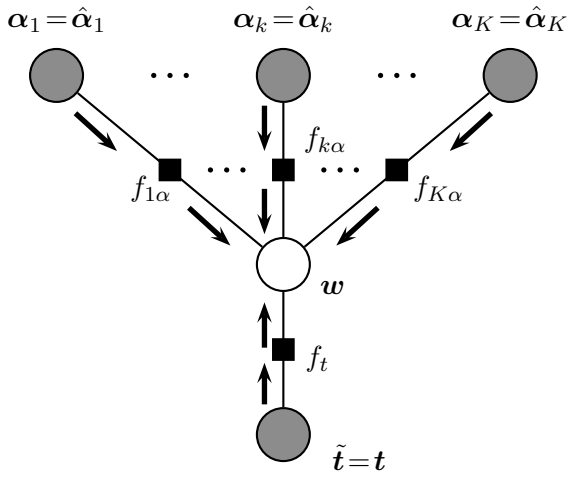
Second, we modify the update expression (4.4) for $q(\boldsymbol{w})$ slightly to obtain an intermediate model with K hyperparameter vectors $\hat{\boldsymbol{\alpha}}_1, \ldots, \hat{\boldsymbol{\alpha}}_K$, one for each sensor k with $\hat{\boldsymbol{\alpha}}_k = [\hat{\alpha}_{k,1}, \ldots, \hat{\alpha}_{k,M}]^T$. Then, we introduce a factor graph representation and show that we can obtain $q(\boldsymbol{w})$ when $\hat{\boldsymbol{\alpha}}_1 = \ldots = \hat{\boldsymbol{\alpha}}_K = \hat{\boldsymbol{\alpha}}$ by using message passing inference schemes.

Finally, we tackle the remaining question. Namely, how to obtain $q(\boldsymbol{w})$ at each sensor when our data $\mathcal{D}$ is distributed across the network. Since we are not interested in collecting all the data $\{\boldsymbol{x}_i, t_i\}$ for $i \neq k$ at each sensor k, we propose a further modification of the update expression (4.4) that better resembles the distributed nature of our problem. The resulting factor graph representation includes multiple marginal densities $q(\boldsymbol{w}_k)$ at each sensor k with $\boldsymbol{w}_k = [w_{k,1}, \ldots, w_{k,M}]^T$. These marginals can be estimated by applying LBP message passing which only requires communication between neighboring nodes. We show that under certain assumptions, $q(\boldsymbol{w}_k) = q(\boldsymbol{w})$, $\forall k$.

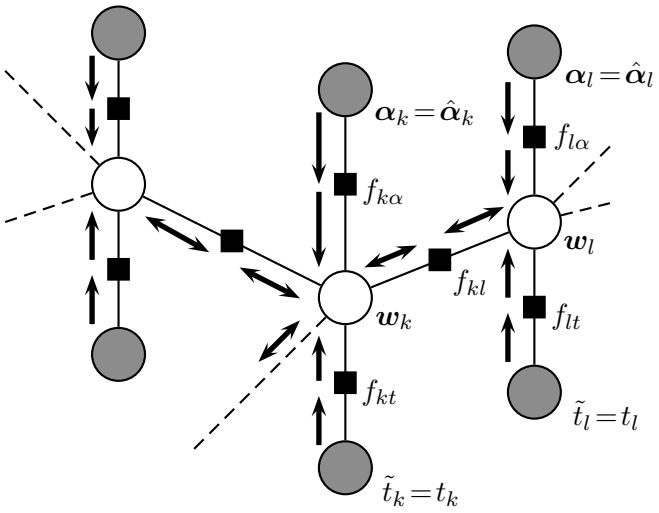
4.3.1 Factor graph representation

We have already introduced factor graphs in the context of CP in Section 2.3.

We now show that the weight update expression (4.4), defined by its sufficient statistics (4.6), can be obtained through inference performed on the factor graph in Figure 4.2(a), where we now have multiple vectors $\hat{\boldsymbol{\alpha}}_1, \ldots, \hat{\boldsymbol{\alpha}}_K$ instead of one



(a) Distributing the hyperparameters



(b) V-dSBL

Figure 4.2: Message flow on two factor graphs representing (a) an intermediate model to derive (b) variational distributed SBL (V-dSBL).

vector $\hat{\boldsymbol{\alpha}}$. The factors of the graph in Fig. 4.2(a) are specified as[3]

$$f_t(\tilde{\boldsymbol{t}}, \boldsymbol{w}) = \exp\{-\frac{\tau}{2}\|\tilde{\boldsymbol{t}} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2\}, \tag{4.8}$$

[3]Note that factors need not to be normalized.

and

$$f_{k\alpha}(\boldsymbol{\alpha}_k, \boldsymbol{w}) = \exp\{-\frac{1}{2K}\boldsymbol{w}^T \boldsymbol{A}_k \boldsymbol{w}\}, \ \ \forall k, \tag{4.9}$$

where $\boldsymbol{A}_k = \text{diag}(\boldsymbol{\alpha}_k)$.

Note that for the definition of the local factors (4.9), each sensor k needs to know K, the total number of sensors in the network. Although we assume that K is known everywhere, it can be easily obtained via consensus algorithms like [104, 67], as pointed out in [120]. The fundamental idea is to average the value 1 at one node with the values 0 at all other nodes accross the network. Using consensus algorithms, each node should obtains $1/K$, which can be locally inverted to obtain K. Note that this estimation method, equivalently to the concepts derived here for V-dSBL in the following, can be performed fully decentralized without the need for routing protocols.

Let us now apply the *sum-product* algorithm [4] to the factor graph in Figure 4.2(a) by noting that due to our previous discussion about distributed hyperparameter updates, when $q(\boldsymbol{w})$ is available everywhere, we have $\hat{\boldsymbol{\alpha}}_k = \hat{\boldsymbol{\alpha}}$, $\forall k$, where $\hat{\boldsymbol{\alpha}}_k$ is the expectation of the hyperparameter vector at sensor k. We start to pass messages from the leave nodes to $\boldsymbol{w}$ and consider observed variables as Dirac delta distributions, i.e., $m_{\boldsymbol{\alpha}_k \to f_{k\alpha}} = \delta(\boldsymbol{\alpha}_k - \hat{\boldsymbol{\alpha}}_k)$, $\forall k$, and $m_{\tilde{\boldsymbol{t}} \to f_{kt}} = \delta(\tilde{\boldsymbol{t}} - \boldsymbol{t})$. Furthermore we have the messages from the factors to $\boldsymbol{w}$ defined as the integral over the product of all incoming messages times the factors itself, i.e., $m_{f_{k\alpha} \to \boldsymbol{w}} = f_{k\alpha}(\hat{\boldsymbol{\alpha}}_k, \boldsymbol{w})$, $\forall k$, and $m_{f_{kt} \to \boldsymbol{w}} = f_t(\boldsymbol{t}, \boldsymbol{w})$. Finally, we compute $q(\boldsymbol{w})$ as the normalized product of all incoming messages given as

$$\frac{1}{Z} f_t(\boldsymbol{t}, \boldsymbol{w}) \prod_{k=1}^{K} f_{k\alpha}(\hat{\boldsymbol{\alpha}}_k, \boldsymbol{w}), \tag{4.10}$$

where Z is a normalization constant. By plugging (4.8) and (4.9) into (4.10), it is now straight forward to show that (4.10) is a Gaussian distribution with mean vector and covariance matrix as defined in (4.6) and thus equivalent to centralized V-SBL.

4.3.2 Distributed computation of $q(\boldsymbol{w})$

So far we have assumed that $q(\boldsymbol{w})$ is known to all sensors. Before we investigate on how to distribute $q(\boldsymbol{w})$ across the network, let us define a WSN as an undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which consists of vertices (or sensors) $\mathcal{V}$ and edges (or links) $\mathcal{E}$ defined as a set of unordered pairs $\{k, l\} \subset \mathcal{V}$ representing the communi-

cation links. Note that by defining an undirected graph, we implicitly assume that sensors can communicate in both directions along a communication link[4]. We define the set of neighbors of sensor k as $N(k) = \{l \mid \{k,l\} \in \mathcal{E}\}$ and its cardinality, i.e., the number of neighbors of sensor k, as $\#N(k)$.

Tree Networks

When we consider tree networks, i.e., networks without loops, we can define a factor graph, as depicted in Figure 4.2(b), that has marginal distributions $q(\boldsymbol{w}_k),\ \forall k$, which are equal to the centralized $q(\boldsymbol{w})$ distribution. That is, we can obtain $q(\boldsymbol{w}_k) = q(\boldsymbol{w})$ at each sensor in the network. The marginals $q(\boldsymbol{w}_k)$ can be efficiently obtained by again using the *sum-product* algorithm, where we pass messages from the leaves to an arbitrary root node and from there back to the leaves [4]. Let us define the new factors used in Figure 4.2(b), as

$$f_{kt}(\tilde{t}_k, \boldsymbol{w}_k) = \exp\{-\frac{\tau}{2}(\tilde{t}_k - \boldsymbol{\phi}_k^T \boldsymbol{w}_k)^2\},\ \forall k, \tag{4.11}$$

and

$$f_{kl}(\boldsymbol{w}_k, \boldsymbol{w}_l) = \exp\left\{-\frac{\beta}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_l\|_2^2\right\},\ \forall \{k,l\} \in \mathcal{E}, \tag{4.12}$$

where (4.11), as opposed to (4.8), now has only a single (the locally available) target variable $\tilde{t}_k$, and we denote (4.12) as the *coupling factors*. The factors $f_{k\alpha}$ are defined as in (4.9), but are now connected to a local $\boldsymbol{w}_k$ instead of a global $\boldsymbol{w}$. Compared to Figure 4.2(a), the factors (4.12) are conceptually new and form a fundamental part in the V-dSBL model. Wherever sensors k and l are connected in the WSN graph, a factor f_{kl} is also defined in the factor graph, i.e., the physical structure $\mathcal{G}$ coincides with the probabilistic structure between the weights $q(\boldsymbol{w}_k)$. That is, the *sum-product* algorithm can be distributedly performed by the WSN, where no routing scheme with any kind of multi-hop communication is needed. By inspecting (4.12) with $\beta > 0$, we have larger values of f_{kl} when $\boldsymbol{w}_k$, in terms of ℓ_2 distance, is close to $\boldsymbol{w}_l$, which is the driving force to achieve consensus in the network since there is a larger probability that neighbors have similar weight distributions. This coupling is even stronger when the precision parameter β, further denoted as coupling parameter, is large. When $\beta \to \infty$, then $f_{kl}(\boldsymbol{w}_k, \boldsymbol{w}_l) \propto \delta(\boldsymbol{w}_k - \boldsymbol{w}_l)$ and we obtain the same weight distributions everywhere.

In the following, we generally derive the messages and show that as $\beta \to \infty$, for

[4]It is assumed that the physical and logical topologies in the communication network coinside.

tree structured graphs $\mathcal{G}$, the equivalence $q(\boldsymbol{w}_k) = q(\boldsymbol{w})$ holds for all k. Similar as before, the messages from the observations to $\boldsymbol{w}_k$ can be determined as $m_{\tilde{t}_k \to f_{kt}} = \delta(\tilde{t}_k - t_k)$ and $m_{f_{kt} \to \boldsymbol{w}_k} = f_{kt}(t_k, \boldsymbol{w}_k)$, $\forall k$. In the same manner we obtain $m_{\boldsymbol{\alpha}_k \to f_{k\alpha}} = \delta(\boldsymbol{\alpha}_k - \hat{\boldsymbol{\alpha}}_k)$ and $m_{f_{k\alpha} \to \boldsymbol{w}_k} = f_{k\alpha}(\hat{\boldsymbol{\alpha}}_k, \boldsymbol{w}_k)$, $\forall k$. Since all factors in Figure 4.2(b) have Gaussian shape, also the messages between the sensors have Gaussian shape. Additionally, as normalization can be done at any stage during message passing, we further assume normalized messages for simplicity. Thus, we can generally define the incoming message at node $\boldsymbol{w}_k$, coming from some factor f_{uk}, as normalized Gaussian

$$m_{f_{uk} \to \boldsymbol{w}_k} \propto \exp\left\{ -\frac{1}{2} (\boldsymbol{w}_k - \hat{\boldsymbol{\mu}}_{uk})^T \hat{\boldsymbol{\Lambda}}_{uk} (\boldsymbol{w}_k - \hat{\boldsymbol{\mu}}_{uk}) \right\}, \tag{4.13}$$

with mean vector $\hat{\boldsymbol{\mu}}_{uk}$ and precision matrix (inverse covariance matrix) $\hat{\boldsymbol{\Lambda}}_{uk}$. We can determine the messages send from sensor k to sensor l as[5]

$$m_{\boldsymbol{w}_k \to f_{kl}} \propto m_{f_{kt} \to \boldsymbol{w}_k} \, m_{f_{k\alpha} \to \boldsymbol{w}_k} \prod_{u \in N(k) \backslash l} m_{f_{uk} \to \boldsymbol{w}_k}, \tag{4.14}$$

and

$$m_{f_{kl} \to \boldsymbol{w}_l} \propto \int f_{kl}(\boldsymbol{w}_k, \boldsymbol{w}_l) \, m_{\boldsymbol{w}_k \to f_{kl}} \, d\boldsymbol{w}_k, \tag{4.15}$$

where the parameters of the Gaussian message (4.15) can be obtained from standard results for multivariate Gaussian distributions (see, e.g., [4, Appendix B]). That is,

$$\hat{\boldsymbol{\Lambda}}_{kl} = \left(\left(\frac{\hat{\boldsymbol{A}}_k}{K} + \tau \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T + \sum_{u \in N(k) \backslash l} \hat{\boldsymbol{\Lambda}}_{uk} \right)^{-1} + \beta^{-1} \boldsymbol{I} \right)^{-1}, \tag{4.16}$$

$$\hat{\boldsymbol{\mu}}_{kl} = (\hat{\boldsymbol{\Lambda}}_{kl}^{-1} - \beta^{-1} \boldsymbol{I}) \Big(\tau \boldsymbol{\phi}_k t_k + \sum_{u \in N(k) \backslash l} \hat{\boldsymbol{\Lambda}}_{uk} \hat{\boldsymbol{\mu}}_{uk} \Big), \tag{4.17}$$

with $\hat{\boldsymbol{A}}_k = \mathrm{diag}(\hat{\boldsymbol{\alpha}}_k)$. Passing messages (4.16) and (4.17) from the leave nodes to an arbitrary root node $\boldsymbol{w}_k$ in the tree network, leads to the marginal distribution $q(\boldsymbol{w}_k)$ with mean vector $\hat{\boldsymbol{\mu}}_k$ and precision matrix $\hat{\boldsymbol{\Lambda}}_k$ defined as

$$\hat{\boldsymbol{\Lambda}}_k = \frac{\hat{\boldsymbol{A}}_k}{K} + \tau \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T + \sum_{u \in N(k)} \hat{\boldsymbol{\Lambda}}_{uk} \tag{4.18}$$

[5]The notation $N(k) \backslash l$ denotes the set of all neighbors of k except l.

and

$$\hat{\boldsymbol{\mu}}_k = \hat{\boldsymbol{\Lambda}}_k^{-1}\Big(\tau\boldsymbol{\phi}_k t_k \; + \sum_{u\in N(k)} \hat{\boldsymbol{\Lambda}}_{uk}\hat{\boldsymbol{\mu}}_{uk}\Big). \tag{4.19}$$

Passing messages from the root node back to all other nodes $\boldsymbol{w}_l$, for $l \neq k$, allows all other marginal distributions to be obtained. Now, inserting $\beta \to \infty$, for trees it is easy to verify that

$$\big(\hat{\boldsymbol{\Lambda}}_k^{[\beta\to\infty]}\big)^{-1} = \left(\frac{1}{K}\sum_{k=1}^{K}\hat{\boldsymbol{A}}_k + \tau\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1} \tag{4.20}$$

and

$$\hat{\boldsymbol{\mu}}_k^{[\beta\to\infty]} = \tau\left(\hat{\boldsymbol{\Lambda}}_k^{[\beta\to\infty]}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}, \tag{4.21}$$

which is equivalent to (4.6) for $\hat{\boldsymbol{\alpha}}_k = \hat{\boldsymbol{\alpha}}$, $\forall k$, as we intended to show. That is, $\big(\hat{\boldsymbol{\Lambda}}_k^{[\beta\to\infty]}\big)^{-1} = \hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{\mu}}_k^{[\beta\to\infty]} = \hat{\boldsymbol{\mu}}$ for all k.

General Loopy Networks

When we consider loopy graphs $\mathcal{G}$, which is a more general model for a WSN, applying the *sum-product* algorithm leads to LBP, an iterative approximate inference method [4]. Convergence of LBP cannot be guaranteed in general, but for the particular case of Gaussian LBP, which applies here, there exist convergence criteria for the marginal mean and marginal variance; see [107, 106].

Unfortunately, these criteria are not suitable for V-dSBL, since they are designed for what we denote as *elementary Gaussian LBP*, i.e., when we consider messages between all individual scalar random variables. More specifically, in V-dSBL as depicted in Figure 4.2(b) we have multivariate Gaussian random variables, i.e., vectors instead of scalars. Thus, in V-dSBL the messages are also multivariate, which is not considered in [107, 106]. In this sense, even if we have tree networks as discussed before, the elementary Gaussian graph has loops because the elements of all vectors $\boldsymbol{w}_k$, i.e., $w_{k,1}, \ldots, w_{k,M}$, in general are fully connected. This is already due to the local factors f_{kt} at each node. Note that when we see the factor f_{kt} as a multivariate Gaussian over $\boldsymbol{w}_k$, the resulting precision matrix $\tau\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T$ typically has only non-zero off-diagonal elements and thus corresponds already to a fully connected graph [106].

However, as the messages sent in V-dSBL are costly to communicate and due to the matrix inversion (4.16) also costly to compute, elementary Gaussian LBP seems to be an efficient alternative. Unfortunately, in our experiments elementary Gaussian LBP almost never converged. Also the sufficient convergence criterion derived

in [106] was almost never fulfilled. In Appendix C, we give more details about the elementary Gaussian LBP approach to V-dSBL and discuss further research directions to overcome the convergence problems. We also show in detail in Appendix C how the sufficient convergence criterion [106] can be evaluated for the V-dSBL model.

In consensus propagation (CP), as pointed out in [67], the coupling parameter β, is critical to trade off fast convergence (small β) and accurate mean estimates (large β). In V-dSBL, the same applies for β, since the V-dSBL coupling factor (4.12) is just a vector extention of the coupling factors in CP defined in Section 2.3.

Furthermore, in loopy graphs, we need to define a schedule in which messages are updated until convergence. Algorithm 4.1 presents V-dSBL with synchronous message updates, where we have reformulated the messages as functions of intermediate marginals, which directly follow from (4.16)-(4.19). Note that we only consider synchronous message updates in all our simulations throughout this thesis.

Algorithm 4.1 Variational distributed SBL (V-dSBL)

Initialize: $\hat{\boldsymbol{\alpha}}_k$ (e.g., $\hat{\alpha}_{k,m} = 10^{-1}$, $\forall m$), $\forall k$.

% Variational update loop
while (Not converged) **do**
  Initialize: $n = 0$, $\hat{\boldsymbol{\Lambda}}_{uk}^{(0)} = \mathbf{0}$ and $\hat{\boldsymbol{\mu}}_{uk}^{(0)} = \boldsymbol{0}$, $\forall (u,k) \in \mathcal{E}$

  % Message passing update loop
  while (LBP not converged) **do**
    $n = n + 1$
    % Compute intermediate marginal estimates, $\forall k$
    $\hat{\boldsymbol{\Lambda}}_k^{(n)} = K^{-1}\hat{\boldsymbol{A}}_k + \tau\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T + \sum_{u\in N(k)} \hat{\boldsymbol{\Lambda}}_{uk}^{(n-1)}$
    $\hat{\boldsymbol{\mu}}_k^{(n)} = \left(\boldsymbol{\Lambda}_k^{(n)}\right)^{-1}\left(\tau\boldsymbol{\phi}_k t_k + \sum_{u\in N(k)} \hat{\boldsymbol{\Lambda}}_{uk}^{(n-1)}\hat{\boldsymbol{\mu}}_{uk}^{(n-1)}\right)$
    % Send messages, $\forall (k,l) \in \mathcal{E}$
    $\hat{\boldsymbol{\Lambda}}_{kl}^{(n)} = \left(\left(\hat{\boldsymbol{\Lambda}}_k^{(n)} - \hat{\boldsymbol{\Lambda}}_{lk}^{(n-1)}\right)^{-1} + \beta^{-1}\boldsymbol{I}\right)^{-1}$
    $\hat{\boldsymbol{\mu}}_{kl}^{(n)} = \left(\hat{\boldsymbol{\Lambda}}_k^{(n)} - \hat{\boldsymbol{\Lambda}}_{lk}^{(n-1)}\right)^{-1}\left(\hat{\boldsymbol{\Lambda}}_k^{(n)}\hat{\boldsymbol{\mu}}_k^{(n)} - \hat{\boldsymbol{\Lambda}}_{lk}^{(n-1)}\hat{\boldsymbol{\mu}}_{lk}^{(n-1)}\right)$
  end while

  % Compute the marginal estimates, $\forall k$
  $\hat{\boldsymbol{\Sigma}}_k = \left(K^{-1}\hat{\boldsymbol{A}}_k + \tau\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T + \sum_{u\in N(k)} \hat{\boldsymbol{\Lambda}}_{uk}^{(n)}\right)^{-1}$
  $\hat{\boldsymbol{\mu}}_k = \hat{\boldsymbol{\Sigma}}_k\left(\tau\boldsymbol{\phi}_k t_k + \sum_{u\in N(k)} \hat{\boldsymbol{\Lambda}}_{uk}^{(n)}\hat{\boldsymbol{\mu}}_{uk}^{(n)}\right)$

  % Update hyperparameters, $\hat{\alpha}_{k,m}$, $\forall m$, $\forall k$
  $\hat{\alpha}_{k,m} = (2a+1)/(2b + \hat{\mu}_{k,m}^2 + \hat{\Sigma}_{k,mm})$

end while
Compute the final marginals with LBP as before

In conclusion, as the convergence criteria [107, 106] do not apply for V-dSBL, we cannot guarantee convergence of the marginal distribution estimates, i.e., marginal mean vectors and marginal covariance matrices. Furthermore, if the marginal mean vectors and marginal covariance matrices converge, we cannot guarantee their correctness[6].

However, in none of our simulations presented in Section 4.4 we had convergence problems. Especially for the synthetic data experiments presented in Section 4.4.1 V-dSBL performed similar to V-SBL. This was achieved at a much faster convergence rate over the variational inference iterations. That is, the algorithm requires less iterations in the variational update loop (outer loop in Algorithm 4.1) compared to centralized V-SBL. We further show in the real data experiments presented in Section 4.4.2, that the performance V-dSBL strongly depends on the network topology, more specifically on the number of connections. For a large number of connections, even though Gaussian LBP always converged, the performance was not that good.

4.3.3 Exploiting sparsity for efficient communication

When a hyperparameter $\hat{\alpha}_{k,m}$ diverges, or practically considered as infinite after reaching a large predefined threshold θ_{th}, the basis function m can be pruned at node k. Also the messages, i.e., $\hat{\boldsymbol{\mu}}_{kl}$ and $\hat{\boldsymbol{\Lambda}}_{kl}$, can be transmitted to all neighbors $l \in N(k)$ without the irrelevant components.

By inspecting Equation (4.16), it is easy to show that the inner inverse has only zeros in the m-th row and column if $\hat{\alpha}_{k,m} \to \infty$. When we also consider the addition of the $\beta^{-1}\mathbf{I}$ term, after computing the outer inverse, we obtain a matrix $\hat{\boldsymbol{\Lambda}}_{kl}$ with the m-th row and column equal to zero, except the m-th main diagonal element which is β. This can be easily shown by using matrix permutation and blockwise inversion rules. Since β is a design parameter in our model and is known by all sensors, there is no need for transmitting it to the neighbors. Thus, we can delete the m-th row and column of $\hat{\boldsymbol{\Lambda}}_{kl}$ and transmit the resulting sparse message, which we denote $\breve{\boldsymbol{\Lambda}}_{kl}$. The receiving sensor l, which is getting the message $\breve{\boldsymbol{\Lambda}}_{kl}$, can reinsert β in the diagonal and thus reproduce $\hat{\boldsymbol{\Lambda}}_{kl}$ if it knows the correct positions of the pruned elements. Thus, the only information we need to transmit additionally from a sensor k to l is a binary mask of size M indicating the used basis functions[7].

[6] Note that even for elementary Gaussian LBP, as we further discuss in Appendix C, if the algorithm converges, only the mean is guaranteed to be correct [107, 106].

[7] Note that the binary mask does not add much extra communication, since M is limited anyway. This is due to the initial communicational complexity of $O(M^2)$ stemming from the precision matrix (4.16).

Similarly, it can be shown that the mean message $\hat{\boldsymbol{\mu}}_{kl}$ defined in (4.17) obtains zero elements at all positions m if $\hat{\alpha}_{k,m}$ diverges. After pruning the zero elements from the vector, we send the sparse mean message $\breve{\boldsymbol{\mu}}_{kl}$, which can also be reproduced at sensor l to obtain $\hat{\boldsymbol{\mu}}_{kl}$ by reinserting zeros according to the binary mask.

It is important to mention that also all local computations can be done efficiently by taking sparsity patterns into account, i.e., to consider only the matrix and vector elements corresponding to finite $\hat{\alpha}_{k,m}$ values similar to the previous discussion. For each $\hat{\alpha}_{k,m} = \infty$, the marginal covariance matrix $\hat{\boldsymbol{\Sigma}}_k$ at sensor k, defined in Algorithm 4.1, only consists of zeros in the m-th row and m-th column. Likewise the mean vector $\hat{\boldsymbol{\mu}}_k$ has a zero at position m. Thus, the corresponding elements, including the corresponding hyperparameters, can be pruned from the local model.

4.4 Simulations

In this section we show simulation results for synthetic data and real data experiments, where we compare the performance to centralized V-SBL. For both, the synthetic data and real data experiment, we define the basis functions like in RVMs, i.e., a bias $\psi_1(\boldsymbol{x}) = 1$ and Gaussian kernels $\psi_m(\boldsymbol{x}) = \kappa(\boldsymbol{x}, \boldsymbol{x}_{m-1})$ for $m = 2, \ldots, K+1$ centered on the sensor positions[8], where $\kappa(\boldsymbol{x}, \boldsymbol{x}_{m-1}) = \exp\{-\theta_\kappa \|\boldsymbol{x} - \boldsymbol{x}_{m-1}\|^2\}$ with kernel parameter θ_κ.

4.4.1 Synthetic data

We randomly deploy $K = 50$ sensors in a $D = 2$ dimensional area, where both coordinates lie in the interval $[0, 1]$. Figure 4.3 shows the sensor positions and connections, where we generated the connections by increasing the transmission radius equally for each sensor until a connected graph was obtained. The sensor measurements (targets) are generated by adding zero-mean white Gaussian noise with variance $\sigma^2 = 10^{-3}$ to the underlying spatial function $f(\boldsymbol{x}) = \frac{1}{2}\operatorname{sinc}(5x_1 - \frac{5}{2})) + \frac{1}{2} + x_2$. We assume $\tau = \sigma^{-2}$, where this is only an approximation, since we neglect any possible model mismatches. Nonetheless, this is a good choice since we do not know how well our model fits the data in advance. Comments on possible implementations for distributed estimation of τ are given in Section 4.5. The hyperprior parameters a

[8] For kernel basis functions, it is assumed that each sensor knows about the position of the others initially to be able to define the fixed basis function set. This is not necessary for general predefined fixed basis functions, where sensors need only information about their own position to compute $\boldsymbol{\phi}_k$.

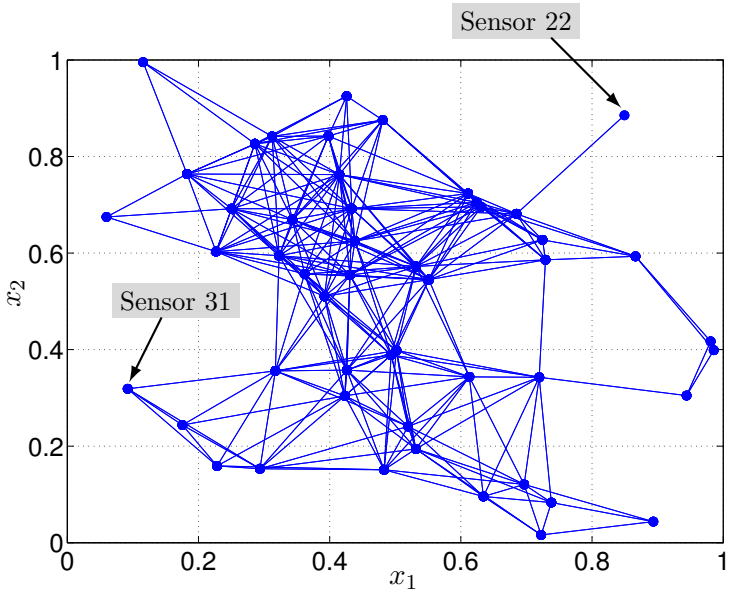


Figure 4.3: Sensor network with 50 randomly deployed sensors.

and b were both defined as zero, we use a kernel parameter of $\theta_\kappa = 15$ and initialize the hyperparameters to $\hat{\alpha}_{k,m} = 10^{-1}$.

Table 4.1 shows a comparison of V-dSBL and V-SBL, where we compare the mean squared error (MSE), the no. of basis functions (#Bfs) and the no. of variational inference iterations (#varIter) until the algorithms converge. We define the MSE for the variational approximate predictor given in (2.20) for V-SBL as $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left(f(\boldsymbol{x}_{\text{test},i}) - \boldsymbol{\phi}_{\text{test},i}^T \hat{\boldsymbol{\mu}} \right)^2$, where $N_{\text{test}} = 10^4$ is the number of test points $\boldsymbol{x}_{\text{test},i}$, $i = 1, \ldots, N_{\text{test}}$, which we placed on a uniform 100×100 grid on the sensor deployment area and $\boldsymbol{\phi}_{\text{test},i} = [\psi_1(\boldsymbol{x}_{\text{test},i}), \ldots, \psi_M(\boldsymbol{x}_{\text{test},i})]^T$. For V-dSBL, we have multiple MSE_k (one for each sensor), where for its computation we use $\hat{\boldsymbol{\mu}}_k$ instead of $\hat{\boldsymbol{\mu}}$. In this sense, we give average and maximum numbers for MSE and #Bfs over the set of sensors. The algorithms are considered to have converged when the maximum absolute change of all $\hat{\alpha}_{k,m}$ or $\hat{\alpha}_m$, respectively for V-dSBL and V-SBL, over the variational inference iterations is smaller than 10^{-3}. Similar, for the V-dSBL algorithm, we consider Gaussian LBP as converged when the maximum absolute change of all elements of the intermediate marginal parameters $\hat{\boldsymbol{\mu}}_k^{(n)}$ and $\hat{\boldsymbol{\Lambda}}_k^{(n)}$ for all sensors k over the message iterations n to be smaller than 10^{-3}. We denote the final number n, after convergence, as the number of message iterations (#msgIter).

		V-dSBL							**V-SBL**		
θ_{th}	β	MSE [dB] avg.	MSE [dB] max.	#Bfs avg.	#Bfs max.	#msgIter avg.	#msgIter max.	#varIter	MSE [dB]	#Bfs	#varIter
10^4	10^4	-20.72	-20.60	**12**	**12**	25.3	29	22	**-21.04**	15	29524
	10^6	**-22.30**	**-22.30**	**14**	**14**	24.3	27	19	-21.04	15	29524
	10^8	**-22.32**	**-22.32**	**14**	**14**	28.4	31	18	-21.04	15	29524
10^6	10^4	-20.57	-20.44	51	51	25.9	29	25	**-21.09**	**15**	3047768
	10^6	**-22.30**	**-22.30**	**14**	**14**	24.3	27	19	-21.09	15	3047768
	10^8	**-22.32**	**-22.32**	**14**	**14**	28.4	31	18	-21.09	15	3047768
10^8	10^4	-20.57	-20.44	51	51	25.9	29	25	**-21.09**	**15**	307263780
	10^6	**-22.30**	**-22.30**	51	51	24.2	27	29	-21.09	**15**	307263780
	10^8	**-22.32**	**-22.32**	**14**	**14**	28.4	31	18	-21.09	15	307263780

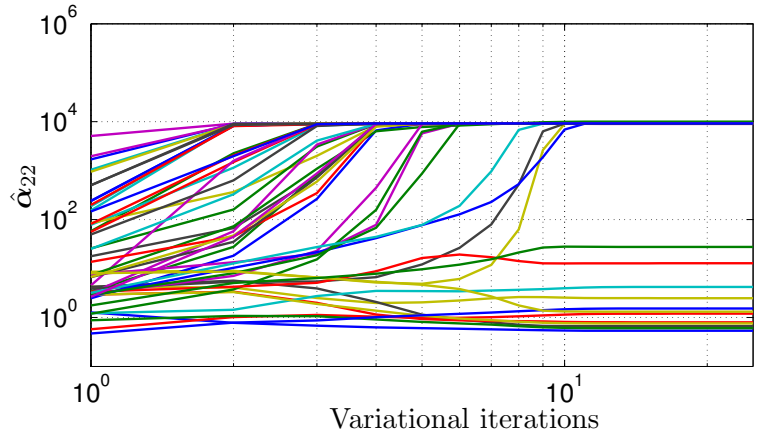
Table 4.1: Synthetic data performance comparison of V-dSBL and V-SBL for the network shown in Figure 4.3. See text for details.

The average and maximum numbers for #msgIter are computed over the variational iterations.

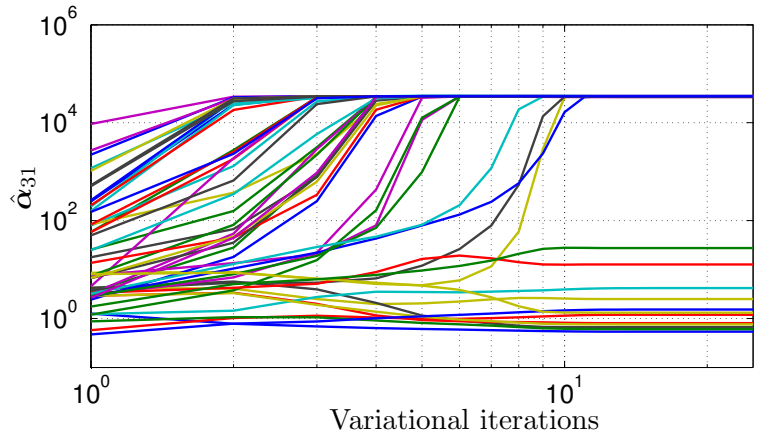
The results in Table 4.1 are given for different pruning thresholds θ_{th} and coupling parameters β. Note that V-SBL only depends on θ_{th} and not on β. Boldface numbers highlight better MSE or sparsity performance. It is very interesting to see that when β is large compared to θ_{th}, V-dSBL performes similar to V-SBL but requires much less variational inference iterations. We note that centralized V-SBL is known for its slow convergence as, e.g., compared to MML* (cf. to Section 2.2.2), but this does not hold for V-dSBL. During all simulations, no convergence problems occurred in V-dSBL. As we further see in Table 4.1, the *max.* and *avg.* values of MSE and #Bfs are mostly equivalent in V-dSBL, which means that the network has achieved consensus. When θ_{th} is large compared to some fixed β, no pruning happens and all V-dSBL results are equivalent. This is because all elements of $\hat{\boldsymbol{\alpha}}_k$ quickly converge to finite values lower than θ_{th}. As we empirically observed in our simulations, the largest values of $\hat{\boldsymbol{\alpha}}_k$ after convergence settle at a level around $\beta\#N(k)$, where in V-SBL they diverge towards infinity. Later in Section 4.4.3 we try to give some explanation on why this is happening. Figure 4.4 compares two examples of the hyperparameter evolution of Sensor 22 and Sensor 31 with centralized V-SBL, whith settings $\theta_{\text{th}} = 10^6$ and $\beta = 10^4$. Note that Sensor 22 and 31 are also highlighted in Figure 4.3, where we see that $\#N(22) = 1$ and $\#N(31) = 4$. Thus, the largest hyperparameters $\hat{\boldsymbol{\alpha}}_k$ converge at a level around β on Sensor 22 and 4β on Sensor 31 as can be seen in Figure 4.4(a) and 4.4(b), respectively.

4.4.2 Real data

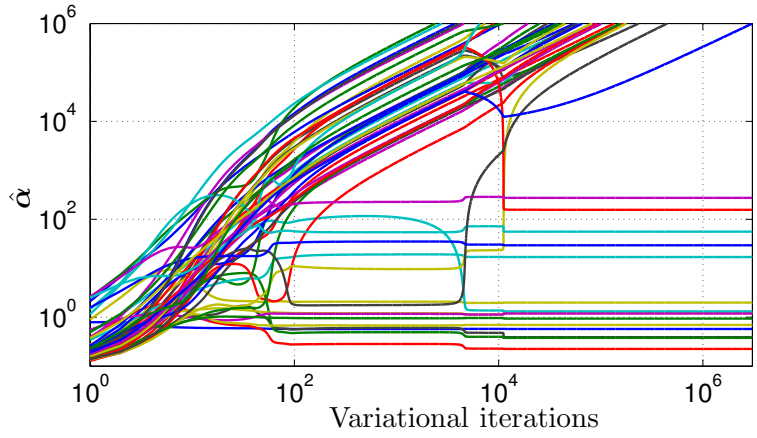
For the real data experiments on V-dSBL, we used atmospheric pressure data over Europe obtained from [2]. The data is depicted in Figure 4.5, where we also show



(a) V-dSBL, Sensor 22, $\beta = 10^4$, $\theta_{\text{th}} = 10^6$, $\#N(22) = 1$

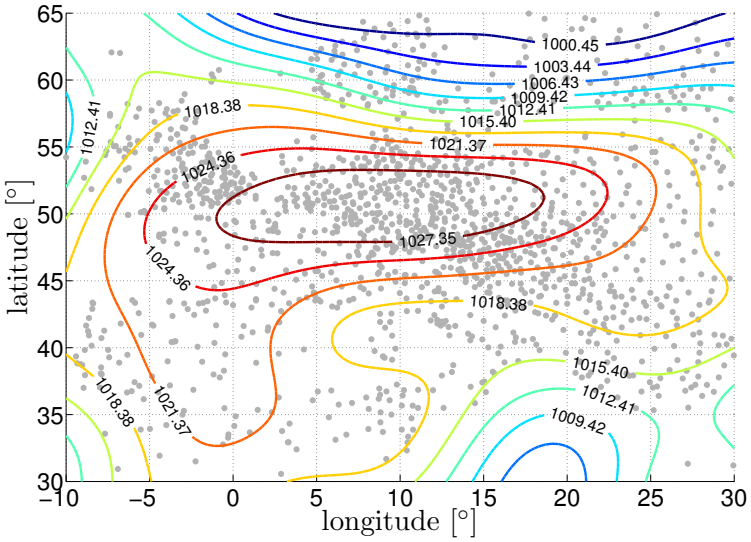


(b) V-dSBL, Sensor 31, $\beta = 10^4$, $\theta_{\text{th}} = 10^6$, $\#N(31) = 4$

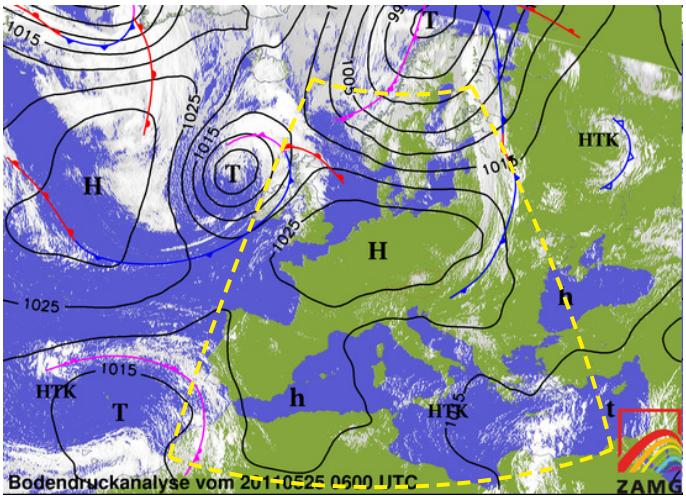


(c) V-SBL, $\theta_{\text{th}} = 10^6$

Figure 4.4: Hyperparameter evolution over the variational iterations.



(a)



(b)

Figure 4.5: Atmospheric pressure over Europe (reduced to sea level) on May 25, 2011, 06:00 UTC. The sensor data was obtained from [2]. (a) Gray dots depict the locations of the 1466 weather stations from the dataset. The contours were generated using centralized MML* kernel regression to qualitatively depict the pressure field. The contour levels are given in hectopascal (hPa). (b) Atmospheric pressure at the same day and time obtained from the Austrian Central Institution for Meteorology and Geodynamics (ZAMG) [3] for comparison. The yellow dashed region approximately highlights the area depicted in (a).

			V-dSBL							V-SBL		
θ_{th}	β	NT	MSE [dB] avg.	MSE [dB] max.	#Bfs avg.	#Bfs max.	#msgIter avg.	#msgIter max.	#varIter	MSE [dB]	#Bfs	#varIter
10^4	10^6	(a)	14.10	14.10	**8**	**8**	291.2	377	23	**2.40**	25	7413170
.	.	(b)	8.10	8.14	**8**	**8**	464.6	688	25	**2.40**	25	7413170
⋮	⋮	(c)	9.73	9.74	**8**	**8**	944.8	1375	14	**2.40**	25	7413170
10^6	10^8	(a)	14.74	14.75	**9**	**9**	276.8	375	22	**2.40**	25	741274899
.	.	(b)	6.51	6.52	**10**	**10**	466.7	692	24	**2.40**	25	741274899
⋮	⋮	(c)	8.97	8.98	**9**	**9**	956.3	1383	40	**2.40**	25	741274899

Table 4.2: Real data performance comparison of V-dSBL and V-SBL for the three networks (NTs) depicted in Figure 4.6(a), (b) and (c). See text for details.

the pressure field obtained from the Austrian Central Institution for Meteorology and Geodynamics (ZAMG) website [3] for comparison. The pressure is reduced to sea level and stems from May 25, 2011, 06:00 UTC. The total number of sensor measurements is 1466. However, for training we only use 293 measurements for computational reasons and hold back the remaining for testing. The training measurements are considered as the sensor measurements in our network, i.e., we have $K = 293$. Each sensor's position is given by the longitude and latitude coordinates of the weather stations from the training set.

We artificially design three sensor networks by connecting the sensors that are close to each other using different numbers of connections. The networks are depicted in Figure 4.6.

We use a kernel parameter $\theta_\kappa = 10^{-2}$ and a noise precision of $\tau = 1$. Table 4.2 shows a performance comparison of V-dSBL and V-SBL for the three networks from Figure 4.6. Table 4.2 can be interpreted the same way as Table 4.1 (see Section 4.4.1 for details). There is only one additional column, titled *NT*, in Table 4.2 which indicates the network topology as shown in Figure 4.6. In all cases β was chosen larger than θ_{th}, which guarantees that the pruning threshold can be reached for all $\hat{\alpha}_m$ that correspond to irrelevant basis functions as we discussed in Section 4.4.1. All other V-dSBL settings, like, e.g., the convergence detection, are equivalent as in Section 4.4.1.

From Table 4.2, we clearly see that V-dSBL performs different for the particular network topologies, even though the sensor positions and measurements are equivalent in all three cases. As the boldface numbers indicate, the centralized V-SBL algorithm performs much better in terms of MSE compared to V-dSBL. However, in all cases V-dSBL leads to sparser results. The required number of variational inference iterations of V-dSBL is again in all cases much lower than for centralized V-SBL. The two settings of the pruning threshold θ_{th} do not influence the performance of centralized V-SBL. But, as the hyperparameters for irrelevant ba-

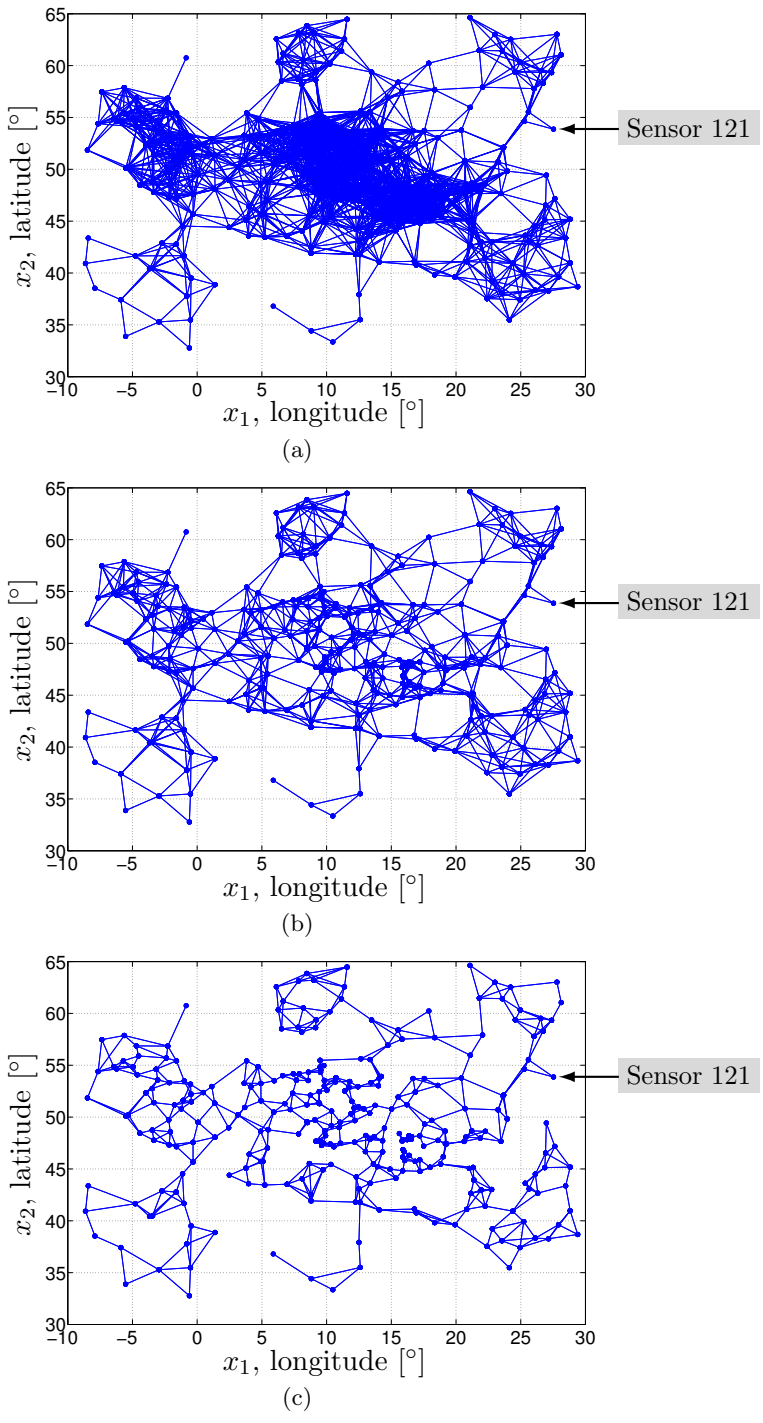
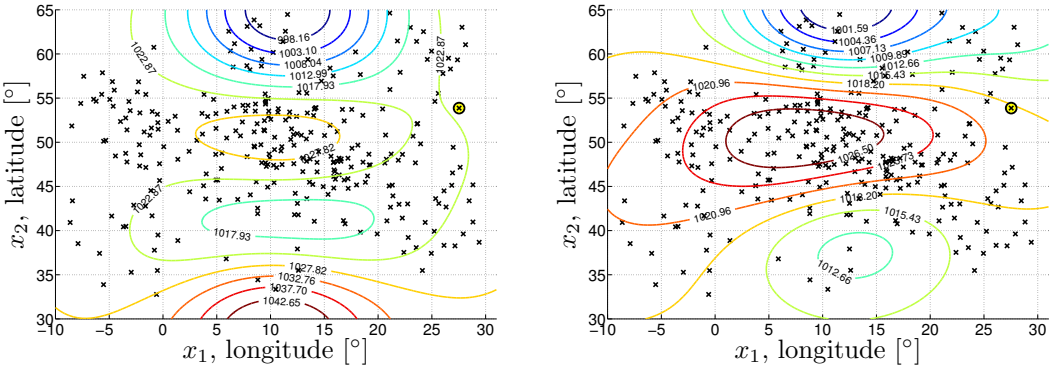


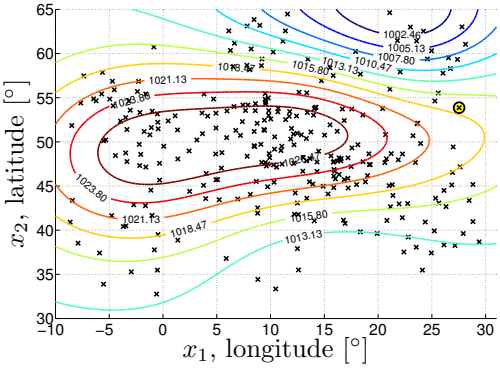
Figure 4.6: Three sensor networks with equivalent sensor locations but different connectivities. The maximum number of neighbors in the networks (a), (b), (c) are 51, 10, 5, respectively.

sis functions diverge quite linearly (cf. Figure 4.4(c)), the number of variational inference iterations is about 100 times larger due to the change of θ_{th} by a factor of 100.

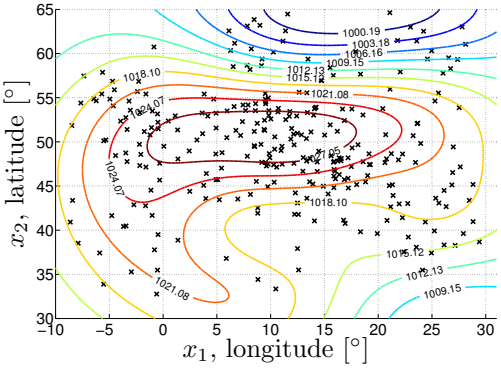


(a) V-dSBL, Sensor 121, Network Fig 4.6(a)

(b) V-dSBL, Sensor 121, Network Fig 4.6(b)



(c) V-dSBL, Sensor 121, Network Fig 4.6(c)



(d) Centralized V-SBL

Figure 4.7: Comparison of the estimated V-dSBL pressure fields (a)–(c) of the networks given in Figure 4.6(a)–4.6(c), respectively, with (d) centralized V-SBL. The crosses indicate the sensor positions. We use the settings $\theta_{\text{th}} = 10^4$ and $\beta = 10^6$. The V-dSBL contours are given for one particular sensor with index 121. Its position is highlighted in the plots (a)–(c) and also in Figure 4.6.

We exemplarily show the contours of the estimated pressure field for all three networks based on the estimate of one particular sensor with index 121 in Figures 4.7(a)–4.7(c). The position of Sensor 121 was already highlighted in Figure 4.6. We used the settings $\theta_{\text{th}} = 10^4$ and $\beta = 10^6$. For comparison, we also show the estimated pressure field of centralized V-SBL in Figure 4.7(d). As we know from Table 4.2, the networks in Fig 4.6(b) and 4.6(c) lead to better predictions, i.e, smaller MSE. This can be also clearly seen by comparing the contour plots of Figure 4.7(b)

and 4.7(c) with Figure 4.5. Note that for the real data experiments, the prediction of centralized V-SBL, also shown in Figure 4.7(d), obtains the smallest MSE.

4.4.3 Discussion of simulation results

As we mentioned in Section 4.4.1, we observed that the largest elements $\hat{\alpha}_{k,m}$ of $\hat{\boldsymbol{\alpha}}_k$ that correspond to irrelevant basis functions do not diverge towards infintiy as in V-SBL, but rather convergence at a level around $\beta\#N(k)$. Unfortunately, due to LBP, it is very difficult to analyze in detail why this is happening. However, intuitively, one may argue that because of the coupling factors f_{kl}, we add some extra noise to the messages between the sensors, i.e., we somehow reduce the trust in the beliefs of the direct neighbors. The variance of this noise term is inversely proportional to the coupling parameter β. Thus, the marginal variance estimate $\hat{\boldsymbol{\Sigma}}_k$, coarsely speaking, collects the incoming noise terms via $\hat{\boldsymbol{\Lambda}}_{uk}$. The more neighbors, the more of these terms are added together as we see in the $\hat{\boldsymbol{\Sigma}}_k$ update in Algorithm 4.1. This could be seen as an attempt to explain the dependency on the number of neighbors, $\#N(k)$. Since $\hat{\alpha}_{k,m}$ is inversely proportional to $\hat{\Sigma}_{k,m}$ (see Algorithm 4.1) and $\hat{\Sigma}_{k,mm}$ has some sort of noise floor, i.e., $\hat{\Sigma}_{k,mm} > 0$, due to the noise corrupted incoming messages, $\hat{\alpha}_{k,m}$ cannot obtain larger values than $\hat{\Sigma}^{-1}_{k,mm}$. That is, $\hat{\alpha}_{k,m}$ is upper bounded due to the noise created by the coupling factors. Note that this is only an intuitive explaination of why the hyperparameters do not diverge towards infinity. To clarify this in detail, much more investigations are needed. However, as we mentioned above, this is a very complicated task due to LBP. For practical situations on the other hand, we showed that by setting the coupling parameter β larger than the pruning threshold θ_{th}, we can find $\alpha_{k,m} > \theta_{\text{th}}$, i.e., we can detect irrelevant basis functions. Especially for the synthetic data experiments, in such situations, the performance of V-dSBL was similar to centralized V-SBL.

4.5 Conclusions and open issues

In this chapter we have presented a variational distributed sparse Bayesian learning (V-dSBL) algorithm for sensor networks based on the idea of consensus propagation (CP) [67]. When the network graph is a tree and the coupling parameter $\beta \to \infty$, we have shown that V-dSBL is equivalent to centralized variational sparse Bayesian learning (V-SBL), where all data is available at one central location. Based on this equivalence, we have extended the algorithm for arbitrary networks with loops, which leads to an iterative Gaussian loopy belief propagation (LBP) sub-

routine. In this case, the coupling parameter β becomes crucial to trade-off LBP's convergence speed vs. its accuracy as described for consensus propagation (CP) in [67]. Performance comparison of V-dSBL and V-SBL on synthetic data showed, that for large β, V-dSBL performs very similar to V-SBL. This was however achieved at a much faster convergence rate over the variational inference iterations. This convergence speed discrepancy even increases for higher pruning thresholds. Unfortunately, for the real data experiments, the performance of centralized V-SBL was better, even though V-dSBL always led to sparser models.

As widely known, the convergence of LBP is not guaranteed in general. However, for Gaussian LBP, there exist convergence criteria [107, 106]. Unfortunately, these criteria are not applicable for V-dSBL as we explicitly pointed out in Section 4.3.2 and Appendix C. Thus, even though we have never observed any convergence problems during all of our simulations, there is no theoretical proof, that Gaussian LBP converges. More specifically, as far as we know, there exists no convergence criteria for Gaussian LBP that is generalized to multivariate nodes and messages. We see this as a very interesting open research issue which has also not been touched by this thesis.

Throughout this chapter we have assumed τ to be known for simplicity. However, as we know from centralized V-SBL (see Section 2.2.2 and [35]), it is possible to estimate τ during the variational inference updates. Although not done in this thesis, it would be straight forward to also implement the update for the expectation $\hat{\tau}$ given by (2.44) in a distributed way using distributed averaging protocols like [104, 67]. Note that the inverse of $\hat{\tau}$ (i.e., the noise variance estimator) can be interpreted as a distributed average of squared errors across the network.

Chapter 5

Fast Variational Distributed Sparse Bayesian Learning

In Chapter 4 we have shown that based on the idea of consensus propagation (CP), a distributed V-SBL (V-dSBL) algorithm can be derived. This method requires the distributed estimation of the SBL weight posterior mean vector and covariance matrix using Gaussian loopy belief propagation (LBP). For elementary Gaussian LBP, as we discuss in Appendix C, it is known [107, 106] that the marginal variance estimates do not converge to the true marginal variances in general, even if convergence is guaranteed based on the criteria in [107, 106]. However, the marginal mean in elementary Gaussian LBP always converges to the true mean if convergence is guaranteed [107, 106]. Unfortunately, to our knowledge, such criteria do not exist for multivariate Gaussian LBP as we proposed for V-dSBL in Chapter 4. Thus, we are not able to guarantee the convergence of Gaussian LBP in V-SBL.

In this chapter, we investigate on fast variational distributed SBL (FV-dSBL), a distributed adaptive version of FV-SBL, which we discussed in Chapter 3. The method allows us to use any kind of distributed averaging protocol, e.g., [104, 67], to compute the FV-SBL decision to determine whether to keep or prune a basis functions. Additionally, to be computationally more efficient, the method further allows us to add new basis functions based on an extension of the *keeping* or *pruning* criteria given in Theorem 1, Chapter 3. We further denote this extension as *adaptive FV-SBL* and will first explain it from a centralized perspective in Section 5.1 (also cf. [117]). Note that in Appendix D we also show a sliding-window online learning method for *adaptive FV-SBL* based on our publication [73]. The obvious advantage of using distributed averaging protocols for the distributed implementation of this *adaptive FV-SBL*, is that no problematic variance estimators as in V-dSBL are required. For instance, if *Consensus Propagation* (CP) [67] (see also Section 2.3) is used for averaging, we only need the Gaussian LBP mean, which has guaranteed

convergence properties for any initialization and any coupling parameter $\beta > 0$ [67]. However, other distributed averaging protocols may be used as well. The performance of FV-dSBL, as compared to centralized *FV-SBL*, only depends on the quality of the distributed average estimates. The FV-dSBL algorithm will be explained in Section 5.2.

We finally note that the methods derived in this chapter should be only seen as a rather straight forward distributed implementation of FV-SBL. It only shows one particular choice of how to collaboratively estimate the relevant basis components, where one may also think of many other alternative approaches in this direction.

5.1 Incorporating new basis functions in fast variational sparse Bayesian learning

The FV-SBL stationary point (3.9) from Theorem 1, Chapter 3, provides a fast rule for deciding whether to keep or delete a basis function from the model, corresponding to a finite or infinite value $\hat{\alpha}_m^{[\infty]}$, respectively. In the following we will show how to use this mechanism to decide if a new candidate basis function should be added to a given model or not. Note that this method is similar to the adaptive learning approach presented in *fast marginal likelihood maximization* [68].

To test a new candidate basis function $M+1$ for an existing model with M components, we need to compute ς_{M+1} and ρ_{M+1} similarly as in (3.5). Thus, we first need to define

$$\bar{\boldsymbol{\Sigma}}_{M+1} = \begin{bmatrix} \hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathrm{diag}(\hat{\boldsymbol{\alpha}}) & \hat{\tau}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{M+1} \\ \hat{\tau}\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\Phi} & \hat{\tau}\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\varphi}_{M+1} \end{bmatrix}^{-1}, \tag{5.1}$$

which is equivalent to (3.4) with $m = M+1$ if we extend $\boldsymbol{\Phi}$ by a new basis vector $\boldsymbol{\varphi}_{M+1}$ in an additional column at the end. By using the inversion rule for structured matrices [116, 113], (5.1) is equivalent to

$$\bar{\boldsymbol{\Sigma}}_{M+1} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}} + \gamma_{M+1}\hat{\tau}^2\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{M+1}\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}} & -\gamma_{M+1}\hat{\tau}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{M+1} \\ -\gamma_{M+1}\hat{\tau}\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}} & \gamma_{M+1} \end{bmatrix} \tag{5.2}$$

where $\gamma_{M+1} = (\hat{\tau}\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\varphi}_{M+1} - \hat{\tau}^2\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{M+1})^{-1}$. We can now compute the

stationary point $\hat{\alpha}_{M+1}^{[\infty]}$ equivalently to (3.9) as

$$\hat{\alpha}_{M+1}^{[\infty]} = \begin{cases} (\rho_{M+1}^2 - \varsigma_{M+1})^{-1}, & \rho_{M+1}^2 > \varsigma_{M+1} \\ \infty, & \rho_{M+1}^2 \leq \varsigma_{M+1} \end{cases}, \tag{5.3}$$

with

$$\begin{aligned} \varsigma_{M+1} &= \boldsymbol{e}_{M+1}^T \bar{\boldsymbol{\Sigma}}_{M+1} \boldsymbol{e}_{M+1} = \gamma_{M+1} = \\ &= (\hat{\tau} \boldsymbol{\varphi}_{M+1}^T \boldsymbol{\varphi}_{M+1} - \hat{\tau}^2 \boldsymbol{\varphi}_{M+1}^T \boldsymbol{\Phi} \hat{\boldsymbol{\Sigma}} \boldsymbol{\Phi}^T \boldsymbol{\varphi}_{M+1})^{-1} \end{aligned} \tag{5.4}$$

and

$$\begin{aligned} \rho_{M+1} &= \hat{\tau} \boldsymbol{e}_{M+1}^T \bar{\boldsymbol{\Sigma}}_{M+1} [\boldsymbol{\Phi}, \boldsymbol{\varphi}_{M+1}]^T \boldsymbol{t} \\ &= \hat{\tau} \varsigma_{M+1} \boldsymbol{\varphi}_{M+1}^T \boldsymbol{t} - \hat{\tau}^2 \varsigma_{M+1} \boldsymbol{\varphi}_{M+1}^T \boldsymbol{\Phi} \hat{\boldsymbol{\Sigma}} \boldsymbol{\Phi}^T \boldsymbol{t}. \end{aligned} \tag{5.5}$$

For finite stationary points $\hat{\alpha}_{M+1}^{[\infty]} < \infty$, we can efficiently compute the extended covariance matrix, which we denote as $\hat{\boldsymbol{\Sigma}}_{M+1}$, from

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{M+1} &= \begin{bmatrix} \hat{\boldsymbol{\Sigma}} + \lambda_{M+1} \hat{\tau}^2 \hat{\boldsymbol{\Sigma}} \boldsymbol{\Phi}^T \boldsymbol{\varphi}_{M+1} \boldsymbol{\varphi}_{M+1}^T \boldsymbol{\Phi} \hat{\boldsymbol{\Sigma}} & -\lambda_{M+1} \hat{\tau} \hat{\boldsymbol{\Sigma}} \boldsymbol{\Phi}^T \boldsymbol{\varphi}_{M+1} \\ -\lambda_{M+1} \hat{\tau} \boldsymbol{\varphi}_{M+1}^T \boldsymbol{\Phi} \hat{\boldsymbol{\Sigma}} & \lambda_{M+1} \end{bmatrix} \\ &= \bar{\boldsymbol{\Sigma}}_{M+1} - \frac{\bar{\boldsymbol{\Sigma}}_{M+1} \boldsymbol{e}_{M+1} \boldsymbol{e}_{M+1}^T \bar{\boldsymbol{\Sigma}}_{M+1}}{\left(\hat{\alpha}_{M+1}^{[\infty]}\right)^{-1} + \boldsymbol{e}_{M+1}^T \bar{\boldsymbol{\Sigma}}_{M+1} \boldsymbol{e}_{M+1}}, \end{aligned} \tag{5.6}$$

where $\lambda_{M+1} = (\hat{\tau} \boldsymbol{\varphi}_{M+1}^T \boldsymbol{\varphi}_{M+1} + \hat{\alpha}_{M+1}^{[\infty]} - \hat{\tau}^2 \boldsymbol{\varphi}_{M+1}^T \boldsymbol{\Phi} \hat{\boldsymbol{\Sigma}} \boldsymbol{\Phi}^T \boldsymbol{\varphi}_{M+1})^{-1}$.

For $\hat{\alpha}_{M+1}^{[\infty]} = \infty$, there is no need to extend the model by the corresponding basis function. This can be also seen from (5.6), where for the case if we nevertheless would add the basis $M+1$, its corresponding weight would be zero. Note that in this situation $\lambda_{M+1} = 0$ and

$$\hat{\boldsymbol{\Sigma}}_{M+1} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}.$$

That is, there is no need to add the basis, since also the mean for this component would be zero, i.e., $\hat{\boldsymbol{\mu}}_{M+1} = \tau \hat{\boldsymbol{\Sigma}}_{M+1} [\boldsymbol{\Phi}, \boldsymbol{\varphi}_{M+1}]^T \boldsymbol{t} = [\hat{\boldsymbol{\mu}}^T, 0]^T$, and thus it does not influence the model.

Algorithm 5.1 summarizes the main steps for adding or rejecting a new candidate basis function in FV-SBL.

Algorithm 5.1 Testing and adding or rejecting a new basis $\boldsymbol{\varphi}_{M+1}$

Compute ς_{M+1} from (5.4) and ρ_{M+1} from (5.5)
if $\rho_{M+1}^2 > \varsigma_{M+1}$ **then**
  % add new basis
  $\hat{\alpha}_{M+1} = (\rho_{M+1}^2 - \varsigma_{M+1})^{-1}$
  Compute: $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}}_{M+1}$ using (5.6)
  $\boldsymbol{\Phi} = [\boldsymbol{\Phi}, \boldsymbol{\varphi}_{M+1}]$, $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}^T, \hat{\alpha}_{M+1}]^T$, $M = M + 1$
  If needed, update $\hat{\boldsymbol{\mu}} = \hat{\tau}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{t}$
else
  % reject new basis - no action needed
end if

5.2 Distributed processing

5.2.1 Computation of ς_m and ρ_m via distributed averaging

To compute the FV-SBL stationary points of the hyperparameters $\{\hat{\alpha}_m^{[\infty]}\}_{m=1}^M$ for the current basis functions and $\hat{\alpha}_{M+1}^{[\infty]}$ for the new basis function from (3.9) and (5.3), respectively, we need to determine $\{\varsigma_m, \rho_m\}_{m=1}^M$ and $\{\varsigma_{M+1}, \rho_{M+1}\}$. To compute these stationary points, which indicate the relevance of the corresponding basis functions, in a distributed manner, we thus need to distributedly compute $\{\varsigma_m, \rho_m\}_{m=1}^M$ and $\{\varsigma_{M+1}, \rho_{M+1}\}$. For simplicity, in the following we do not distinguish between current and new basis functions in our notation, i.e, we now consider m being in the range $1, \ldots, M+1$ instead of $1, \ldots, M$. This is no problem, since (3.9) and (3.23) are equivalent to (5.3) and $\{(5.4), (5.5)\}$, respectively, by setting m to $M+1$. Note that $\boldsymbol{\Phi}_{\overline{M+1}} = \boldsymbol{\Phi}$ and $\hat{\boldsymbol{\Sigma}}_{\overline{M+1}} = \hat{\boldsymbol{\Sigma}}$. Let us thus rewrite Equation (3.23) over the new range as

$$\begin{aligned} \varsigma_m &= (\tau\boldsymbol{\varphi}_m^T\boldsymbol{\varphi}_m - \tau^2\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}\hat{\boldsymbol{\Sigma}}_{\bar{m}}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\varphi}_m)^{-1} \\ \rho_m &= \tau\varsigma_m\boldsymbol{\varphi}_m^T\boldsymbol{t} - \tau^2\varsigma_m\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}\hat{\boldsymbol{\Sigma}}_{\bar{m}}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{t}. \end{aligned} \quad , \quad m = 1, \ldots, M+1 \qquad (5.7)$$

for our further analysis.

As in V-dSBL from Chapter 4, we assume that each sensor k knows the current set of basis functions $\psi_1(\cdot), \ldots, \psi_M(\cdot)$, its own position $\boldsymbol{x}_k$ and its local measurement t_k. Furthermore, for FV-dSBL we assume that each sensor knows the current weight covariance matrix $\hat{\boldsymbol{\Sigma}}$, the current weight mean vector $\hat{\boldsymbol{\mu}}$ and the current hyperparameter vector $\hat{\boldsymbol{\alpha}}$. Additionally, when a new basis function $\psi_{M+1}(\cdot)$ is tested, each

sensor will be informed about this function[1]. As in Chapter 4, we assume the noise precision τ to be known for simplicity, although its distributed estimation would be straight forward as we discussed in Section 4.5.

The fundamental question in FV-dSBL is how to distribute the computation of (5.7) across the sensor network without using any routing protocols. More specifically, we want to use distributed averaging protocols like CP. Unfortunately, there is no direct way of transforming the computation of ς_m and ρ_m from (5.7) into a distributed average form. However, the inner products, $\boldsymbol{\varphi}_i^T\boldsymbol{\varphi}_m$ and $\boldsymbol{\varphi}_i^T\boldsymbol{t}$ inside (5.7) are distributed sums. That is[2],

$$\boldsymbol{\varphi}_i^T\boldsymbol{\varphi}_m = \sum_{k=1}^{K} \psi_i(\boldsymbol{x}_k)\psi_m(\boldsymbol{x}_k), \quad i \in \{1,\dots,M\} \cup \{m\} \tag{5.8}$$

and

$$\boldsymbol{\varphi}_i^T\boldsymbol{t} = \sum_{k=1}^{K} \psi_i(\boldsymbol{x}_k)t_k, \quad i \in \{1,\dots,M\} \cup \{m\}, \tag{5.9}$$

where all $\psi_i(\boldsymbol{x}_k)$ and t_k are available at sensor k. We note that $\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\varphi}_m = (\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}})^T = [\boldsymbol{\Phi}^T\boldsymbol{\varphi}_m]_{\bar{m}} = \left[[\boldsymbol{\varphi}_1^T\boldsymbol{\varphi}_m, \boldsymbol{\varphi}_2^T\boldsymbol{\varphi}_m, \dots, \boldsymbol{\varphi}_M^T\boldsymbol{\varphi}_m]^T\right]_{\bar{m}}$. If we know the number of sensors K, we can compute the sums in (5.8) and (5.9) by multiplying K times the distributed average, i.e, $K(\boldsymbol{\varphi}_i^T\boldsymbol{\varphi}_m/K) = \boldsymbol{\varphi}_i^T\boldsymbol{\varphi}_m$ and $K(\boldsymbol{\varphi}_i^T\boldsymbol{t}/K) = \boldsymbol{\varphi}_i^T\boldsymbol{t}$. Luckily, as we already pointed out in Section 4.3.1, K can be also easily obtained from consensus protocols like [104, 67]; see [120]. However, in the following we just assume K to be known. By computing (5.8) and (5.9) in a distributed way using consensus protocols, we can now easily compute ς_m and ρ_m from (5.7) at each sensor. Note that $\hat{\boldsymbol{\Sigma}}_{\bar{m}}$ can be computed locally at each sensor from $\hat{\boldsymbol{\Sigma}}$ using the rank-1 update $\hat{\boldsymbol{\Sigma}}_{\bar{m}} = \left[\hat{\boldsymbol{\Sigma}} - \frac{\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_m\boldsymbol{e}_m^T\hat{\boldsymbol{\Sigma}}}{\boldsymbol{e}_m^T\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_m}\right]_{\bar{m}\bar{m}}$; cf. to Section 3.3.1.

5.2.2 Fast variational distributed sparse Bayesian learning algorithm with average consensus

We now show how to implement FV-dSBL using average consensus (AC) [104, 121]. We have implemented AC in the form as described in [120]. We also tried to use CP, but empirical observations show that the averaging errors occurring in AC are much smaller compared to CP for reasonable coupling parameter setting β. Large

[1] Or about the parameterization of a family of the functions, e.g., kernel center and kernel parameter for kernel basis functions.

[2] Note that the element $M+1$ is only included in the set $\{1,\dots,M\} \cup \{m\}$ if $m = M+1$.

averaging errors can lead to wrong results of the overall FV-dSBL algorithm. Note that the averaging errors of CP can be made arbitrarily small for large values of β, but this is due to a dramatic increase in the convergence time.

Let us now present a simple form of AC as presented in [120]. For details we refer to [120, 121, 104] and references therein. Consider a sensor network with distributed variables $\nu_1^{(l)}, \ldots, \nu_K^{(l)}$ initialized with the values that are to be averaged at time $l = 0$. By combining all values to a vector $\boldsymbol{\nu}^{(l)} = [\nu_1^{(l)}, \ldots, \nu_K^{(l)}]^T$, we can distributedly iterate

$$\boldsymbol{\nu}^{(l)} = \boldsymbol{Q}\boldsymbol{\nu}^{(l-1)}, \quad l = 1, 2, \ldots \tag{5.10}$$

until the network converges to the desired average: $\lim_{l\to\infty} \boldsymbol{\nu}^{(l)} = \frac{1}{K}\sum_{k=1}^{K} \nu_k^{(0)}$. The matrix $\boldsymbol{Q}$ is defined as $\boldsymbol{Q} = \boldsymbol{I} - (\gamma_{\mathrm{AC}}/\Delta)\boldsymbol{L}$, where the AC convergence coefficient γ_{AC} must lie in the interval $0 < \gamma_{\mathrm{AC}} < 1$ (larger values γ_{AC} lead to faster convergence), the variable Δ denotes the maximum number of neighbors in the network[3], i.e., $\Delta = \max_k \#N(k)$ and $\boldsymbol{L}$ is the graph Laplacian with elements defined as[4]

$$L_{kl} = \begin{cases} N(k), & k = l \\ -1, & k \neq l, \quad l \in N(k) \\ 0, & \text{otherwise} \end{cases}. \tag{5.11}$$

In FV-dSBL, since we consider sensor networks without any superior node, like a fusion center, we randomly select a *responsible node* that initiates the distributed testing of a new candidate basis function. That means, given an initial model, any arbitrary node suggests to the network a candidate basis function, which has to be tested using (5.3) with ς_{M+1} and ρ_{M+1} computed distributedly. If $\hat{\alpha}_{M+1}^{[\infty]}$ is finite, the new basis function is added to the model, otherwise the basis is rejected and another randomly selected node initiates a new candidate. The main idea is that one node, after a random waiting time, broadcasts a testing request for some particular basis function across the network without any routing protocol by using, e.g., *flooding* or *gossiping* [38]. After that, the network computes the required inner products from (5.8) and (5.9) to determine ς_{M+1} and ρ_{M+1} using AC as described above. To

[3] In [120], Δ is described as the maximum node degree, but since we do not consider self-connections, this is equivalent to the maximum number of neighbors. Note that a maximum in a sensor network can be simply obtained distributedly by propagating the maxima of the current maxima of all neighbors recursively across the network.

[4] Note that in [120], the diagonal elements of the graph Laplacian L_{kl} for $k = l$ are defined as the node degree of sensor k, but since we do not consider self connections, this is equivalent to the number of neighbors.

guarantee that only one node suggests a basis function at a given time, one can, e.g., think of timestamps to allow each node within *time synchronized networks* [108] to decide which one was first. If we consider unsynchronized networks, we can also think of sending a random number along with the basis function test request, such that each node only accepts requests with, e.g., the smallest number. However, these are just basic ideas and there are many different ways of how to randomly suggest a new basis function in a distributed way. In the following we simply assume that at one time only a single node is randomly selected and that data from this particular node can be easily broadcasted to all other nodes via multihop communication. The above example was only meant to show that this is in principle possible.

Another practical issue of using distributed averaging protocols, is that in realistic scenarios, we need to stop the iterations after a finite time and the averaging estimates are, although not totally correct, also slightly different. This could lead to different decisions at different sensors whether to add or reject a new basis function. To avoid such situations, the responsible node simply broadcasts its own final estimates to the others to guarantee that all nodes are in an equivalent state.

Similar to the discussion above, also basis functions that are currently in the model can be tested and then kept or pruned depending on $\hat{\alpha}_m^{[\infty]}$ for $m \leq M$. In both cases we need to compute ς_m and ρ_m from (5.7). The fundamental difference in testing a new versus a current basis function, is that we only need to run AC for testing a new one. This can be achieved, if each sensor locally stores the current inner products between all basis vectors, i.e., $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$, and the inner products between all current basis vectors and the targets, i.e., $\boldsymbol{\Phi}^T\boldsymbol{t}$. Thus, according to (5.7), all information for testing a current basis function is locally available. These inner products, can be easily updated if some basis function m is pruned. Namely, by locally removing the appropriate entries of the matrix $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ and the vector $\boldsymbol{\Phi}^T\boldsymbol{t}$ with $\boldsymbol{\Phi}^T\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\Phi}_{\bar{m}} = [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]_{\bar{m}\bar{m}}$ and $\boldsymbol{\Phi}^T\boldsymbol{t} \leftarrow \boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{t} = [\boldsymbol{\Phi}^T\boldsymbol{t}]_{\bar{m}}$. If a new basis function $M+1$ is added, we locally need to update

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi} \leftarrow \begin{bmatrix} \boldsymbol{\Phi}^T\boldsymbol{\Phi} & \boldsymbol{\Phi}^T\boldsymbol{\varphi}_{M+1} \\ \boldsymbol{\varphi}_{M+1}^T\boldsymbol{\Phi} & \boldsymbol{\varphi}_{M+1}^T\boldsymbol{\varphi}_{M+1} \end{bmatrix} \tag{5.12}$$

and

$$\boldsymbol{\Phi}^T\boldsymbol{t} \leftarrow \begin{bmatrix} \boldsymbol{\Phi}^T\boldsymbol{t} \\ \boldsymbol{\varphi}_{M+1}^T\boldsymbol{t} \end{bmatrix}. \tag{5.13}$$

That is, in this case each sensor only additionally needs to know $\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{M+1}$, $\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\varphi}_{M+1}$

and $\boldsymbol{\varphi}_{M+1}^T \boldsymbol{t}$. Exactly these values need to be broadcasted from the responsible sensor to all others. The prior distributed estimation of these inner products for testing a new basis function only requires $M+2$ parallel AC runs. Thus, the communicational complexity during message passing is $O(M)$, which is smaller than $O(M^2)$, as used in V-dSBL; see Chapter 4.

Algorithm 5.2 summarizes the main steps of FV-dSBL using AC. In our notation, we omit the sensor index k for all distributed variables. These are, although processed in parallel and stored locally at each sensor, anyway equivalent everywhere in the network. This is guaranteed, since we broadcast the estimates from the current responsible sensors to all others as discussed above.

Finally, we use a simple convergence criteria. Namely, the algorithm stops if the number of consecutive basis function rejects R_c is larger than $R_{c,\max}$. Thus R_c is increased if a basis function is rejected and reset to zero if accepted. Note that the responsible sensor can simply broadcast the current state of R_c, so the next responsible sensor can use it for further processing.

5.3 Simulations

In this section we apply the FV-dSBL algorithm on the same datasets as in Chapter 4, i.e, the synthetic *Sinc* dataset and the real atmospheric pressure dataset. A detailed description of the datasets can be found in Section 4.4. We compare FV-dSBL to an adaptive version of centralized FV-SBL, i.e., FV-SBL where basis functions can be added to the the model in the same manner as we discussed for FV-dSBL in Algorithm 5.2. Basically, centralized adaptive FV-SBL can be seen as FV-dSBL without AC, where the correct inner products are directly computed. Note that all data is assumed to be centrally available and the basis functions are tested in the same way as in in Algorithm 5.2 for FV-dSBL.

For both, the synthetic and real data experiments, like in Section 4.4, we use a constant bias and kernel basis functions that are centered at the sensor positions. More specifically, we first initialize the model with the bias function $\psi_1(\boldsymbol{x}) = 1$ and set $\hat{\alpha}_1 = 0$. This leads to $\boldsymbol{\varphi}_1 = [1, \ldots, 1]^T$ and $\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{\varphi}_1^T\boldsymbol{\varphi}_1 = K$, which does not depend on the sensor position and can be initialized everywhere without AC (assuming we already know K). However, for the inner product $\boldsymbol{\Phi}^T\boldsymbol{t} = \boldsymbol{\varphi}_1^T\boldsymbol{t}$ we need AC and thus a responsible sensor as described in Algorithm 5.2. Afterwards, each responsible sensor k suggests a candidate basis function which we define as the Gaussian kernel centered at its own position, i.e., $\psi_{M+1}(\boldsymbol{x}) = \kappa(\boldsymbol{x}, \boldsymbol{x}_k) = \exp\{-\theta_\kappa \|\boldsymbol{x} - \boldsymbol{x}_k\|^2\}$. If the

Algorithm 5.2 Fast Variational distributed SBL (FV-dSBL)

% Start with some basis function $\psi_1(\cdot)$
Initialize: $\hat{\boldsymbol{\alpha}} = \hat{\alpha}_1$, $M = 1$, $R_c = 0$
Compute: K, the number of sensors, using AC; cf. Section 4.3.1
• Select a responsible sensor at random and compute the inner products:
$\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{\varphi}_1^T\boldsymbol{\varphi}_1$ and $\boldsymbol{\Phi}^T\boldsymbol{t} = \boldsymbol{\varphi}_1^T\boldsymbol{t}$ using AC
• Broadcast and use the responsible sensor's estimates everywhere
At each sensor, compute: $\boldsymbol{\Sigma} = (\tau\boldsymbol{\varphi}_1^T\boldsymbol{\varphi}_1 + \hat{\alpha}_1)^{-1}$
while ($R_c \leq R_{c,\max}$) **do**
% Test a new basis function
• Randomly select a responsible sensor which then suggests a new basis $M+1$
Compute: $\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{M+1}$, $\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\varphi}_{M+1}$ and $\boldsymbol{\varphi}_{M+1}^T\boldsymbol{t}$ using AC
% Only at the responsible sensor:
Compute: ς_{M+1} and ρ_{M+1} from (5.7)
if $\rho_{M+1}^2 > \varsigma_{M+1}$ **then**
% Add new basis function everywhere
• Broadcast and use the responsible sensor's estimates for $\boldsymbol{\Phi}^T\boldsymbol{\varphi}_{M+1}$, $\boldsymbol{\varphi}_{M+1}^T\boldsymbol{\varphi}_{M+1}$ and $\boldsymbol{\varphi}_{M+1}^T\boldsymbol{t}$ everywhere
Compute: ς_{M+1} and ρ_{M+1} from (5.7) everywhere
Update: $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}^T, \hat{\alpha}_{M+1}^{[\infty]}]^T$, $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}^T\boldsymbol{t}$ from (5.3), (5.12) and (5.13)
Update: $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}}_{M+1}$ from (5.6), $M = M+1$, $R_c = 0$
% Test current basis functions
• For all desired basis functions $m \leq M$, repeat the following everywhere
% - - - - - begin - - - - -
Compute: ς_m and ρ_m from (5.7)
if $\rho_m^2 > \varsigma_m$ **then**
% Keep basis
$\hat{\alpha}_m = (\rho_m^2 - \varsigma_m)^{-1}$, update $\hat{\boldsymbol{\Sigma}}$ from (3.24)
else
$\hat{\boldsymbol{\Sigma}} = \left[\hat{\boldsymbol{\Sigma}} - \frac{\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_m\boldsymbol{e}_m^T\hat{\boldsymbol{\Sigma}}}{\boldsymbol{e}_m^T\hat{\boldsymbol{\Sigma}}\boldsymbol{e}_m}\right]_{\bar{m}\bar{m}}$, $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}]_{\bar{m}}$, $M = M-1$
end if
% - - - - - end - - - - -
else
% Reject new basis, increase the reject counter
$R_c = R_c + 1$
end if
% If desired, locally compute the mean at each sensor
$\hat{\boldsymbol{\mu}} = \tau\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T\boldsymbol{t}$
end while

		FV-dSBL				**adapt. FV-SBL** (*centralized*)	
γ_{AC}	ξ_{AC}	MSE [dB]	#Bfs	#msgIter max.	#msgIter avg.	MSE [dB]	#Bfs
0.9	10^{-6}	**−20.85**	**17**	330	292.79	−20.61	18
⋮	10^{-8}	**−20.61**	**18**	478	441.08	**−20.61**	**18**
	10^{-10}	**−20.61**	**18**	626	589.04	**−20.61**	**18**
	10^{-12}	**−20.61**	**18**	774	737.15	**−20.61**	**18**
0.99	10^{-6}	**−20.80**	**18**	302	268.45	−20.61	**18**
⋮	10^{-8}	**−20.61**	**18**	436	403.35	**−20.61**	**18**
	10^{-10}	**−20.61**	**18**	571	537.67	**−20.61**	**18**
	10^{-12}	**−20.61**	**18**	705	671.98	**−20.61**	**18**

Table 5.1: Synthetic data performance comparison of FV-dSBL and adaptive FV-SBL for the network shown in Figure 4.3. See text for details.

basis function is rejected, another responsible sensor suggests the kernel centered at its own position. If the basis function gets accepted, each sensor locally tests all current basis functions except for the bias. Note that in our simulations we do not test the bias, i.e., $\hat{\alpha}_1$ will never be updated and will always stay at $\hat{\alpha}_1 = 0$. More specifically, this means that there is no regularization for the bias weight and it will always remain in the model.

The implementation of AC also requires some convergence criteria. In our simulations we consider AC as converged if the maximum absolute change of the intermediate average estimates between the updates (5.10), is smaller than ξ_{AC}, i.e., if $\|\boldsymbol{\nu}^{(l)} - \boldsymbol{\nu}^{(l-1)}\|_\infty < \xi_{\text{AC}}$, where $\|\boldsymbol{u}\|_\infty = \max(|u_1|, \ldots, |u_N|)$ for some $\boldsymbol{u} = [u_1, \ldots, u_N]^T$. In both experiments, we analyze three different settings for ξ_{AC} and two different settings for the AC convergence coefficient γ_{AC}. Namely, $\xi_{\text{AC}} = \{10^{-6},\ 10^{-8},\ 10^{-10},\ 10^{-12}\}$ and $\gamma_{\text{AC}} = \{.9,\ .99\}$.

Finally, in both experiments, for the overall convergence, we define the maximum number of consecutive basis function rejects as $R_{c,\max} = K$.

5.3.1 Synthetic data

The specific settings for the *Sinc* dataset is basically equivalent to Section 4.4.1. In particular, $\theta_\kappa = 15$ and $\tau = \sigma^{-2} = 10^3$, where σ^2 is the true underlying noise variance; see Section 4.4.1. The comparison to centralized adaptive FV-SBL is shown in Table 5.1, where better or equivalent mean square error (MSE) or number of basis functions (#Bfs) performance is highlighted with boldface numbers. For the number of AC message iterations (#msgIter), we give maximum and average numbers. From Table 5.1 it is clearly visible, that AC needs less iterations if γ_{AC} and/or ξ_{AC} are large. In general, the MSE and #Bfs performance of FV-dSBL and

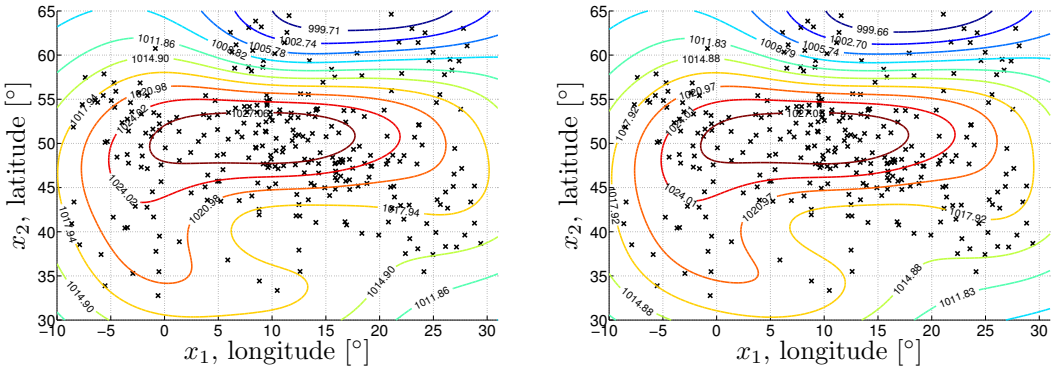
			FV-dSBL				**adapt. FV-SBL** (*centralized*)	
γ_{AC}	ξ_{AC}	NT	MSE [dB]	#Bfs	#msgIter max.	#msgIter avg.	MSE [dB]	#Bfs
0.9	10^{-6}	(a)	2.35	**19**	5956	5372.50	**2.27**	20
⋮	10^{-8}	⋮	**2.26**	**20**	7827	7210.55	2.27	**20**
	10^{-10}		2.37	**20**	9705	9060.61	**2.27**	**20**
	10^{-12}		2.43	**18**	11576	10931.40	**2.27**	20
	10^{-6}	(b)	2.31	**18**	2782	2601.65	**2.27**	20
⋮	10^{-8}	⋮	2.37	**18**	3640	3459.90	**2.27**	20
	10^{-10}		2.28	**20**	4499	4319.94	**2.27**	**20**
	10^{-12}		2.41	**17**	5358	5177.51	**2.27**	20
	10^{-6}	(c)	2.48	**19**	7809	7295.99	**2.27**	20
⋮	10^{-8}	⋮	2.35	**20**	10409	9878.53	**2.27**	**20**
	10^{-10}		2.36	**19**	13010	12482.11	**2.27**	20
	10^{-12}		2.33	**17**	15617	15078.77	**2.27**	20
0.99	10^{-6}	(a)	2.46	**18**	5449	4921.59	**2.27**	20
⋮	10^{-8}	⋮	2.28	**20**	7150	6590.49	**2.27**	**20**
	10^{-10}		2.43	**16**	8857	8280.57	**2.27**	20
	10^{-12}		2.35	**20**	10558	9979.81	**2.27**	**20**
	10^{-6}	(b)	2.32	**17**	2544	2385.26	**2.27**	20
⋮	10^{-8}	⋮	2.28	**20**	3325	3162.22	**2.27**	**20**
	10^{-10}		2.35	21	4105	3941.80	**2.27**	**20**
	10^{-12}		2.41	**18**	4885	4722.54	**2.27**	20
	10^{-6}	(c)	–	–	–	–	2.27	20
⋮	10^{-8}	⋮	2.51	**20**	9511	9043.84	**2.27**	**20**
	10^{-10}		2.35	**18**	11875	11400.45	**2.27**	20
	10^{-12}		2.35	**19**	14243	13768.78	**2.27**	20

Table 5.2: Real data performance comparison of FV-dSBL and adaptive FV-SBL for the three networks (NTs) depicted in Figure 4.6(a), (b) and (c). For $\gamma_{\text{AC}} = .99$ and $\xi_{\text{AC}} = 10^{-6}$, the algorithm did not converge. See text for details.

adaptive FV-SBL is similar and in most cases equivalent.

5.3.2 Real data

Equivalently to Section 4.4.2, for the real atmospheric pressure experiment, we have $\theta_\kappa = 10^{-2}$ and $\tau = 1$. The results for the three different networks shown in Figure 4.6 are presented in Table 5.2. The three networks only differ in the number of connections (see Figure 4.6). In most cases the algorithm converged and obtained similar results as its centralized counterpart. Note that since the centralized adaptive FV-SBL considers the knowledge of all sensor measurements, it is independent of the network structure. This explains why the adaptive FV-dSBL performance is are always the same. In one situation, the algorithm did not converge. Namely for the network shown in Figure 4.6(c) when $\gamma_{\text{AC}} = .99$ and $\xi_{\text{AC}} = 10^{-6}$. According do other experiments, not shown here, such situations can happen when ξ_{AC} is too large. If this is the case, the error of the AC estimator leads to poor inner prod-



(a) FV-dSBL, $\gamma_{\text{AC}} = .9$, $\xi_{\text{AC}} = 10^{-8}$, Network Fig 4.6(b)

(b) adaptive FV-SBL (*centralized*)

Figure 5.1: Contour plots of (a) the prediction of one particular setting of FV-dSBL and (b) the centralized adaptive FV-SBL.

uct estimators, which prevents the algorithm from selecting a proper set of basis functions.

Figure 5.1 shows an example of the FV-dSBL prediction compared to centralized adaptive FV-SBL.

5.4 Conclusions and discussion

In this chapter we discussed an efficient adaptive and distributed implementation of FV-SBL, which we denote as FV-dSBL. We first showed how to adaptively add and delete arbitrary basis functions within the FV-SBL framework. This methodology, which we denote as *adaptive FV-SBL*, offers an interesting and efficient alternative to the other centralized SBL methods. Due to the fact that basis function are only added if required, the algorithm is more efficient especially at the initial phase. Note that all previously discussed SBL methods start from a full model where basis functions can be only deleted. In Appendix D, we present a sliding-window online learning algorithm based on *adaptive FV-SBL*.

The difference of distributed FV-dSBL as compared to *adaptive FV-SBL* is that parts of the computation of the basis function test parameters have to be done distributedly. More specifically, we need to compute inner products that are basically distributed sums of values across the network. As we have shown, this sums can be simply computed using distributed averaging protocols. Furthermore, as we have shown, the distributed computation of the inner products is only required when new

basis functions are tested. For current basis functions, the test parameters can be obtained locally at each sensor.

When we compare FV-dSBL to V-dSBL from Chapter 4, FV-dSBL has several advantages. First, since we directly compute the FV-SBL stationary points , we do not need to iterate the hyperparameters until a threshold for pruning is reached. Thus, we do not even require such a threshold. Secondly, since we can use any distributed averaging protocol, convergence is guaranteed as long as the averaging protocol converges to the true average. Note that we were not able to guarantee the convergence of multivariate Gaussian loopy belief propagation (LBP) as used in V-dSBL. Finally, the communicational complexity of FV-dSBL is $O(M)$ during averaging, whereas for V-dSBL it is $O(M^2)$ for LBP. The computational complexity of FV-dSBL is $O(M^2)$ at each iteration, whereas $O(M^3)$ in V-dSBL for each message.

In our synthetic and real data experiments we used *average consensus* (AC) as distributed averaging protocol. From the results we see that the overall performance of FV-dSBL strongly depends on the quality of the average estimates. In nearly all cased the performance of FV-dSBL was very similar to centralized *adaptive FV-SBL*. However, for inaccurate averaging estimates, as for one particular parameter setting of the real data experiments, the algorithm did not convergence.

Chapter 6

Conclusions

6.1 Summary and discussion

Throughout this thesis we have discussed and analyzed several extentions of variational sparse Bayesian learning (V-SBL).

In Chapter 2, we introduced variational Bayesian (VB) inference and discussed different sparse Bayesian learning (SBL) approximation techniques used in the literature. We have shown that all methods lead to the same update expressions under slightly different model assumptions. As we pointed out extensively, the VB approximation is the only method that can be consistently derived from the desired scale-invariant model assumption, i.e., the model assumption that makes the predictions invariant of the unit of scale of the data and/or the basis functions [1]. We discussed the particular model assumptions of the other methods and clarified some conceptual inconsistencies involved in one of the methods that is also derived from the scale-invariant model assumption. A fundamental difference between the VB approximation and the other discussed methods is that VB approximates the involved probability density functions (pdfs) with other approximating pdfs rather then point estimates, which the other discussed methods do.

Based on the conceptual advantages of the VB approximation to SBL discussed in Chapter 2, we derived a fast V-SBL (FV-SBL) method analogous to *fast marginal likelihood maximization* (FMLM) [68] in Chapter 3. We showed, that the resulting pruning conditions of FV-SBL are equivalent to FMLM even though our method is consistent with the scale-invariant SBL model assumption, which is not the case for FMLM. The slightly different structure in which the pruning condition of FV-SBL is given after its derivation, however, led to the insight that it corresponds to the comparison of a basis function's *signal-to-noise ratio* (SNR) with a 0dB threshold. By changing this threshold to larger values, a *trade-off* between *model sparsity* and *prediction accuracy* can be achieved. As we have shown in our experiments, the

sparsity vs. accuracy trade-off behavior strongly depends on wether we have a *model match* or *mismatch*, i.e., if basis functions involved in the data generation process are also included in the set of basis functions during learning or not, respectively. For the *model mismatch* case, which holds for most real world data experiments, we have shown that there is no optimal *trade-off*, i.e., if we increase the sparsity of the model, we reduce the accuracy at the same time and vice versa. However, if we have a *model match*, a clear *trade-off* optimum can be found as we showed in our experiments. We note that this *sparsity vs. accuracy* trade-off extension for FV-SBL has many advantages for practical applications, since the complexity of the SBL model can be intuitively adapted, e.g., to hardware constraints.

In Chapter 4, we proposed a distributed variant of V-SBL that can be implemented in distributed environments like sensor networks. The method is based on a probabilistic model that is specified by a loopy factor graph, which combines the probabilistic models of SBL and consensus propagation (CP), as described in Section 2.3. A combination of VB inference and Gaussian LBP as subroutine is used to learn the model parameters at each sensor. The sensor's measurements are not shared directly, which makes the algorithm also useful for privacy aware learning problems. Each message sent between neighboring sensors in V-dSBL has a communication complexity of $O(M^2)$ and its computation is an $O(M^3)$ operation, which is very inefficient if M, the number of basis functions, is large. This complexities, however, are strongly reduced during runtime when basis functions are pruned from the model, i.e., when M gets smaller. In our synthetic data experiment, the performance of V-dSBL was similar to centralized V-SBL in terms of sparsity and prediction error. However, the required number of VB updates was much smaller for V-dSBL. In the real data experiments, even though V-dSBL again required less VB updates and led to sparser results, the prediction error was larger compared to centralized V-SBL. We are not able to guarantee the convergence of the Gaussian LBP subroutine in V-dSBL even though convergence problems never occurred during our experiments. The reason is that the general Gaussian LBP convergence criteria [107, 106] are not valid for multivariate Gaussian messages like in V-dSBL. Extending these criteria for such cases is a research topic on its own, which we also point out in the following outlook, Section 6.2; also cf. our discussion in Section 4.3.2 and Appendix C. We have shown that if the coupling parameter $\beta \to \infty$ and the network graph is a tree, i.e., LBP turns into the sum-product algorithm, V-dSBL is always equal to centralized V-SBL.

In Chapter 5, we presented a straightforward distributed implementation of FV-

SBL discussed in Chapter 3. We showed how the inner products involved in the computations of the pruning conditions can be obtained within a sensor network using distributed averaging protocols. Furthermore, we showed how new candidate basis functions can be tested and added to the model bases on the same FV-SBL conditions. The computational complexity of FV-dSBL is $O(M^2)$ at each iteration, whereas the communication complexity for each averaging message is $O(M)$. Thus, FV-dSBL is more efficient than V-dSBL. Furthermore, since FV-dSBL is based on the stationary point analysis as in FV-SBL, no hyperparameter threshold has to be specified as opposed to V-dSBL. The comparison of FV-dSBL with centralized adaptive FV-SBL led to a very similar performance in both sparsity and prediction error. However, the algorithm strongly depends on the quality of the averaging estimate, where convergence problems can occur for larger errors as we have seen in one situation.

Both distributed methods, V-dSBL and FV-dSBL, are based on consensus protocols for distributed averaging. This makes them robust against changes in the network topology as long as the sensor network remains connected. Even though consensus protocols are also robust against sensor failures, we have not considered this situation in this thesis, where we give some comments in the following outlook; Section 6.2.

Finally, as we already mentioned in Section 2.2.3, the VB approach to SBL did not receive as much attention in the literature as its counterparts MML and MMP-log (cf. Section 2.2.2), even though, as we have shown in Section 2.2.3 and also mentioned above, it is the most profound method from a Bayesian perspective. That is, it naturally incorporates scale invariance through non-informative Jeffreys priors [1]. We have shown in this thesis that a fast version similar to FMLM and even distributed variants of SBL can be derived from the VB approach. We hope, that based on our results and our clarifications of the conceptual advantages of V-SBL, future reseach will have more focus on the VB approach, which may lead to better insights and further variants of SBL.

6.2 Outlook

In the previous section we have summarized and discussed the major contributions made by this thesis. During our research, however, we found interesting issues, problems and ideas that we would like to list in the following and hope that these may serve as entry points for future investigations.

- In Section 2.2.4, we showed that SBL MAP estimation leads to the problematic optimization problem (2.55), where every solution, with at least one element in $\boldsymbol{w}$ being zero, leads to a global minimum. However, if we consider general gamma hyperpriors $p(\alpha_m)$, i.e., with unspecified a and b, we obtain the following MAP estimation problem

$$\boldsymbol{w}_{\text{MAP}}^{(a,b)} = \underset{\boldsymbol{w}}{\text{argmin}} \ \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{t}\|_2^2 + \frac{2}{\tau}\Big(a + \frac{1}{2}\Big) \sum_{m=1}^{M} \ln\Big(b + \frac{w_m^2}{2}\Big), \tag{6.1}$$

  where for $b > 0$ the singularities, caused by weights w_m being zero through the logarithm in (6.1), vanish. It would be very interesting to analyze how the structure of local minima in (6.1) changes for different settings of a and b. The idea is to investigate if it is possible to converge to useful optima of (2.55) by changing a and b during the optimization process, e.g., using gradient descent, and finally let both parameters converge to $a \to 0$, $b \to 0$. Note that it is straightforward to compute the gradient of (6.1). However, it is not clear how such a method relates to the other methods discussed in this thesis, which all require some form of optimization over the hyperparameters. Note, that the MAP estimation over $\boldsymbol{w}$, as shown in (6.1), directly assumes an effective *Student's-t* weight prior and does not involve any hyperparameters.

- In Appendix D we presented a sliding window online SBL method based on FV-SBL. Due to the sliding window, the algorithm abruptly forgets the past. It would be very interesting to develop a recursive version of FV-SBL that adds and prunes basis functions as data samples arrive online. Such a method would allow to learn from huge datasets, since the samples are processed online and must not be kept in memory. Since SBL can be interpreted as a Gaussian process [1, 56], a related work in online Gaussian process learning is, e.g., [122], which itself is strongly related to the kernel recursive least squares (Kernel-RLS) [74] algorithm that we used for comparison in Appendix D. Note that in Kernel-RLS, kernels are only added but never pruned from the model, which is sometimes also denoted as *constructive sparsity*.

- As we already mentioned above, the convergence of Gaussian LBP in V-dSBL cannot be guaranteed. The reason is that the known convergence criteria for Gaussian LBP in [107, 106] are only valid for scalar messages. Since we have multivariate messages in V-dSBL, the convergence criteria [107, 106] need to be extended for such a case. Note that a multivariate graph like in V-

dSBL can always be dissected into an elementary scalar graph. However, as discussed in Appendix C, the convergence criteria [107, 106] for the elementary dissected graph of V-dSBL were not fulfilled in our test experiments. There is, nonetheless, a clear argument for the development of a convergence criterion for multivariate Gaussian LBP. Namely, for tree networks, when the (multivariate) sum product algorithm can be applied and convergence is always guaranteed, the convergence criteria [107, 106] were still not fulfilled for the elementary dissected graph.

- When Gaussian LBP converges according to [107, 106], the marginal mean estimates are guaranteed to be correct. However, although the convergence of the marginal variance estimators are also guaranteed, their estimates are typically different from the true marginal variances. For most applications, like consensus propagation [67], only the marginal mean estimates are relevant, where the variance estimates are mostly ignored. However, as we have seen for multivariate Gaussian LBP in V-dSBL, also the covariance matrix estimates are of particular interest. It would be very interesting to investigate how the variance (or covariance) estimates can be corrected to coincide with the true marginal variances (or covariances).

- In the distributed algorithms V-dSBL and FV-dSBL, each sensor k was considered to observe a measurement t_k. If a sensor fails, which we did not consider throughout the thesis, also a sample from the dataset gets lost. It would be interesting to investigate how to make even centralized V-SBL or FV-SBL robust against the removal of a sample t_k during runtime. Extending such concepts to V-dSBL and FV-dSBL would make both algorithms robust against sensor failures. Note that both algorithms are already robust against changing network topologies.

Appendix A

Proofs and Derivations

A.1 Priors $p(\tau)$ and $p(\boldsymbol{\alpha})$ for scale-invariant sparse Bayesian learning prediction

In this section we show, based on a single conjecture, that the non-informative priors $p(\boldsymbol{\alpha}) \propto \prod_{m=1}^{M} \frac{1}{\alpha_m}$ and $p(\tau) \propto 1/\tau$ are the only priors that lead to scale invariance in the SBL prediction $p(t^*|\boldsymbol{t})$ if we scale the targets $\boldsymbol{t}$. Note that the pdf $p(t^*|\boldsymbol{t})$ represents the prediction for target t^* given a new input sample $\boldsymbol{x}^*$ and all observations from the training dataset $\mathcal{D} = \{\boldsymbol{x}_n, t_n\}_{n=1}^{N}$. Since we only use random variables in our notation for readability reasons, and thus leave out all deterministic variables like $\boldsymbol{x}_n$ and $\boldsymbol{x}^*$, we, however, simply write the prediction as $p(t^*|\boldsymbol{t})$. The above mentioned non-informative priors correspond to the limit case $a = b = c = d = 0$ of the gamma prior parameterizations, as discussed in Section 2.2.1.

The scale invariance property of the SBL prediction for $a = b = c = d = 0$ was mentioned already in [1], but has not been proven there. Here, in this section, we will formally derive these priors from the scale invariance assumption.

By scaling the targets $\boldsymbol{t}$ by a factor $\nu > 0$, we obtain an SBL prediction $p(t^*|\nu\boldsymbol{t})$. The prediction is invariant under this scaling of $\boldsymbol{t}$ if

$$\boxed{p(t^*|\nu\boldsymbol{t}) = \frac{1}{\nu} p\Big(\frac{1}{\nu} t^* \Big| \boldsymbol{t}\Big)}, \tag{A.1}$$

where $p(\tilde{t}^*|\boldsymbol{t})$ is an unscaled prediction over $\tilde{t}^*$, which we have scaled by setting $t^* = \nu\tilde{t}^*$ using the transformation rule for random variables [91]

$$p_Y(y) = p_X\left(g^{-1}(y)\right) \left| \frac{dg^{-1}(y)}{dy} \right|, \quad \text{for } Y = g(X). \tag{A.2}$$

Note that $g^{-1}(\cdot)$ is the inverse function, which in our case is $\tilde{t}^* = \frac{1}{\nu} t^*$. The scale

invariance in (A.1) means that if we scale the data $\boldsymbol{t}$ by a factor ν we should obtain the same prediction as if we scale (random variable transformation) the unscaled prediction $p(\tilde{t}^*|\boldsymbol{t})$.

Based on the SBL model definition in Section 2.2.1 we can write the left- and right-hand sides in (A.1) as

$$p(t^*|\nu\boldsymbol{t}) = \int p(t^*|\tau, \boldsymbol{w})\; p(\tau, \boldsymbol{w}|\nu\boldsymbol{t})\, d\tau\, d\boldsymbol{w}, \tag{A.3}$$

and

$$\frac{1}{\nu}p\Big(\frac{1}{\nu}t^*\Big|\boldsymbol{t}\Big) = \frac{1}{\nu}\int p\Big(\frac{1}{\nu}t^*\Big|\tilde{\tau}, \tilde{\boldsymbol{w}}\Big)\; p(\tilde{\tau}, \tilde{\boldsymbol{w}}|\boldsymbol{t})\, d\tilde{\tau}\, d\tilde{\boldsymbol{w}}, \tag{A.4}$$

respectively. From the definitions $p(t^*|\tau, \boldsymbol{w}) = \mathcal{N}(t^*|\boldsymbol{w}^T\boldsymbol{\phi}^*, \tau^{-1})$ and $p(\tilde{t}^*|\tilde{\tau}, \tilde{\boldsymbol{w}}) = \mathcal{N}(\tilde{t}^*|\tilde{\boldsymbol{w}}^T\boldsymbol{\phi}^*, \tilde{\tau}^{-1})$ with $\boldsymbol{\phi}^* = [\psi_1(\boldsymbol{x}^*), \ldots, \psi_M(\boldsymbol{x}^*)]^T$ it is easy to see that

$$\begin{aligned}
\frac{1}{\nu}p\Big(\frac{1}{\nu}t^*\Big|\tilde{\tau}, \tilde{\boldsymbol{w}}\Big) &= \frac{1}{\nu}\sqrt{\frac{\tilde{\tau}}{2\pi}}\exp\left\{-\frac{\tilde{\tau}}{2}\left(\frac{1}{\nu}t^* - \tilde{\boldsymbol{w}}^T\boldsymbol{\phi}^*\right)^2\right\} \\
&= \sqrt{\frac{\tilde{\tau}}{2\pi\nu^2}}\exp\left\{-\frac{\tilde{\tau}}{2\nu^2}\left(t^* - \nu\tilde{\boldsymbol{w}}^T\boldsymbol{\phi}^*\right)^2\right\} \\
&= \mathcal{N}\left(t^*\middle|\nu\tilde{\boldsymbol{w}}^T\boldsymbol{\phi}^*, \left(\frac{\tilde{\tau}}{\nu^2}\right)^{-1}\right) \\
&= p\big(t^*\big|\frac{\tilde{\tau}}{\nu^2}, \nu\tilde{\boldsymbol{w}}\big).
\end{aligned} \tag{A.5}$$

Let us now analyze the posterior $p(\tilde{\tau}, \tilde{\boldsymbol{w}}|\boldsymbol{t})$ from (A.4), which is defined as

$$p(\tilde{\tau}, \tilde{\boldsymbol{w}}|\boldsymbol{t}) = \frac{1}{p(\boldsymbol{t})}p(\boldsymbol{t}|\tilde{\tau}, \tilde{\boldsymbol{w}})p(\tilde{\tau})\int p(\tilde{\boldsymbol{w}}|\tilde{\boldsymbol{\alpha}})p(\tilde{\boldsymbol{\alpha}})\; d\tilde{\boldsymbol{\alpha}}. \tag{A.6}$$

By transforming the variables $\tilde{\tau}$ and $\tilde{\boldsymbol{w}}$ to $\tau = \frac{\tilde{\tau}}{\nu^2}$ and $\boldsymbol{w} = \nu\tilde{\boldsymbol{w}}$, the posterior in (A.6) can be written as

$$\nu^{(2-M)}\; p\big(\nu^2\tau, \frac{\boldsymbol{w}}{\nu}\big|\boldsymbol{t}\big), \tag{A.7}$$

which gives

$$\frac{\nu^{(2-M)}}{p(\boldsymbol{t})}p\Big(\boldsymbol{t}\Big|\nu^2\tau, \frac{\boldsymbol{w}}{\nu}\Big)p(\nu^2\tau)\int p\Big(\frac{\boldsymbol{w}}{\nu}\Big|\tilde{\boldsymbol{\alpha}}\Big)p(\tilde{\boldsymbol{\alpha}})\; d\tilde{\boldsymbol{\alpha}}, \tag{A.8}$$

where we have used the transformation rule for multivariate random variables; see [91]. When we apply the same transformation to the parameters of the pdf (A.5) and insert its new form together with (A.7) into (A.4), we obtain the trans-

formed integral over τ and $\boldsymbol{w}$ as

$$\frac{1}{\nu}p\Big(\frac{1}{\nu}t^*\Big|\boldsymbol{t}\Big) = \nu^{(2-M)} \int p(t^*|\tau,\boldsymbol{w})\; p\big(\nu^2\tau, \frac{\boldsymbol{w}}{\nu}\big|\boldsymbol{t}\big)\, d\tau\, d\boldsymbol{w}. \tag{A.9}$$

For scale invariant predictions, i.e., when (A.1) holds, (A.9) needs to be equivalent to (A.3), which obviously is the case for all values of t^* if

$$\boxed{p(\tau,\boldsymbol{w}|\nu\boldsymbol{t}) = \nu^{(2-M)}\; p\big(\nu^2\tau, \frac{\boldsymbol{w}}{\nu}\big|\boldsymbol{t}\big)}, \tag{A.10}$$

Note that all further derivations are based on the **conjecture** that (A.1) **only** holds if (A.10) holds, i.e., we strongly belief that the integrals (A.9) and (A.3) are only equivalent for all values t^*, given the model assumptions from Section 2.2.1, **if and only if** (A.10) holds.

We can rewrite the terms $p\big(\boldsymbol{t}\big|\nu^2\tau, \frac{\boldsymbol{w}}{\nu}\big)$ and $p\big(\frac{\boldsymbol{w}}{\nu}\big|\tilde{\boldsymbol{\alpha}}\big)$ in (A.8) as

$$\begin{aligned}
p\Big(\boldsymbol{t}\Big|\nu^2\tau, \frac{\boldsymbol{w}}{\nu}\Big) &= \mathcal{N}\left(\boldsymbol{t}\Big|\frac{\boldsymbol{w}}{\nu}, (\nu^2\tau)^{-1}\boldsymbol{I}\right) \\
&= \left(\frac{\nu^2\tau}{2\pi}\right)^{\frac{N}{2}} \exp\left\{-\frac{\nu^2\tau}{2}\left(\boldsymbol{t}-\frac{1}{\nu}\boldsymbol{\Phi}\boldsymbol{w}\right)^T\left(\boldsymbol{t}-\frac{1}{\nu}\boldsymbol{\Phi}\boldsymbol{w}\right)\right\} \\
&= \nu^N \left(\frac{\tau}{2\pi}\right)^{\frac{N}{2}} \exp\Big\{-\frac{\tau}{2}\left(\nu\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{w}\right)^T\left(\nu\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{w}\right)\Big\} \\
&= \nu^N\; \mathcal{N}(\,\nu\boldsymbol{t}\,|\,\boldsymbol{\Phi}\boldsymbol{w}, \tau^{-1}\boldsymbol{I}\,) \\
&= \nu^N\; p(\,\nu\boldsymbol{t}\,|\,\tau,\boldsymbol{w}\,)
\end{aligned} \tag{A.11}$$

and

$$\begin{aligned}
p\Big(\frac{\boldsymbol{w}}{\nu}\Big|\tilde{\boldsymbol{\alpha}}\Big) &= \mathcal{N}\left(\frac{\boldsymbol{w}}{\nu}\Big|\boldsymbol{0},\; \mathrm{diag}(\tilde{\boldsymbol{\alpha}})^{-1}\right) \\
&= \left(\frac{\prod_{m=1}^M \tilde{\alpha}_m}{2\pi}\right)^{\frac{M}{2}} \exp\left\{-\frac{1}{2}\left(\frac{\boldsymbol{w}}{\nu}\right)^T \mathrm{diag}(\tilde{\boldsymbol{\alpha}})\left(\frac{\boldsymbol{w}}{\nu}\right)\right\} \\
&= \nu^M \left(\frac{\prod_{m=1}^M \frac{\tilde{\alpha}_m}{\nu^2}}{2\pi}\right)^{\frac{M}{2}} \exp\left\{-\frac{1}{2}\boldsymbol{w}^T\,\mathrm{diag}\left(\frac{\tilde{\boldsymbol{\alpha}}}{\nu^2}\right)\boldsymbol{w}\right\} \\
&= \nu^M \mathcal{N}\left(\boldsymbol{w}\Big|\boldsymbol{0},\; \mathrm{diag}\left(\frac{\tilde{\boldsymbol{\alpha}}}{\nu^2}\right)^{-1}\right) \\
&= \nu^M p\Big(\boldsymbol{w}\Big|\frac{\tilde{\boldsymbol{\alpha}}}{\nu^2}\Big),
\end{aligned} \tag{A.12}$$

where $\tilde{\boldsymbol{\alpha}} = [\tilde{\alpha}_1, \ldots, \tilde{\alpha}_M]^T$. By transforming the hyperparameters $\tilde{\boldsymbol{\alpha}}$ in $p(\tilde{\boldsymbol{\alpha}})$ to

$\boldsymbol{\alpha} = \frac{\tilde{\boldsymbol{\alpha}}}{\nu^2}$, we can write the transformed hyperprior as

$$\nu^{2M} p(\nu^2 \boldsymbol{\alpha}), \tag{A.13}$$

which also changes the integral in (A.8) to

$$\nu^{3M} \int p(\boldsymbol{w}|\boldsymbol{\alpha}) p(\nu^2 \boldsymbol{\alpha}) \, d\boldsymbol{\alpha}, \tag{A.14}$$

where we have also inserted (A.12). We can thus rewrite (A.8), which is equivalent to the right-hand side term of (A.10), as

$$\nu^{(2-M)} \, p\big(\nu^2 \tau, \frac{\boldsymbol{w}}{\nu} \big| \boldsymbol{t}\big) = \frac{\nu^N}{p(\boldsymbol{t})} \, p(\, \nu \boldsymbol{t} \,|\, \tau, \boldsymbol{w} \,) \nu^2 p(\nu^2 \tau) \int p(\boldsymbol{w}|\boldsymbol{\alpha}) \nu^{2M} p(\nu^2 \boldsymbol{\alpha}) \, d\boldsymbol{\alpha} \tag{A.15}$$

where we also used (A.11).

Similar to (A.6), the left-hand side of (A.10) is defined as

$$p(\tau, \boldsymbol{w}|\nu \boldsymbol{t}) = \frac{1}{p(\nu \boldsymbol{t})} p\big(\nu \boldsymbol{t} \big| \tau, \boldsymbol{w}\big) p(\tau) \int p(\boldsymbol{w}\big|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) \, d\boldsymbol{\alpha}. \tag{A.16}$$

Thus, if and only if (A.15) equals (A.16) from (A.10), then (A.1) holds according to our conjecture. This is the case if and only if the following two equalities are satisfied:

$$p(\tau) = \nu^2 p(\nu^2 \tau), \tag{A.17}$$

$$p(\boldsymbol{\alpha}) = \nu^{2M} p(\nu^2 \boldsymbol{\alpha}). \tag{A.18}$$

Note that another dimensionality dependent allocation of the normalization terms $\nu^{(\cdot)}$ makes no sense, since the vectors $\boldsymbol{t}$ and $\boldsymbol{\alpha}$ are N and M dimensional, respectively. Furthermore note that the integrals in (A.15) and (A.16) lead to Student's-t distributions, as we discussed in Section 2.2.1, which are only equivalent if (A.18) holds.

In the following we show that $p(\boldsymbol{t}) = \nu^N p(\nu \boldsymbol{t})$, as we further need for the equivalence of (A.15) and (A.16), if (A.17) and (A.18) hold. The term $\nu^N p(\nu \boldsymbol{t})$ is defined as

$$\nu^N p(\nu \boldsymbol{t}) = \nu^N \int p(\nu \boldsymbol{t}|\tau, \boldsymbol{w}) p(\tau) p(\boldsymbol{w}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) \, d\tau d\boldsymbol{w} d\boldsymbol{\alpha}. \tag{A.19}$$

We can further rewrite $p(\boldsymbol{t})$ as

$$\begin{aligned} p(\boldsymbol{t}) &= \int p(\boldsymbol{t}|\tilde{\tau}, \tilde{\boldsymbol{w}}) p(\tilde{\tau}) p(\tilde{\boldsymbol{w}}|\tilde{\boldsymbol{\alpha}}) p(\tilde{\boldsymbol{\alpha}})\, d\tilde{\tau} d\tilde{\boldsymbol{w}} d\tilde{\boldsymbol{\alpha}} \\ &= \nu^{N+2M+2} \int p(\nu\boldsymbol{t}|\tau, \boldsymbol{w}) p(\nu^2\tau) p(\boldsymbol{w}|\boldsymbol{\alpha}) p(\nu^2\boldsymbol{\alpha})\, d\tau d\boldsymbol{w} d\boldsymbol{\alpha}, \end{aligned} \tag{A.20}$$

where we have transformed the variables $\tilde{\tau}$, $\tilde{\boldsymbol{w}}$, $\tilde{\boldsymbol{\alpha}}$ to $\tau = \frac{\tilde{\tau}}{\nu^2}$, $\boldsymbol{w} = \nu\tilde{\boldsymbol{w}}$, $\boldsymbol{\alpha} = \frac{\tilde{\boldsymbol{\alpha}}}{\nu^2}$ like above and made use of the results (A.11) and (A.12). If we now insert (A.17) and (A.18) into (A.20), we see that (A.20) is equivalent to (A.19).

We would like to stress here that the scale invariance of the SBL model according to (A.1) thus only holds for priors that have the properties (A.17) and (A.18). From the definition (2.17), Section 2.2.1, we see that due to the factorization, (A.18) can be separated to

$$p(\alpha_m) = \nu^2 p(\nu^2\alpha_m), \quad m = 1\ldots, M, \tag{A.21}$$

where $p(\boldsymbol{\alpha}) = \prod_{m=1}^{M} p(\alpha_m)$. Since both, $p(\alpha_m)$ and $p(\tau)$, are gamma priors and have the same constraints (A.17) and (A.21), both obviously have to be of the same form.

Priors with the property $p(\eta) = \nu^2 p(\nu^2\eta)$, as in (A.17) and (A.21), are known as scale invariant priors and are defined as $p(\eta) \propto 1/\eta$ as we wanted to show; cf. [4]. Note that this is easy to see if we insert an arbitrary value, e.g. $\eta^* = 1$ into $p(\eta) = \nu^2 p(\nu^2\eta)$, which gives $p(1) = \nu^2 p(\nu^2)$. It follows that $p(\nu^2) = p(1)/\nu^2$ and thus $p(\eta) = p(1)/\eta \propto 1/\eta$. Such priors are also known to assign equal probability mass to the intervals $A \leq \eta \leq B$ and $\nu^2 A \leq \eta \leq \nu^2 B$, i.e.,

$$\int_A^B p(\eta) d\eta = \int_{\nu^2 A}^{\nu^2 B} p(\eta) d\eta = \nu^2 \int_A^B p(\nu^2\eta) d\eta, \tag{A.22}$$

cf. [4, Section 2.4.3]. Finally note that such priors are improper and flat over a logarithmic scale as we already mentioned in Section 2.2.1.

A.2 Derivation of Equation (2.29) and (2.30)

Let us first derive the result from (2.29). By computing the derivative of the log of the *marginal likelihood* given in (2.28) and equating to zero, we obtain

$$\frac{\partial \ln p(\boldsymbol{t}|\tau, \boldsymbol{\alpha})}{\partial \alpha_m} = \frac{\partial}{\partial \alpha_m} \Big\{ -\frac{N}{2} \ln 2\pi - \frac{1}{2} \ln |\boldsymbol{C}| - \frac{1}{2} \boldsymbol{t}^T \boldsymbol{C}^{-1} \boldsymbol{t} \Big\} = 0, \tag{A.23}$$

where $\boldsymbol{C} = \tau^{-1}\boldsymbol{I} + \boldsymbol{\Phi}\big(\operatorname{diag}(\boldsymbol{\alpha})\big)^{-1}\boldsymbol{\Phi}^T$ and $|\boldsymbol{C}|$ denotes the determinant of $\boldsymbol{C}$. We can transform the term $\ln|\boldsymbol{C}|$ as follows

$$\begin{aligned}
\ln|\boldsymbol{C}| &= \ln|\tau^{-1}\boldsymbol{I} + \boldsymbol{\Phi}\big(\operatorname{diag}(\boldsymbol{\alpha})\big)^{-1}\boldsymbol{\Phi}^T| \\
&= \ln(\tau^{-N}|\boldsymbol{I} + \tau\boldsymbol{\Phi}\operatorname{diag}(\boldsymbol{\alpha})^{-1}\boldsymbol{\Phi}^T|) \\
&= -N\ln\tau + \ln|\boldsymbol{I} + \tau\operatorname{diag}(\boldsymbol{\alpha})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Phi}| \\
&= -N\ln\tau + \ln(|\operatorname{diag}(\boldsymbol{\alpha})^{-1}|\cdot|\operatorname{diag}(\boldsymbol{\alpha}) + \tau\boldsymbol{\Phi}^T\boldsymbol{\Phi}|) \\
&= -N\ln\tau + -\ln|\operatorname{diag}(\boldsymbol{\alpha})| + \ln|\tau\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \operatorname{diag}(\boldsymbol{\alpha})| \\
&= -N\ln\tau + -\sum_{m=1}^{M}\ln\alpha_m - \ln|\boldsymbol{\Sigma}|, \qquad \text{(A.24)}
\end{aligned}$$

where we have used the matrix identities $|\boldsymbol{B}^{-1}| = |\boldsymbol{B}|^{-1}$, $|c\boldsymbol{B}| = c^N|\boldsymbol{B}|$ for any $N\times N$ matrix $\boldsymbol{B}$ and $|\boldsymbol{I} + \boldsymbol{Q}^T\boldsymbol{R}| = |\boldsymbol{I} + \boldsymbol{R}^T\boldsymbol{Q}|$ for $M\times N$ matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, cf. [4, 116], and $\boldsymbol{\Sigma} = \big(\tau\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \operatorname{diag}(\boldsymbol{\alpha})\big)^{-1}$ from (2.25). Furthermore, using the Woodbury matrix inversion identity [113], we can rewrite $\boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t}$ in (A.23) as

$$\begin{aligned}
\boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t} &= \boldsymbol{t}^T(\tau^{-1}\boldsymbol{I} + \boldsymbol{\Phi}\big(\operatorname{diag}(\boldsymbol{\alpha})\big)^{-1}\boldsymbol{\Phi}^T)^{-1}\boldsymbol{t} \\
&= \boldsymbol{t}^T\big[\tau\boldsymbol{I} - \tau^2\boldsymbol{\Phi}\big(\tau\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \operatorname{diag}(\boldsymbol{\alpha})\big)^{-1}\boldsymbol{\Phi}^T\big]\boldsymbol{t} \\
&= \tau\boldsymbol{t}^T\boldsymbol{t} - \tau^2\boldsymbol{t}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t}. \qquad \text{(A.25)}
\end{aligned}$$

By inserting the α_m dependent terms from (A.24) and (A.25) into (A.23) we obtain

$$\begin{aligned}
\frac{\partial\ln p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})}{\partial\alpha_m} &= \frac{\partial}{\partial\alpha_m}\Big\{\frac{1}{2}\sum_{m=1}^{M}\ln\alpha_m + \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \frac{\tau^2}{2}\boldsymbol{t}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t}\Big\} \\
&= \frac{1}{2\alpha_m} + \frac{1}{2}\operatorname{tr}\big(\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\alpha_m}\big) + \frac{\tau^2}{2}\boldsymbol{t}^T\boldsymbol{\Phi}\frac{\partial\boldsymbol{\Sigma}}{\partial\alpha_m}\boldsymbol{\Phi}^T\boldsymbol{t} = 0, \qquad \text{(A.26)}
\end{aligned}$$

where we have used the matrix derivative rule $\frac{\partial\ln|\boldsymbol{B}|}{\partial z} = \operatorname{tr}(\boldsymbol{B}^{-1}\frac{\partial\boldsymbol{B}}{\partial z})$ with $\operatorname{tr}(\cdot)$ being the trace opterator, cf. [116, 4]. Using the derivative rule for inverse matrices $\frac{\partial\boldsymbol{B}^{-1}}{\partial z} = -\boldsymbol{B}^{-1}\frac{\partial\boldsymbol{B}}{\partial z}\boldsymbol{B}^{-1}$ [116, 4], it is easy to show that

$$\frac{\partial\boldsymbol{\Sigma}}{\partial\alpha_m} = \frac{\partial(\boldsymbol{\Sigma}^{-1})^{-1}}{\partial\alpha_m} = -\boldsymbol{\Sigma}\frac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\alpha_m}\boldsymbol{\Sigma},$$

where, by inserting for $\boldsymbol{\Sigma}^{-1}$ using (2.25), we obtain

$$\frac{\partial\boldsymbol{\Sigma}}{\partial\alpha_m} = -\boldsymbol{\Sigma}\frac{\partial\operatorname{diag}(\boldsymbol{\alpha})}{\partial\alpha_m}\boldsymbol{\Sigma}. \qquad \text{(A.27)}$$

Note that the matrix $\frac{\partial \operatorname{diag}(\boldsymbol{\alpha})}{\partial \alpha_m}$ has zeros everywhere except for the mth main diagonal element, which contains a 1. If we insert (A.27) into (A.26) we finally obtain

$$\begin{aligned}\frac{\partial \ln p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})}{\partial \alpha_m} &= \frac{1}{2\alpha_m} - \frac{\Sigma_{mm}}{2} - \frac{1}{2}\boldsymbol{\mu}^T \frac{\partial \operatorname{diag}(\boldsymbol{\alpha})}{\partial \alpha_m}\boldsymbol{\mu} \\ &= \frac{1}{2\alpha_m} - \frac{\Sigma_{mm}}{2} - \frac{1}{2}\mu_m^2 = 0, \end{aligned} \tag{A.28}$$

where we have used $\boldsymbol{\mu} = \tau\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t}$ from (2.26). The result (2.29), namely $\alpha_m = (\mu_m^2 + \Sigma_{mm})^{-1}$, directly follows from (A.28). Note that (2.29) is not explicit in α_m but it can be incorporated into an iterative update procedure depending on the previous state of α_m through μ_m and Σ_{mm} as in (2.22). To finalize the proof, we need to show that (2.29) is a maximum of $\ln p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})$ with respect to α_m for fixed values μ_m and Σ_{mm} from the previous iteration. This can be easily seen from

$$\left.\frac{\partial^2 \ln p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})}{\partial \alpha_m^2}\right|_{\alpha_m=(\mu_m^2+\Sigma_{mm})^{-1}} = -\frac{(\mu_m^2+\Sigma_{mm})^2}{2} \leq 0, \tag{A.29}$$

where we consider $\mu_m^2 + \Sigma_{mm} > 0$ in general. $\mu_m^2 + \Sigma_{mm}$ is only zero if we have reached a mode of $p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})$ that corresponds to an irrelevant basis function $\psi_m(\cdot)$, i.e., $\alpha_m = \infty$.

To derive the result (2.30), we compute the derivative of the log of the marginal likelihood with respect to τ and equate to zero. That is,

$$\frac{\partial \ln p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})}{\partial \tau} = \frac{\partial}{\partial \tau}\Big\{ -\frac{N}{2}\ln 2\pi - \frac{1}{2}\ln|\boldsymbol{C}| - \frac{1}{2}\boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t}\Big\} = 0, \tag{A.30}$$

which can be simplified to

$$\begin{aligned}\frac{\partial \ln p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})}{\partial \tau} &= -\frac{1}{2}\frac{\partial \ln|\boldsymbol{C}|}{\partial \tau} - \frac{1}{2}\boldsymbol{t}^T\frac{\partial \boldsymbol{C}^{-1}}{\partial \tau}\boldsymbol{t} \\ &= -\frac{1}{2}\operatorname{tr}\big(\boldsymbol{C}^{-1}\frac{\partial \boldsymbol{C}}{\partial \tau}\big) + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{C}^{-1}\frac{\partial \boldsymbol{C}}{\partial \tau}\boldsymbol{C}^{-1}\boldsymbol{t} = 0, \end{aligned} \tag{A.31}$$

where

$$\frac{\partial \boldsymbol{C}}{\partial \tau} = \frac{\partial \tau^{-1}\boldsymbol{I}}{\partial \tau} = -\tau^{-2}\boldsymbol{I}. \tag{A.32}$$

By inserting (A.32) into (A.31) and applying the matrix inversion identity, we can finally obtain

$$\frac{\partial \ln p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})}{\partial \tau} = \frac{1}{2}\operatorname{tr}\Big(\frac{1}{\tau^2}\boldsymbol{C}^{-1}\Big) - \frac{1}{2\tau^2}\boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{C}^{-1}\boldsymbol{t}$$

$$
\begin{aligned}
&= \frac{1}{2}\operatorname{tr}\Big(\frac{1}{\tau^2}(\tau\boldsymbol{I} - \tau^2\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T)\Big) - \\
&\quad \frac{1}{2\tau^2}\boldsymbol{t}^T(\tau\boldsymbol{I} - \tau^2\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T)(\tau\boldsymbol{I} - \tau^2\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T)\boldsymbol{t} \\
&= \frac{1}{2}\operatorname{tr}(\tau^{-1}\boldsymbol{I}) - \frac{1}{2}\operatorname{tr}\big(\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\big) - \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} + \tau\boldsymbol{t}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t} - \\
&\quad \tau^2\boldsymbol{t}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t} \\
&= \frac{N}{2\tau} - \frac{1}{2}\operatorname{tr}\big(\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\big) - \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} + \boldsymbol{t}^T\boldsymbol{\Phi}\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\mu} \\
&= \frac{N}{2\tau} - \frac{1}{2}\operatorname{tr}\big(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\big) - \frac{1}{2}\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 = 0, \qquad \text{(A.33)}
\end{aligned}
$$

where we have used the identity $\operatorname{tr}(\boldsymbol{PQR}) = \operatorname{tr}(\boldsymbol{QRP}) = \operatorname{tr}(\boldsymbol{RPQ})$ [116, 4]. From (A.33) we can directly compute (2.30), which is also not explicit in τ and we need to incorporate it into a sequential update scheme that depends on previous states via $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ like in (2.22). For given $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ (from the previous iteration), (2.30) is clearly a maximum, since we have

$$
\left.\frac{\partial^2 \ln p(\boldsymbol{t}|\tau,\boldsymbol{\alpha})}{\partial\tau^2}\right|_{\tau=\frac{N}{\|\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{\mu}\|^2+\operatorname{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi})}} = -\frac{(\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + \operatorname{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}))^2}{N} \leq 0, \qquad \text{(A.34)}
$$

where we consider the general case $\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + \operatorname{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}) > 0$. Note that in the limit case $\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + \operatorname{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}) = 0$, $\tau = \infty$ and the model exactly (over-)fits the training data, since we must have $\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\| = 0$ in this case, because $\operatorname{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi})$ is always greater or equal zero.

A.3 Derivation of Equation (2.37) and (2.38)

If we apply the MAP-EM algorithm exactly as described in Section 2.1.3 for the MAP estimation of $\boldsymbol{\alpha}$ and τ where we consider $\boldsymbol{w}$ as the hidden variables we obtain the following method:

MAP-EM for $\boldsymbol{\alpha}$ and τ in SBL

Goal: maximize $p(\boldsymbol{\alpha}, \tau|\boldsymbol{t})$ w.r.t. $\boldsymbol{\alpha}$ and τ

E-step:

Determine: $q(\boldsymbol{w}) = p(\boldsymbol{w}|\boldsymbol{t}, \tau^{\text{old}}, \boldsymbol{\alpha}^{\text{old}})$

Define: $Q\Big(\{\boldsymbol{\alpha}, \tau\}, \{\boldsymbol{\alpha}^{\text{old}}, \tau^{\text{old}}\}\Big) = \mathbb{E}_{q(\boldsymbol{w})}\{\ln p(\boldsymbol{t}, \tau, \boldsymbol{w}, \boldsymbol{\alpha})\}$

M-step:

Update: $\{\boldsymbol{\alpha}^{\text{new}}, \tau^{\text{new}}\} = \operatorname{argmax}_{\{\boldsymbol{\alpha},\tau\}} Q\Big(\{\boldsymbol{\alpha}, \tau\}, \{\boldsymbol{\alpha}^{\text{old}}, \tau^{\text{old}}\}\Big)$

The weight posterior $q(\boldsymbol{w})$ is equivalent to (2.24), where we have τ^{old} and $\hat{\boldsymbol{\alpha}}^{\text{old}}$ instead of τ and $\boldsymbol{\alpha}$, respectively. This fully specifies the E-step.

For the M-step, the joint maximization over $\boldsymbol{\alpha}$ and τ separate into the individual optimizations over α_m, $\forall m$, and τ. That is,

$$\begin{aligned}\{\boldsymbol{\alpha}^{\text{new}}, \tau^{\text{new}}\} &= \underset{\{\boldsymbol{\alpha},\tau\}}{\operatorname{argmax}} \mathbb{E}_{q(\boldsymbol{w})}\Big\{ \ln p(\boldsymbol{t}, \tau, \boldsymbol{w}, \boldsymbol{\alpha})\Big\} \\ &= \underset{\{\boldsymbol{\alpha},\tau\}}{\operatorname{argmax}} \mathbb{E}_{q(\boldsymbol{w})}\Big\{ \ln p(\boldsymbol{t}|\tau, \boldsymbol{w}) + \ln p(\tau) \\ &\qquad + \ln p(\boldsymbol{w}|\boldsymbol{\alpha}) + \ln p(\boldsymbol{\alpha})\Big\}\end{aligned}$$

separates into

$$\tau^{\text{new}} = \underset{\tau}{\operatorname{argmax}} \mathbb{E}_{q(\boldsymbol{w})}\Big\{ \ln p(\boldsymbol{t}|\tau, \boldsymbol{w}) + \ln p(\tau)\Big\} \tag{A.35}$$

and

$$\boldsymbol{\alpha}^{\text{new}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \mathbb{E}_{q(\boldsymbol{w})}\Big\{ \ln p(\boldsymbol{w}|\boldsymbol{\alpha}) + \ln p(\boldsymbol{\alpha})\Big\}, \tag{A.36}$$

$$= \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \mathbb{E}_{q(\boldsymbol{w})}\Big\{ \sum_{m=1}^{M} \Big[\ln p(w_m|\alpha_m) + \ln p(\alpha_m)\Big]\Big\}, \tag{A.37}$$

where the latter further separates into

$$\alpha_m^{\text{new}} = \underset{\alpha_m}{\operatorname{argmax}} \mathbb{E}_{q(\boldsymbol{w})}\Big\{ \ln p(w_m|\alpha_m) + \ln p(\alpha_m)\Big\}, \quad \forall m. \tag{A.38}$$

Let us now analyze (A.38) first. Since $p(\alpha_m)$ is a *gamma distribution* and a conjugate prior for the Gaussian $p(w_m|\alpha_m)$, the term within the expectation in (A.38) has the form of a log gamma distribution over α_m. Note that the pdf normalization only adds a constant in the log-domain. We can further simply the expectation in (A.38) as

$$\begin{aligned}&\mathbb{E}_{q(\boldsymbol{w})}\Big\{ \ln p(w_m|\alpha_m) + \ln p(\alpha_m)\Big\} \\ &= \mathbb{E}_{q(\boldsymbol{w})}\Big\{ \frac{1}{2}\ln \alpha_m - \frac{\alpha_m}{2} w_m^2 + (a-1)\ln \alpha_m - b\alpha_m\Big\} + const. \\ &= \Big(a - \frac{1}{2}\Big)\ln \alpha_m - \Big(\frac{1}{2}\mathbb{E}_{q(\boldsymbol{w})}\{w_m^2\} + b\Big)\alpha_m + const. \\ &= \ln \mathrm{Ga}(\alpha_m|\tilde{a}_m, \tilde{b}_m) + const.,\end{aligned}$$

where the parameters $\tilde{a}_m$ and $\tilde{b}_m$ can be obtained as

$$\tilde{a}_m = a + \frac{1}{2} \tag{A.39}$$

$$\tilde{b}_m = b + \frac{1}{2}\mathbb{E}_{q(\boldsymbol{w})}\{w_m^2\} = b + \frac{1}{2}(\mu_m^2 + \Sigma_{mm}). \tag{A.40}$$

The mode of a gamma distribution $\mathrm{Ga}(\alpha_m|\tilde{a}_m, \tilde{b}_m) \propto \alpha_m^{\tilde{a}-1}\exp\{-\tilde{b}\alpha_m\}$ is $(\tilde{a}_m - 1)/\tilde{b}_m$ for $\tilde{a}_m > 1$ [4] and we consider it as zero for $0 \leq \tilde{a}_m \leq 1$, since the pdf has its largest values at zero in this case. Note that in [4], the gamma pdf is only defined for $\alpha_m > 0$ and thus the mode at zero does not exist. However, throughout the thesis, as already mentioned in Section 2.2.1, we consider the gamma pdf for $\alpha_m \geq 0$. From this perspective, (A.38) directly leads to (2.37), as we wanted to show.

Similarly, we can derive the update (2.38) from (A.35) with

$$\begin{aligned}
&\mathbb{E}_{q(\boldsymbol{w})}\Big\{\ln p(\boldsymbol{t}|\tau, \boldsymbol{w}) + \ln p(\tau)\Big\} \\
&= \mathbb{E}_{q(\boldsymbol{w})}\Big\{\frac{N}{2}\ln\tau - \frac{\tau}{2}\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 + (c-1)\ln\tau - \tau d\Big\} + const. \\
&= \frac{2c-2+N}{2}\ln\tau - \frac{\tau}{2}\Big(\boldsymbol{t}^T\boldsymbol{t} - 2\boldsymbol{t}^T\boldsymbol{\Phi}\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\mu} + \mathrm{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}) + 2d\Big) + const. \\
&= \frac{2c-2+N}{2}\ln\tau - \frac{\tau}{2}\Big(\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + \mathrm{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi}) + 2d\Big) + const. \\
&= \ln\mathrm{Ga}(\tau|\tilde{c}, \tilde{d}) + const.
\end{aligned} \tag{A.41}$$

where we have used the standard identity for expectations of quadratic forms $\mathbb{E}_{q(\boldsymbol{w})}\{\boldsymbol{w}^T\boldsymbol{B}\boldsymbol{w}\} = \boldsymbol{\mu}^T\boldsymbol{B}\boldsymbol{\mu} + \mathrm{tr}(\boldsymbol{\Sigma}\boldsymbol{B})$, cf. [116].

The parameters of the gamma distribution (A.41) can be obtained as

$$\tilde{c} = c + \frac{N}{2} \tag{A.42}$$

$$\tilde{d} = d + \frac{1}{2}\Big(\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2 + \mathrm{tr}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{\Phi})\Big). \tag{A.43}$$

By computing the mode of the gamma pdf (A.41) defined as $(\tilde{c}-1)/\tilde{d}$ for $\tilde{c} > 1$ and zero otherwise, we can finally obtain (2.38), as we wanted to show.

A.4 Proof of Theorem 2

Using the Woodbury matrix identity [113] for $\boldsymbol{C}_{\bar{m}} = \hat{\tau}^{-1}\boldsymbol{I} + \sum_{l\neq m} \hat{\alpha}_l^{-1}\boldsymbol{\varphi}_l\boldsymbol{\varphi}_l^T$, we obtain the inverse as

$$\begin{aligned}\boldsymbol{C}_{\bar{m}}^{-1} &= \left(\hat{\tau}^{-1}\boldsymbol{I} + \sum_{l\neq m} \hat{\alpha}_l^{-1}\boldsymbol{\varphi}_l\boldsymbol{\varphi}_l^T\right)^{-1} \\ &= \hat{\tau}\boldsymbol{I} - \hat{\tau}^2\boldsymbol{\Phi}_{\bar{m}} \underbrace{\left(\operatorname{diag}(\hat{\boldsymbol{\alpha}}_{\bar{m}}) + \hat{\tau}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\Phi}_{\bar{m}}\right)^{-1}}_{\hat{\boldsymbol{\Sigma}}_{\bar{m}}} \boldsymbol{\Phi}_{\bar{m}}^T,\end{aligned} \tag{A.44}$$

where $\boldsymbol{\Phi}_{\bar{m}}$ is $\boldsymbol{\Phi}$ without the mth basis $\boldsymbol{\varphi}_m$, $\hat{\boldsymbol{\alpha}}_{\bar{m}}$ is the vector $\hat{\boldsymbol{\alpha}}$ without the element $\hat{\alpha}_m$, and we used the equivalence $\sum_{l\neq m} \hat{\alpha}_l^{-1}\boldsymbol{\varphi}_l\boldsymbol{\varphi}_l^T = \boldsymbol{\Phi}_{\bar{m}} \operatorname{diag}(\hat{\boldsymbol{\alpha}}_{\bar{m}})\boldsymbol{\Phi}_{\bar{m}}^T$. With $s_m = \boldsymbol{\varphi}_m^T\boldsymbol{C}_{\bar{m}}^{-1}\boldsymbol{\varphi}_m$ and $q_m = \boldsymbol{\varphi}_m^T\boldsymbol{C}_{\bar{m}}^{-1}\boldsymbol{t}$ we obtain

$$s_m = \hat{\tau}\boldsymbol{\varphi}_m^T\boldsymbol{\varphi}_m - \hat{\tau}^2\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}\hat{\boldsymbol{\Sigma}}_{\bar{m}}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{\varphi}_m \tag{A.45}$$

and

$$q_m = \hat{\tau}\boldsymbol{\varphi}_m^T\boldsymbol{t} - \hat{\tau}^2\boldsymbol{\varphi}_m^T\boldsymbol{\Phi}_{\bar{m}}\hat{\boldsymbol{\Sigma}}_{\bar{m}}\boldsymbol{\Phi}_{\bar{m}}^T\boldsymbol{t}. \tag{A.46}$$

From (A.45) and (A.46), in combination with the results (3.23) obtained in Section 3.3.1 for efficient computation of ς_m and ρ_m, it is easy to see that $\varsigma_m = s_m^{-1}$ and $\rho_m^2 = q_m^2/s_m^2$. Thus, the condition $q_m^2 > s_m$ in (3.19) is equivalent to the condition $\rho_m^2 > \varsigma_m$ in (3.12).

Appendix B

Jeffreys Prior Limit of the General Fast Variational Sparse Bayesian Learning Fixed-Point

In this appendix we show that the fixed point solutions (3.9) can be equivalently obtained by computing the Jeffreys' prior limit of the general fixed point equations.

By solving (3.6) explicitly for $\hat{\alpha}_m$, we obtain three stationary points as a function of the gamma prior parameters a and b as[1]

$$
\begin{aligned}
\hat{\alpha}_{m,I}^{[\infty]}\Big(a,b\Big) =& \frac{1}{3\ 2^{2/3} b\varsigma_m}\Bigg\{ 4a^3\varsigma_m^3 + 3b\Big(\Big[-3\varsigma_m^2\left(a^2\rho_m^4 - \right. \\
& 2(a(12a+11)+3)b\rho_m^2 + b^2\big) - 6\varsigma_m^3\left((a+1)b - a^2(4a+1)\rho_m^2\right) - \\
& 3a^2\varsigma_m^4 + 6b\rho_m^2\varsigma_m\left(2(6a+5)b - (10a+3)\rho_m^2\right) + 6b\left(2b\rho_m + \rho_m^3\right)^2\Big]^{1/2} + \\
& a\varsigma_m\left((4a+3)\varsigma_m - 3\rho_m^2\right)\Big) + 3b^2\left((4a+3)\varsigma_m + 6\rho_m^2\right) + 4b^3\Bigg\}^{1/3} + \\
& \Big(2a^2\varsigma_m^2 + (4a+3)b\varsigma_m + b\left(2b - 3\rho_m^2\right)\Big) / \\
& \Big\{ 3b\varsigma_m\Big[8a^3\varsigma_m^3 + 6b\left(\{ -3\varsigma_m^2\left(a^2\rho_m^4 - 2(a(12a+11)+3)b\rho_m^2 + b^2\right) - \right. \\
& 6\varsigma_m^3\left((a+1)b - a^2(4a+1)\rho_m^2\right) - 3a^2\varsigma_m^4 + 6b\rho_m^2\varsigma_m\left(2(6a+5)b - \right. \\
& \left.\left.(10a+3)\rho_m^2\right) + 6b\left(2b\rho_m + \rho_m^3\right)^2\right\} + a\varsigma_m\left((4a+3)\varsigma_m - 3\rho_m^2\right)\Big) + \\
& \left. 6b^2\left((4a+3)\varsigma_m + 6\rho_m^2\right) + 8b^3\Big]^{1/3}\right\} + \frac{1}{3}\left(\frac{a}{b} - \frac{2}{\varsigma_m}\right),
\end{aligned}
\tag{B.1}
$$

[1] This results and the following limit computations are obtained using Wolfram Mathematica®, Version 8.0.4.0.

$$
\begin{aligned}
\hat{\alpha}^{[\infty]}_{m,II}\Big(a,b\Big) =& \frac{1}{6\ 2^{2/3} b\varsigma_m}\Bigg\{ j\left(\sqrt{3}+j\right)\Bigg[4a^3\varsigma_m^3 + 3b\Big(\Big\{ -3\varsigma_m^2\left(a^2\rho_m^4 - \right.\\
& \left. 2(a(12a+11)+3)b\rho_m^2 + b^2\right) - 6\varsigma_m^3\left((a+1)b - \right.\\
& \left. a^2(4a+1)\rho_m^2\right) - 3a^2\varsigma_m^4 + 6b\rho_m^2\varsigma_m\left(2(6a+5)b - \right.\\
& \left. (10a+3)\rho_m^2\right) + 6b\left(2b\rho_m + \rho_m^3\right)^2\Big\}^{1/2} + a\varsigma_m\left((4a+3)\varsigma_m - 3\rho_m^2\right)\Big) + \\
& 3b^2\left((4a+3)\varsigma_m + 6\rho_m^2\right) + 4b^3\Bigg]^{1/3}\Bigg\} - \\
& \Big(j\left(\sqrt{3}-j\right)\left(2a^2\varsigma_m^2 + (4a+3)b\varsigma_m + b\left(2b - 3\rho_m^2\right)\right)\Big) / \\
& \Big\{6b\varsigma_m\Big[8a^3\varsigma_m^3 + 6b\Big(\left(-3\varsigma_m^2\left(a^2\rho_m^4 - 2(a(12a+11)+3)b\rho_m^2 + b^2\right) - \right.\\
& \left. 6\varsigma_m^3\left((a+1)b - a^2(4a+1)\rho_m^2\right) - 3a^2\varsigma_m^4 + 6b\rho_m^2\varsigma_m\left(2(6a+5)b - \right.\right.\\
& \left.\left. (10a+3)\rho_m^2\right) + 6b\left(2b\rho_m + \rho_m^3\right)^2\right)^{1/2} + a\varsigma_m\left((4a+3)\varsigma_m - 3\rho_m^2\right)\Big) + \\
& 6b^2\left((4a+3)\varsigma_m + 6\rho_m^2\right) + 8b^3\Big]^{1/3}\Big\} + \frac{1}{3}\left(\frac{a}{b} - \frac{2}{\varsigma_m}\right)
\end{aligned}
\tag{B.2}
$$

and

$$
\begin{aligned}
\hat{\alpha}^{[\infty]}_{m,III}\Big(a,b\Big) =& -\frac{1}{6\ 2^{2/3} b\varsigma_m}\Bigg\{ j\left(\sqrt{3}-j\right)\Bigg[4a^3\varsigma_m^3 + 3b\Big(\Big\{ -3\varsigma_m^2\left(a^2\rho_m^4 - \right.\\
& \left. 2(a(12a+11)+3)b\rho_m^2 + b^2\right) - 6\varsigma_m^3\left((a+1)b - \right.\\
& \left. a^2(4a+1)\rho_m^2\right) - 3a^2\varsigma_m^4 + 6b\rho_m^2\varsigma_m\left(2(6a+5)b - \right.\\
& \left. (10a+3)\rho_m^2\right) + 6b\left(2b\rho_m + \rho_m^3\right)^2\Big\}^{1/2} + a\varsigma_m\left((4a+3)\varsigma_m - 3\rho_m^2\right)\Big) + \\
& 3b^2\left((4a+3)\varsigma_m + 6\rho_m^2\right) + 4b^3\Bigg]^{1/3}\Bigg\} + \\
& \Big(j\left(\sqrt{3}+j\right)\left(2a^2\varsigma_m^2 + (4a+3)b\varsigma_m + b\left(2b - 3\rho_m^2\right)\right)\Big) / \\
& \Big\{6b\varsigma_m\Big[8a^3\varsigma_m^3 + 6b\Big(\left(-3\varsigma_m^2\left(a^2\rho_m^4 - 2(a(12a+11)+3)b\rho_m^2 + b^2\right) - \right.\\
& \left. 6\varsigma_m^3\left((a+1)b - a^2(4a+1)\rho_m^2\right) - 3a^2\varsigma_m^4 + 6b\rho_m^2\varsigma_m\left(2(6a+5)b - \right.\right.\\
& \left.\left. (10a+3)\rho_m^2\right) + 6b\left(2b\rho_m + \rho_m^3\right)^2\right) + a\varsigma_m\left((4a+3)\varsigma_m - 3\rho_m^2\right)\Big) + \\
& 6b^2\left((4a+3)\varsigma_m + 6\rho_m^2\right) + 8b^3\Big]^{1/3}\Big\} + \frac{1}{3}\left(\frac{a}{b} - \frac{2}{\varsigma_m}\right),
\end{aligned}
\tag{B.3}
$$

where j stands for the imaginary unit of complex numbers. Note that we are only interested in solutions $\hat{\alpha}^{[\infty]}_m \in \mathbb{R}_0^+ \cup \{\infty\}$, where we used the notation $\mathbb{R}_0^+ = \{\nu \in \mathbb{R} : \nu \geq 0\}$ to denote non-negative real numbers.

We now compute the limit $a \to 0$, $b \to 0$ for all angles in which a and b could possibly reach the origin in the ab-plane. Since for proper Gamma distributions

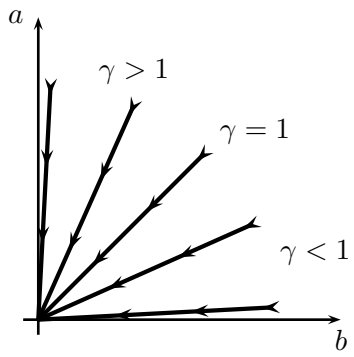


Figure B.1: All possible directions in which a and b could approach the point $(0, 0)$.

we have $a > 0$ and $b > 0$, we can introduce a slope variable $\gamma \in \mathbb{R}^+$, where $\mathbb{R}^+ = \{\nu \in \mathbb{R} : \nu > 0\}$, and define $a = \gamma b$. By computing the limits

$$\hat{\alpha}_{m,s}^{[\infty]} = \lim_{b \to 0} \hat{\alpha}_{m,s}^{[\infty]}\Big(\gamma b, b\Big), \quad \forall s \in \{I, II, III\}, \tag{B.4}$$

it can be shown that the solutions $\hat{\alpha}_{m,s}^{[\infty]}$ are valid for all ab-directions, as depicted in Figure B.1, if they are independent of the slope variable γ.

We determine the solutions which are indeed independent of γ as

$$\hat{\alpha}_{m,I}^{[\infty]} = \begin{cases} (\rho_m^2 - \varsigma_m)^{-1}, & \rho_m^2 > \varsigma_m \\ \infty, & \rho_m^2 \leq \varsigma_m \end{cases}, \tag{B.5}$$

$$\hat{\alpha}_{m,II}^{[\infty]} = \begin{cases} j\infty, & \rho_m^2 > \varsigma_m \\ -\infty, & \rho_m^2 = \varsigma_m = 0 \\ \infty \cdot e^{j\frac{2\pi}{3}}, & \rho_m^2 = \varsigma_m > 0 \\ -\infty & \rho_m^2 < \varsigma_m \end{cases}, \tag{B.6}$$

and

$$\hat{\alpha}_{m,III}^{[\infty]} = \begin{cases} -j\infty, & \rho_m^2 > \varsigma_m \\ -\infty, & \rho_m^2 = \varsigma_m = 0 \\ \infty \cdot e^{-j\frac{2\pi}{3}}, & \rho_m^2 = \varsigma_m > 0 \\ (\rho_m^2 - \varsigma_m)^{-1} & \rho_m^2 < \varsigma_m \end{cases}, \tag{B.7}$$

where the notation $\infty \cdot e^{j\theta}$ denotes a type of infinity in the complex plane with an angle θ, sometimes also denoted as directed infinity. Since we only need to consider

solutions in $\mathbb{R}_0^+ \cup \{\infty\}$, it is easy to see that $\hat{\alpha}_{m,I}^{[\infty]}$ from (B.5) is the only solution that always lies in this domain. The other solutions (B.6) and (B.7) never lie in $\mathbb{R}_0^+ \cup \{\infty\}$ and can thus be ignored. Note that even the finite number $(\rho_m^2 - \varsigma_m)^{-1}$ in (B.7) is not in the desired domain as it becomes negative under the condition $\rho_m^2 < \varsigma_m$.

Since the only valid solution $\hat{\alpha}_{m,I}^{[\infty]}$ from (B.5) is equivalent to (3.9), we have shown that setting $a = b = 0$ before computing the fixed points is the same as computing the limit of the general fixed point with $a > 0$ and $b > 0$ unspecified. In other words, it is reasonable to work with an improper hyperprior. Finally, note that also the stability analysis can be done in the limit case, where it is straight forward to show its equivalence to the analysis in Section 3.2.2.

Appendix C

Elementary Gaussian Loopy Belief Propagation in Variational Distributed Sparse Bayesian Learning

In Chapter 4, Section 4.3.2, we discussed a variational distributed SBL algorithm (V-dSBL) which is based on a combination of V-SBL [35] and consensus propagation [67]. We showed that the update of the weight posterior, which is a subroutine of the V-dSBL algorithm, can be performed distributedly using Gaussian loopy belief propagation (Gaussian LBP). As depicted in Figure 4.2(b), the random variables at each node $k = 1, \ldots, K$ are considered as multivariate Gaussians of dimension M. That is, the messages that need to be communicated between the nodes, by applying Gaussian LBP, are precision matrices (inverse covariance matrices) and mean vectors. These messages, which are defined in (4.16) and (4.17) respectively, have a communicational complexity of $O(M^2)$ due to sending $\hat{\mathbf{\Lambda}}_{kl}$. Furthermore, as matrices of size $M \times M$ need to be inverted, the computational complexity is $O(M^3)$ for each message. Since the communication and computation is very complex for a large number of basis functions M, there would be a more efficient alternative by applying what we denote as elementary Gaussian LBP. This approach considers the graph of all elementary Gaussian random variables, i.e., the messages have to be sent between the elements inside each vector $\boldsymbol{w}_k$ and between the elements of the vectors $\boldsymbol{w}_k$ and $\boldsymbol{w}_l$ for all $(k, l) \in \mathcal{E}$. Note that in this case the communication complexity between the sensors is $O(M)$ due to the non-crossing connections of the coupling factors (4.12), i.e., there are only scalar mean and scalar precision messages between $w_{k,i}$ and $w_{l,j}$ for $i = j$ and $(k, l) \in \mathcal{E}$. In Section C.2, we show how elementary Gaussian LBP can be implemented for the V-dSBL model. The reason why elementary Gaussian LBP is potentially applicable for V-dSBL, is because we only require the marginal mean and marginal variance components $\hat{\mu}_{k,m}$ and $\hat{\Sigma}_{k,mm}$

in the hyperparameter updates $\hat{\alpha}_{k,m} = (2a+1)/(2b+\hat{\mu}_{k,m}^2+\hat{\Sigma}_{k,mm})$ at each sensor k (cf. Algorithm 4.1). For elementary Gaussian graphs in the form of a tree, the sum-product algorithm [4] would exactly lead to marginal means and marginal variances. However, in general, the elementary Gaussian graph for the V-dSBL model has no tree structure since the elements of each $\boldsymbol{w}_k$ are already fully connected via the factors (4.11) and thus form a loopy graph. That means, even multivariate Gaussian tree graphs discussed in Section 4.3.2 are essentially elementary Gaussian loopy graphs.

Sufficient convergence criteria for elementary Gaussian LBP can be found in [107, 106]. Unfortunately in cases where the convergence according to [107, 106] is guaranteed, only the marginal mean estimators are correct. That is, although the marginal variance estimators converge, their resulting estimates are typically wrong. Note that this influences the outcome of the overall V-dSBL prediction, since the hyperparameter updates depend on both, the marginal means and variances. However, in [106, Section 5] it is shown, that the variance error in general tends be small, except for situations close to the point where where the convergence criteria is not fulfilled. In the following we show how the convergence criteria [106] can be evaluated for the elementary Gaussian LBP approach to V-dSBL.

C.1 Convergence of elementary Gaussian loopy belief propagation in variational distributed sparse Bayesian learning

As mentioned before, sufficient convergence criteria for elementary Gaussian LBP are given in [107, 106]. We only consider the criteria [106], which is denoted as *walk-summability*, for our analysis here, since it is more general than the criteria presented in [107]. More specifically, all models satisfying the criteria proposed in [107] also satisfy the criteria proposed in [106].

Consider the joint density over all local weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ collected to a global vector $\bar{\boldsymbol{w}} = [\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_K^T]^T \in \mathbb{R}^{MK}$ given as $q(\bar{\boldsymbol{w}}) = \mathcal{N}(\bar{\boldsymbol{w}}|\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Lambda}}^{-1})$. The global precision matrix $\bar{\boldsymbol{\Lambda}}$ or sometimes denoted as information matrix (as in [106]) is essential for determining if elementary Gaussian LBP converges. It can be obtained from the product of all factors in the graph presented in Figure 4.2(b) with the

observed variables inserted, i.e.,

$$q(\bar{\boldsymbol{w}}) \propto \prod_{k=1}^{K} f_{k\alpha}(\hat{\boldsymbol{\alpha}}_k, \boldsymbol{w}_k) f_{kt}(t_k, \boldsymbol{w}_k) \prod_{\{k,l\}\in\mathcal{E}} f_{kl}(\boldsymbol{w}_k, \boldsymbol{w}_l). \tag{C.1}$$

When we take the logarithm of both sides in (C.1) and use the term '*const.*' to denote all parts that are independent of $\bar{\boldsymbol{w}}$, we obtain

$$\begin{aligned}
\ln q(\bar{\boldsymbol{w}}) =& -\frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{\Lambda}}\bar{\boldsymbol{w}} + \bar{\boldsymbol{\mu}}^T\bar{\boldsymbol{\Lambda}}\bar{\boldsymbol{w}} + const. \\
=& -\frac{1}{2}\sum_{k=1}^{K}\boldsymbol{w}_k^T\Big(\frac{1}{K}\hat{\boldsymbol{A}}_k + \tau\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T\Big)\boldsymbol{w}_k - \frac{\beta}{2}\sum_{\{k,l\}\in\mathcal{E}}\|\boldsymbol{w}_k - \boldsymbol{w}_l)\|_2^2 \\
&+ \tau\sum_{k=1}^{K} t_k\boldsymbol{\phi}_k^T\boldsymbol{w}_k + const. \\
=& -\frac{1}{2}\sum_{k=1}^{K}\boldsymbol{w}_k^T\Big(\frac{1}{K}\hat{\boldsymbol{A}}_k + \tau\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T + \#N(k)\beta\boldsymbol{I}\Big)\boldsymbol{w}_k \\
&- \frac{1}{2}\sum_{\{k,l\}\in\mathcal{E}}\Big[\boldsymbol{w}_k^T(-\beta\boldsymbol{I})\boldsymbol{w}_l + \boldsymbol{w}_l^T(-\beta\boldsymbol{I})\boldsymbol{w}_k\Big] \\
&+ \tau\sum_{k=1}^{K} t_k\boldsymbol{\phi}_k^T\boldsymbol{w}_k + const.\,,
\end{aligned} \tag{C.2}$$

and can directly determine $\bar{\boldsymbol{\Lambda}}$ as

$$\bar{\boldsymbol{\Lambda}} = \begin{bmatrix}
\boldsymbol{D}_1 & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \boldsymbol{D}_2 & & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & & \ddots & & & & & \vdots \\
\mathbf{0} & \mathbf{0} & & \boldsymbol{D}_k & & -\beta\boldsymbol{I} & & \mathbf{0} \\
\vdots & \vdots & & & \ddots & & & \vdots \\
\mathbf{0} & \mathbf{0} & & -\beta\boldsymbol{I} & & \boldsymbol{D}_l & & \mathbf{0} \\
\vdots & \vdots & & & & & \ddots & \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} & & \boldsymbol{D}_K
\end{bmatrix}, \tag{C.3}$$

where $\boldsymbol{D}_k = K^{-1}\hat{\boldsymbol{A}}_k + \tau\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T + \#N(k)\beta\boldsymbol{I}$. Note that in the representation (C.3), the off-diagonal block matrices are all zero, except for those corresponding to a connection between sensors k and l, i.e., for all $\{k,l\} \in \mathcal{E}$ we have two off-diagonal matrices $-\beta\boldsymbol{I}$ symmetrically placed at the block positions (k,l) and (l,k) indicated

by the main diagonal blocks $\boldsymbol{D}_k$ and $\boldsymbol{D}_l$ in (C.3). Note that in (C.3) we have only depicted one particular exemplary connection between k and l. However, as can be seen in (C.2), this off-diagonal block matrices are meant to occur at all positions wherever the respective sensors k and l are connected.

By defining the matrix

$$\boldsymbol{R} = \boldsymbol{I} - \mathrm{diag}(\bar{\boldsymbol{\Lambda}})^{-1/2}\ \bar{\boldsymbol{\Lambda}}\ \mathrm{diag}(\bar{\boldsymbol{\Lambda}})^{-1/2}, \tag{C.4}$$

according to [106], elementary Gaussian LBP is guaranteed to converge if the spectral radius of $|\boldsymbol{R}|$ is smaller than 1, i.e., $\rho(|\boldsymbol{R}|) < 1$, where we have used the operator $|\cdot|$ to denote the elementwise absolute value and define the spectral radius of some matrix $\boldsymbol{V}$ with eigenvalues v_i as $\rho(\boldsymbol{V}) = \max_i(|v_i|)$.

C.2 Implementation of elementary Gaussian loopy belief propagation for variational distributed sparse Bayesian learning

Algorithm C.1 summarizes the *synchronous update* version of the elementary Gaussian LBP algorithm for V-dSBL.

Algorithm C.1 Elementary Gaussian LBP in V-dSBL

We define: $\bar{\boldsymbol{\Lambda}}$ as in (C.3) and $\bar{\boldsymbol{h}} = \tau[t_1\boldsymbol{\phi}_1^T, \ldots, t_K\boldsymbol{\phi}_K^T]^T$
Initialize: $n = 0$, $\lambda_{ui}^{(0)} = 0$ and $h_{ui}^{(0)} = 0$ for all (u, i) where $\bar{\Lambda}_{ij} \neq 0$

% Message passing update loop
while (elementary Gaussian LBP not converged) **do**
  $n = n + 1$
  % Compute messages for all (i, j) where $\bar{\Lambda}_{ij} \neq 0$
  $\lambda_{ij}^{(n)} = -\bar{\Lambda}_{ij}^2(\bar{\Lambda}_{ii} + \sum_{u\in\tilde{N}(i)\setminus j} \lambda_{ui}^{(n-1)})^{-1}$
  $h_{ij}^{(n)} = \bar{\Lambda}_{ij}^{-1}\lambda_{ij}^{(n)}(\bar{h}_i + \sum_{u\in\tilde{N}(i)\setminus j} h_{ui}^{(n-1)})$
end while

% Compute the marginal estimates at each sensor k
Define: $z_{km} = (k-1)M + m$
$\hat{\Sigma}_{k,mm} = \big(\bar{\Lambda}_{z_{km}z_{km}} + \sum_{u\in\tilde{N}(z_{km})} \lambda_{uz_{km}}^{(n)}\big)^{-1}, \quad \forall k,\ \forall m$
$\hat{\mu}_{k,m} = \hat{\Sigma}_{k,mm}\big(h_{z_{km}} + \sum_{u\in\tilde{N}(z_{km})} h_{uz_{km}}^{(n)}\big), \quad \forall k,\ \forall m$

The algorithm is based on [106, 123] and adopted to the elementary V-dSBL

model. We make use of the notation $\tilde{N}(i)$ to denote the elementary neighbors of the elementary node i in the Gaussian graph. Similarly to the notation in Chapter 4, we use $\tilde{N}(i)\backslash j$ to denote the elementary neighbors of the elementary node i excluding the elementary node j. Note that in elementary Gaussian LBP, the messages to such neighbors do not always require communication between different sensors in the network. These messages are often sent locally between the elements of some vector $\boldsymbol{w}_k$ at sensor k. The vector $\bar{\boldsymbol{h}}$ in Algorithm C.1 is denoted as the potential vector [106] and is defined as $\bar{\boldsymbol{h}} = \tau[t_1\boldsymbol{\phi}_1^T, \ldots, t_K\boldsymbol{\phi}_K^T]^T$. This result can be easily obtained from (C.2) by defining $\bar{\boldsymbol{h}}^T\bar{\boldsymbol{w}} = \bar{\boldsymbol{\mu}}^T\bar{\boldsymbol{\Lambda}}\bar{\boldsymbol{w}}$, which leads to the equivalence $\bar{\boldsymbol{h}} = \bar{\boldsymbol{\Lambda}}\bar{\boldsymbol{\mu}}$. From the latter perspective it is clear that elementary Gaussian LBP can be used to solve systems of linear equations, a fundamental problem in computer science [124]. More specifically, if it converges, the marginal mean estimates are guaranteed to be equal to the elements of $\bar{\boldsymbol{\mu}}$ [107, 106].

C.3 Test experiments

We tried to apply the more efficient elementary Gaussian LBP approach for V-dSBL to the synthetic data experiment from Section 4.4.1. The simulation setup was equivalent, except that elementary Gaussian LBP was used instead of multivariate Gaussian LBP. Unfortunately, the convergence criteria [106] was never fulfilled at the initial variational iteration for any of the algorithm settings listed in the first two columns of Table 4.1. When we nevertheless tried to start the algorithm for some of these cases, we always obtained useless results, since all basis functions where pruned from the model. This typically happend already after the first variational iteration.

C.4 Summary and future directions

Unfortunately, in our experiments, V-dSBL did not work in combination with elementary Gaussian LBP and furthermore the convergence criteria based on *walk-summability* [106] was never fulfilled for elementary Gaussian LBP at the initial variational iteration.

However, as the idea seems to be, in terms of computation and communication, much more efficient than V-dSBL with multivariate Gaussian LBP as in Chapter 4, we give some comments on potential future research directions to overcome the problems.

In [123], a method is proposed for obtaining the true marginal means from *non-walk-summable* models. It is based on a double-loop algorithm which at each iteration performs elementary Gaussian LBP on an intermediate *walk-summable* model. This *walk-summability* is achieved by adding a proper diagonal matrix to the *non-walk-summable* precision matrix. The potential vector needs to be adapted at every iteration to assure convergence to the true mean of the original *non-walk-summable* model. Unfortunately, the method [123] has its focus on the application of elementary Gaussian LBP for solving systems of linear equations, and thus only on the marginal means. As we also need the marginal variances in V-dSBL, further research in this direction is required. Furthermore, the method adds another outer loop to the already existing message passing loop, which is assumed to lower the overall convergence speed. In this sense, the single-loop alternative, also proposed in [123], seems more interesting for V-dSBL.

Finally, we note that since even for *walk-summable* models, the marginal variance estimates obtained from elementary Gaussian LBP are typically wrong [107, 106], it would be of great interest in future research to investigate on how to obtain the correct marginal variances in a distributed way.

Appendix D

Sliding Window Online Fast Variational Sparse Bayesian Learning

In this appendix we present an online learning algorithm for fast adaptive nonlinear filtering based on fast variational sparse Bayesian learning (FV-SBL) introduced in Chapter 3. A sequential decision rule for inclusion and deletion of basis functions is applied to a sliding window block of data. In this manner, data can be processed online, where the set of basis functions is adjusted for each time instant.

We show that the proposed method has better sparsity and mean square error (MSE) performance compared to a state of the art online kernel recursive least squares (Kernel-RLS) algorithm. Furthermore, the proposed algorithm implicitly estimates the noise precision (inverse variance), which does not apply for Kernel-RLS.

D.1 Introduction

Online learning is used in diverse areas of signal processing [125]. In applications like system identification, channel equalization and time series prediction, in which data is time varying and arrives sequentially, online learning can be the only method of choice.

We define the available training set at time instant $i \geq 1$ as a set $\{\boldsymbol{x}_n, t_n\}_{n=1}^{i}$ consisting of D-dimensional real input vectors $\boldsymbol{x}_n \in \mathbb{R}^D$ and real valued scalar targets $t_n \in \mathbb{R}$.

We model the target t_i at time i as

$$t_i = \sum_{m=1}^{M_i} w_{i,m}\psi_{i,m}(\boldsymbol{x}_i) + \epsilon_i, \tag{D.1}$$

This appendix is based on our publication [73].

where $\psi_{i,m}(\cdot)$ is the mth basis function from a set of $m = 1, \ldots, M_i$ basis functions at time i, $w_{i,m}$ is the mth weight parameter at time i and ϵ_i is a zero mean additive white Gaussian perturbation with variance τ_i^{-1} at time i.

In online sparse signal representation, the aim is to find a small number of weighted basis functions M_i to represent the targets $t_1, \ldots, t_i$ for their respective inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i$ according to (D.1).

Recently, a new approach called kernel adaptive filtering [17] has emerged, which has its origin in kernel methods [11, 4], a powerful method in machine learning. One famous example of a kernel adaptive filters is the kernel recursive least squares (Kernel-RLS) algorithm [74]. In kernel methods, the basis functions are placed at the inputs $\boldsymbol{x}_n$, for $n = 1, \ldots, i$. Thus, kernel models are build directly from the data itself. Naturally, the online implementation of Kernel-RLS requires some sparsification rule, since data arrives sequentially and the model complexity would otherwise explode. The Kernel-RLS therefore uses a criteria called approximate linear dependency (ALD) which decides if the current kernel is added to the model or not. The ALD criteria requires a threshold parameter to be set in advance. The performance of the Kernel-RLS strongly depends on the adjustment of this threshold. Another drawback of the Kernel-RLS is its limitation to only kernel basis functions, which is due to the incorporation of the kernel-trick [11, 4].

In sparse Bayesian learning (SBL) [1, 35], there is no need for such a sparsification parameter and selecting components is done automatically using automatic relevance determination (ARD). Also, because SBL uses no kernel trick, it does not only rely on kernel basis functions like other kernel methods. Since SBL is designed for batch learning, i.e., access to all data is needed, it is not directly suitable for online processing. Another drawback, which limits the use of SBL in many online applications, is its slow convergence speed.

To overcome these drawbacks of SBL, we first suggest not to use all the data, which we cannot even access in most situations. Instead we propose to use a block of the last l samples. More specifically, we model the targets included in a sliding window block of length l at time $i \geq l$ as

$$\boldsymbol{t}_i = \boldsymbol{\Phi}_i \boldsymbol{w}_i + \boldsymbol{\epsilon}_i, \tag{D.2}$$

where $\boldsymbol{\Phi}_i = [\boldsymbol{\varphi}_{i,1}, \ldots, \boldsymbol{\varphi}_{i,M_i}]$ is a design matrix consisting of M_i basis vectors $\boldsymbol{\varphi}_{i,m} = [\psi_{i,m}(\boldsymbol{x}_{i-l+1}), \ldots, \psi_{i,m}(\boldsymbol{x}_i)]^T$ with basis functions $\psi_{i,m}(\cdot)$ at time i, the vector $\boldsymbol{t}_i = [t_{i-l+1}, \ldots, t_i]^T$ contains all the targets within the window at time i, the vector

$\boldsymbol{w}_i = [w_{i,1}, \ldots, w_{i,M_i}]^T$ is denoted as weight vector at time i and $\boldsymbol{\epsilon}_i$ is a zero mean additive white Gaussian perturbation vector with covariance matrix $\tau_i^{-1}\mathbf{I}$ as before.

Secondly, to learn the set of basis functions and their respective weights $\boldsymbol{w}_i$ at time i, we make use of the FV-SBL method proposed in Chapter 3, which is a variational counterpart to the fast marginal likelihood maximization method [68]. This method provides a fast decision rule for selecting model components and allows for addition of new basis functions as well as deletion of components currently in the model. This is opposed to the constructive sparsity of the Kernel-RLS, which only controls the addition of new components by using the ALD criteria. Furthermore, since no kernel-trick is used, this method is not limited to only use kernel basis functions and thus arbitrary functions can be incorporated.

Note that in (D.2) only the last l targets are modeled with the weights and basis functions at the current time i. This can be seen as a batch learning problem with the last l samples as training data. However, there is a time dependency between the sliding window batches in terms of the set of basis functions. This is due to the way we add and delete basis functions. The current set of basis functions depends on the previous set and is only adapted to fit the current batch of training data $\boldsymbol{t}_i$ at time i. This mechanism will be later explained in detail.

D.2 Adaptive deletion and inclusion of basis functions

In Chapter 3 we have presented a simple criteria to decide whether to keep or delete a basis function from a given model corresponding to an $\hat{\alpha}_m^{[\infty]}$ of finite or infinite value respectively. Furthermore, as we have presented in Section 5.1, we can detect if a new candidate basis function should be added to the current model or not. In the following we will adapt both methods to our online learning problem and summarize the main results.

D.2.1 Testing basis functions currently in the model

To decide if the mth basis function corresponding to the mth column in $\boldsymbol{\Phi}_i$ at time i should be kept in or pruned from the model we need to compute the FV-SBL stationary point $\hat{\alpha}_{i,m}^{[\infty]}$ (see Section 3.2.2, Theorem 1) given as

$$\hat{\alpha}_{i,m}^{[\infty]} = \begin{cases} (\rho_{i,m}^2 - \varsigma_{i,m})^{-1}, & \rho_{i,m}^2 > \varsigma_{i,m} \\ \infty, & \rho_{i,m}^2 \leq \varsigma_{i,m}, \end{cases} \tag{D.3}$$

with

$$\begin{aligned} \varsigma_{i,m} &= (\hat{\tau}_i \boldsymbol{\varphi}_{i,m}^T \boldsymbol{\varphi}_{i,m} - \hat{\tau}_i^2 \boldsymbol{\varphi}_{i,m}^T \boldsymbol{\Phi}_{i,\bar{m}} \hat{\boldsymbol{\Sigma}}_{i,\bar{m}} \boldsymbol{\Phi}_{i,\bar{m}}^T \boldsymbol{\varphi}_{i,m})^{-1} \\ \text{and} \quad \rho_{i,m} &= \hat{\tau}_i \varsigma_{i,m} \boldsymbol{\varphi}_{i,m}^T \boldsymbol{t}_i - \hat{\tau}_i^2 \varsigma_{i,m} \boldsymbol{\varphi}_{i,m}^T \boldsymbol{\Phi}_{i,\bar{m}} \hat{\boldsymbol{\Sigma}}_{i,\bar{m}} \boldsymbol{\Phi}_{i,\bar{m}}^T \boldsymbol{t}_i, \end{aligned} \tag{D.4}$$

where the particular variables are basically defined as in Chapter 3 but are now applied to the sliding window block of data at time i. Note that based on the notation in Section 3.3.1, we now have $\hat{\boldsymbol{\Sigma}}_{i,\bar{m}} = (\hat{\tau}_i \boldsymbol{\Phi}_{i,\bar{m}}^T \boldsymbol{\Phi}_{i,\bar{m}} + \text{diag}(\hat{\boldsymbol{\alpha}}_{i,\bar{m}}))^{-1} = \left[\hat{\boldsymbol{\Sigma}}_i - \frac{\hat{\boldsymbol{\Sigma}}_i \boldsymbol{e}_m \boldsymbol{e}_m^T \hat{\boldsymbol{\Sigma}}_i}{\boldsymbol{e}_m^T \hat{\boldsymbol{\Sigma}}_i \boldsymbol{e}_m}\right]_{\bar{m}\bar{m}}$, where $\boldsymbol{\Phi}_{i,\bar{m}}$ is a matrix obtained from $\boldsymbol{\Phi}_i$ by removing the mth basis vector $\boldsymbol{\varphi}_{i,m}$ and $\hat{\boldsymbol{\alpha}}_{i,\bar{m}} = [\hat{\boldsymbol{\alpha}}_i]_{\bar{m}}$ is the hyperparameter vector $\hat{\boldsymbol{\alpha}}_i$ at time i without the element $\hat{\alpha}_{i,m}$. We summarize the procedure for testing a particular basis function m in Algorithm D.1. Note that in Algorithm D.1 we use $\hat{\alpha}_{i,m}^{\text{old}}$ to denote the value of $\hat{\alpha}_{i,m}$ before it is updated according to (D.3).

Algorithm D.1 Testing the mth basis $\boldsymbol{\varphi}_{i,m}$ at time i

Compute $\varsigma_{i,m}$ and $\rho_{i,m}$ from (D.4)
if $\rho_{i,m}^2 > \varsigma_{i,m}$ **then**
  % keep basis
  Compute: $\hat{\alpha}_{i,m} = \hat{\alpha}_{i,m}^{[\infty]}$ from (D.3)
  Update: $\hat{\boldsymbol{\Sigma}}_i = \hat{\boldsymbol{\Sigma}}_i - \frac{\hat{\boldsymbol{\Sigma}}_i \boldsymbol{e}_m \boldsymbol{e}_m^T \hat{\boldsymbol{\Sigma}}_i}{(\hat{\alpha}_{i,m} - \hat{\alpha}_{i,m}^{\text{old}})^{-1} + \boldsymbol{e}_m^T \hat{\boldsymbol{\Sigma}}_i \boldsymbol{e}_m}$, cf. (3.24) in Chapter 3
else
  % prune basis
  Compute: $\hat{\boldsymbol{\Sigma}}_i = \hat{\boldsymbol{\Sigma}}_{i,\bar{m}}$, $\hat{\boldsymbol{\alpha}}_i = \hat{\boldsymbol{\alpha}}_{i,\bar{m}}$, $\boldsymbol{\Phi}_i = \boldsymbol{\Phi}_{i,\bar{m}}$, $M_i = M_i - 1$
end if

D.2.2 Testing basis functions not included in the model

To test a new candidate basis function M_i+1 corresponding to a new column $\boldsymbol{\varphi}_{i,M_i+1}$ added to the design matrix at time i, i.e., $[\boldsymbol{\Phi}_i,\ \boldsymbol{\varphi}_{i,M_i+1}]$, we need to compute the FV-SBL stationary point $\hat{\alpha}_{i,M_i+1}^{[\infty]}$ from (D.3) with

$$\begin{aligned} \varsigma_{i,M_i+1} &= (\hat{\tau}_i \boldsymbol{\varphi}_{i,M_i+1}^T \boldsymbol{\varphi}_{i,M_i+1} - \hat{\tau}_i^2 \boldsymbol{\varphi}_{i,M_i+1}^T \boldsymbol{\Phi}_i \hat{\boldsymbol{\Sigma}}_i \boldsymbol{\Phi}_i^T \boldsymbol{\varphi}_{i,M_i+1})^{-1} \\ \text{and} \quad \rho_{i,M_i+1} &= \hat{\tau}_i \varsigma_{i,M_i+1} \boldsymbol{\varphi}_{i,M_i+1}^T \boldsymbol{t}_i - \hat{\tau}_i^2 \varsigma_{i,M+1} \boldsymbol{\varphi}_{i,M_i+1}^T \boldsymbol{\Phi}_i \hat{\boldsymbol{\Sigma}}_i \boldsymbol{\Phi}_i^T \boldsymbol{t}_i. \end{aligned} \tag{D.5}$$

This results are adopted from Section 5.1, Chapter 5. The resulting algorithm is presented in Algorithm D.2, were $\hat{\boldsymbol{\Sigma}}_{i,M_i+1}$ is given as

$$\begin{bmatrix} \hat{\boldsymbol{\Sigma}}_i + \lambda_{i,M_i+1}\hat{\tau}_i^2\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\Phi}_i^T\boldsymbol{\varphi}_{i,M_i+1}\boldsymbol{\varphi}_{i,M_i+1}^T\boldsymbol{\Phi}_i\hat{\boldsymbol{\Sigma}}_i & -\lambda_{i,M_i+1}\hat{\tau}_i\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\Phi}_i^T\boldsymbol{\varphi}_{i,M_i+1} \\ -\lambda_{i,M_i+1}\hat{\tau}_i\boldsymbol{\varphi}_{i,M_i+1}^T\boldsymbol{\Phi}_i\hat{\boldsymbol{\Sigma}}_i & \lambda_{i,M_i+1} \end{bmatrix} \tag{D.6}$$

with $\lambda_{i,M_i+1} = (\hat{\tau}_i\boldsymbol{\varphi}_{i,M_i+1}^T\boldsymbol{\varphi}_{i,M_i+1} + \hat{\alpha}_{i,M_i+1}^{[\infty]} - \hat{\tau}_i^2\boldsymbol{\varphi}_{i,M_i+1}^T\boldsymbol{\Phi}_i\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\Phi}_i^T\boldsymbol{\varphi}_{i,M_i+1})^{-1}$ $= (\varsigma_{i,M_i+1}^{-1} + \hat{\alpha}_{i,M_i+1}^{[\infty]})^{-1}$. Note that for $\rho_{i,M_i+1}^2 > \varsigma_{i,M_i+1}$, i.e., $\hat{\alpha}_{i,M_i+1}^{[\infty]}$ from (D.3) has a finite value, we simply obtain $\lambda_{i,M_i+1} = \varsigma_{i,M_i+1} - \varsigma_{i,M_i+1}^2/\rho_{i,M_i+1}^2$.

Algorithm D.2 Testing and adding or rejecting a new basis $\boldsymbol{\varphi}_{i,M_i+1}$

Compute ς_{i,M_i+1} and ρ_{i,M_i+1} from (D.5)
if $\rho_{i,M_i+1}^2 > \varsigma_{i,M_i+1}$ **then**
  % add new basis
  Compute $\hat{\alpha}_{i,M_i+1}$ from (D.3)
  Update $\hat{\boldsymbol{\Sigma}}_i = \hat{\boldsymbol{\Sigma}}_{i,M+1}$ from (D.6)
  Update $\boldsymbol{\Phi}_i = [\boldsymbol{\Phi}_i, \boldsymbol{\varphi}_{i,M_i+1}]$, $\hat{\boldsymbol{\alpha}}_i = [\hat{\boldsymbol{\alpha}}_i^T, \hat{\alpha}_{i,M_i+1}]^T$, $M_i = M_i + 1$
else
  % reject new basis - no action needed
end if

D.3 Sliding window online learning

Now we are ready to combine the two key elements of FV-SBL, namely pruning and adding basis functions, with the online sliding window idea discussed before. We denote the resulting method as the sliding window fast variational sparse Bayesian learning (SW-FV-SBL) algorithm. Algorithm D.3 presents the implementation details of SW-FV-SBL.

In the initial phase of Algorithm D.3, the sliding window length l is assumed to grow until $l = L$, where L is the final sliding window length. Computing $\boldsymbol{\Phi}_i$ and $\boldsymbol{t}_i$ during the algorithm depends on the current window length l and thus the number of rows in both terms also grows until $l = L$. The method starts with an arbitrary basis function ψ^* evaluated at the first input sample $\boldsymbol{x}_1$ in the design matrix. The other variables are then initialized according to Algorithm D.3. In some cases it is better not to start the re-estimation of the noise precision $\hat{\tau}_i$ at the very beginning. We have observed that especially at the initial phase where basis functions are rare, the noise precision estimate performs poorly and a delayed re-estimation start could give better results. At each iteration, all basis functions in the model are tested, then

Algorithm D.3 Sliding-window (SW) online learning algorithm

Initialize $\Phi_1 = \psi^*(\boldsymbol{x}_1)$ with some basis function $\psi^*(\cdot)$
Set $\hat{\tau}_1$ to an initial value, $i = 1$, $l = 1$, $M_1 = 1$, $\hat{\alpha}_{i,1} = 0$
Compute $\hat{\Sigma}_1 = (\hat{\tau}_1 \Phi_1^2)^{-1}$ and $\hat{\mu}_1 = \hat{\tau}_1 \hat{\Sigma}_1 \Phi_1 t_1 = \Phi_1^{-1} t_1$
for i = 2, 3, ... **do**
  Update $\hat{\tau}_i = \frac{l+2c}{\|\boldsymbol{t}_i - \boldsymbol{\Phi}_{i-1}\hat{\boldsymbol{\mu}}_{i-1}\|^2 + \mathrm{tr}\left(\hat{\boldsymbol{\Sigma}}_{i-1}\boldsymbol{\Phi}_{i-1}^T\boldsymbol{\Phi}_{i-1}\right)+2d}$, cf. (2.44), Chapter 2
  if $l < L$ **then**
    $l = l + 1$ *% grow sliding window (SW) length until* $l = L$
  end if
  Update $\boldsymbol{\Phi}_i$ with the current SW data using the basis functions at time $i-1$
  Update $\boldsymbol{t}_i$ with current SW data
  Set $\hat{\boldsymbol{\alpha}}_i = \hat{\boldsymbol{\alpha}}_{i-1}$
  Compute $\hat{\boldsymbol{\Sigma}}_i = (\hat{\tau}_i \boldsymbol{\Phi}_i^T \boldsymbol{\Phi}_i + \mathrm{diag}(\hat{\boldsymbol{\alpha}}_i))^{-1}$, cf. (2.40), Chapter 2

  % Test all current basis functions
  Run Algorithm D.1, $\forall m = 1, \ldots, M_i$

  % Test a new candidate basis function $\psi_{i,M_i+1}(\cdot)$
  Run Algorithm D.2 with candidate basis $\boldsymbol{\varphi}_{i,M_i+1}$

  Compute the mean at time i as $\hat{\boldsymbol{\mu}}_i = \hat{\tau}_i \hat{\boldsymbol{\Sigma}}_i \boldsymbol{\Phi}_i^T \boldsymbol{t}_i$, cf (2.40), Chapter 2
end for

updated or pruned according to Algorithm D.1. A new candidate basis function is tested, then kept or rejected according to Algorithm D.2. As given in [35] (see also V-SBL in Section 2.2.2), the model prediction at time i for a test input sample $\boldsymbol{x}^*$ can be computed as

$$y_i^* = \boldsymbol{\phi}_i(\boldsymbol{x}^*)^T \hat{\boldsymbol{\mu}}_i, \tag{D.7}$$

with $\boldsymbol{\phi}_i(\boldsymbol{x}) = [\psi_{i,1}(\boldsymbol{x}), \ldots, \psi_{i,M_i}(\boldsymbol{x})]^T$, the vector of all basis functions currently in the model evaluated at the input $\boldsymbol{x}$.

D.4 Simulation

In this section, we evaluate the performance of the proposed algorithm on one-step-ahead prediction of Mackey-Glass chaotic time series, where we have generated the data as described in [17, Section 2.11.1]. Mackey-Glass has been extensively used for benchmarking different time-series prediction algorithms [126, 127]. In one-step-ahead time-series prediction, the training (and test) data $\{\boldsymbol{x}_n, t_n\}_{n=1}^i$ can be generated with $\boldsymbol{x}_n = [t_{n-D}, \ldots, t_{n-2}, t_{n-1}]^T$, where each element in $\boldsymbol{x}_n$ is a sample from the time-series with additive white Gaussian noise having variance σ^2. For

the simulations we have used an input dimension of $D = 7$ and a noise variance $\sigma^2 = 10^{-3}$. We compare the simulation results with a Kernel-RLS [74], which uses ALD as a constructive sparsification criterion. That is, the criteria restrains the model from growing too much but never reduces its complexity. As basis functions $\psi_{i,m}(\cdot)$, we use the same Gaussian kernels as for Kernel-RLS. These are defined as $\kappa(\boldsymbol{x}, \boldsymbol{x}_j) = \exp\{-||\boldsymbol{x} - \boldsymbol{x}_j||^2\}$. Similarly to the Kernel-RLS, we define the candidate basis vector in Algorithm D.3 at time i as $\boldsymbol{\varphi}_{i,M_i+1} = [\kappa(\boldsymbol{x}_{i-l+1}, \boldsymbol{x}_i), \ldots, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)]^T$ and the initial basis function as $\psi^*(\cdot) = \kappa(\cdot, \boldsymbol{x}_1)$. For the noise precision estimate $\hat{\tau}$, as in Algorithm D.3, we consider a non-informative prior with $c = d = 0$. For the initialization we set $\hat{\tau}_1 = 10^5$.

For the ALD criterion in the Kernel-RLS a threshold parameter needs to be specified [74]. This parameter, which we denote as ALD threshold ν, qualitatively adjusts the amount of sparsity in Kernel-RLS. That is, large ALD thresholds lead to more sparsity whereas smaller lead to less sparsity. In comparison to Kernel-RLS, our method has no need for such a parameter. However, in our case, since we work with blocks of windowed data, a window length L has to be chosen. Another difference from the Kernel-RLS is that the SW-FV-SBL method provides an estimator for the noise precision $\hat{\tau}_i$, where in the Kernel RLS no such estimator exists. An exemplary learning curve resulting in the same sparsity of 14 kernels is presented in Fig. D.1. A sliding window length of $L = 300$ for the SW-FV-SBL and an ALD threshold $\nu = 0.3$ for the Kernel-RLS was used. In Fig. D.2 we see a comparison of both methods for different parameter settings. One can see that for a particular estimated sparsity, the Kernel-RLS has a higher test mean square error (MSE) than our proposed method. Additionally, for a specific test MSE performance, SW-FV-SBL leads to a sparser model, i.e., is uses less kernels than Kernel-RLS.

D.5 Conclusions

In this appendix, we have presented a sliding window fast variational sparse Bayesian learning (SW-FV-SBL) algorithm for online learning and nonlinear filtering. The method exploits fast variational SBL, introduced in Chapter 3, to determine the set of relevant basis functions from a sliding window block of data. Based on the FV-SBL pruning criterion, basis functions are consecutively added and removed from the model. By performing this addition and deletion at each time for the current block of data, the model is able to adapt to changes in the data over time. However, as the set of basis functions is updated based on the previous time instant, the

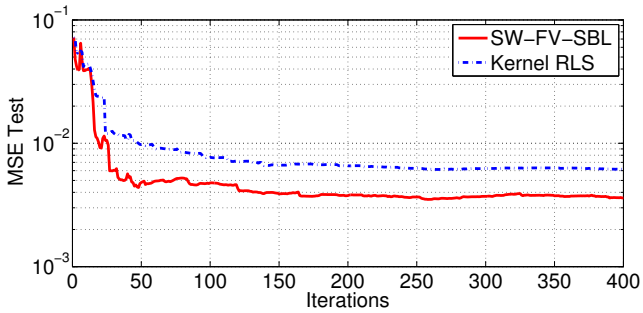


Figure D.1: Example learning curves of the SW-FV-SBL algorithm using a sliding window length of $L = 300$ and the Kernel-RLS using an ALD threshold of $\nu = 0.3$. Both methods result in the same number of 14 kernels at the last iteration. For the MSE, a test set of 200 samples was used.

model benefits from older choices of basis functions and thus incorporates some sort of memory in the set of basis functions.

The simulation results for SW-FV-SBL on Mackey-Glass chaotic time-series prediction with kernel basis functions showed better performance compared to a state of the art Kernel-RLS. More specifically, for different settings of the ALD threshold and sliding window length of the Kernel-RLS and the SW-FV-SBL algorithm, respectively, SW-FV-SBL uses less basis functions at a smaller achieved test MSE performance over a wide region as depicted in Figure D.2.

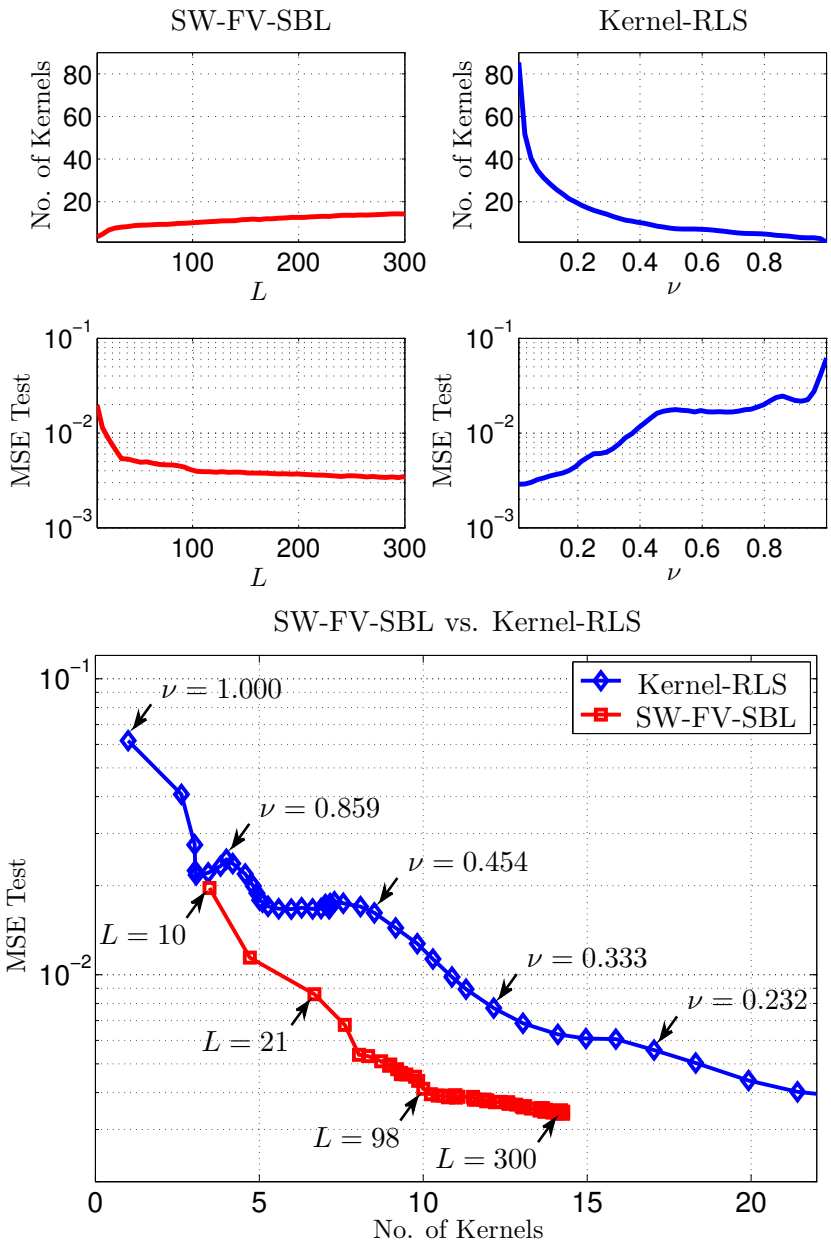


Figure D.2: Comparison of the SW-FV-SBL algorithm with an ALD Kernel-RLS for one-step ahead prediction of Mackey-Glass data with sliding window length L and ALD threshold ν. Results are averaged over 200 independent realizations with data 500 samples. For the MSE, a test set of 200 samples was used.

Bibliography

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

[2] I. Fischer-Bruns, "MetWatch - Datenquellen im Internet," `http://www.metwatch.de/index.htm?daten.htm`, Aug 2007, [Online; accessed 16-Aug-2011].

[3] Central Institution for Meteorology and Geodynamics ZAMG Austria, "Atmospheric pressure at sea level," `http://www.zamg.ac.at/cms/de/wetter/wetterkarte`, [Online; accessed 03-Dec-2012].

[4] Christopher M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, New York, NY, 2006.

[5] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck, et al., *Discrete-time signal processing*, vol. 2, Prentice Hall Englewood Cliffs, NJ, 1989.

[6] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo, *Introduction to wavelets and wavelet transforms: A primer*, vol. 23, Prentice Hall Upper Saddle River, NJ, 1998.

[7] Jooyoung Park and Irwin W Sandberg, "Universal approximation using radial-basis-function networks," *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[8] C. M. Bishop et al., *Neural networks for pattern recognition*, Clarendon press Oxford, 1995.

[9] J. Honerkamp and J. Weese, "Tikhonov's regularization method for ill-posed problems," *Continuum Mechanics and Thermodynamics*, vol. 2, no. 1, pp. 17–30, 1990.

[10] Jerome H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[11] Bernhard Schölkopf and Alexander J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, USA, 2001.

[12] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc*, vol. 68, no. 3, pp. 337–404, 1950.

[13] HaQuang Minh, Partha Niyogi, and Yuan Yao, "Mercer's theorem, feature maps, and smoothing," in *Learning Theory*, Gábor Lugosi and Hans Ulrich Simon, Eds., vol. 4005 of *Lecture Notes in Computer Science*, pp. 154–168. Springer Berlin Heidelberg, 2006.

[14] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*, Cambridge university press, New York, NY, 2004.

[15] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *The Journal of Machine Learning Research*, vol. 2, pp. 67–93, 2001.

[16] B. Hammer and K. Gersmann, "A note on the universal approximation capability of support vector machines," *Neural Processing Letters*, vol. 17, no. 1, pp. 43–53, 2003.

[17] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering*, Wiley Publishing, Hoboken, NJ, 2010.

[18] V. Vapnik, *The nature of statistical learning theory*, Springer, New York, NY, 1999.

[19] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press, New York, NY, 2000.

[20] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, Cambridge, Massachusetts and London, England, 2009.

[21] M. Jordan, Ed., *Learning in Graphical Models (Adaptive Computation and Machine Learning)*, A Bradford Book, Cambridge, Massachusetts, Nov 1998.

[22] G. L. Bretthorst, "An introduction to parameter estimation using Bayesian probability theory," in *Maximum Entropy and Bayesian Methods*, P. F. Fougère, Ed., pp. 53–79. Kluwer Academic Press, Dordrecht, 1990.

[23] Micheal E. Tipping, "The relevance vector machine," in *Advances in Neural Information Processing Systems 12 (NIPS)*, Todd K. Leen Sara A. Solla and Klaus-Robert Müller, Eds., pp. 652–658. MIT Press, Cambridge, MA, 2000.

[24] D. P. Wipf and B. D. Rao, "Sparse Bayesian learning for basis selection," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2153–2164, Aug 2004.

[25] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998.

[26] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21 –30, March 2008.

[27] S. F. Cotter and B. D. Rao, "Sparse channel estimation via matching pursuit with application to equalization," *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 374–377, 2002.

[28] S. D. Babacan, R. Molina, and A. K. Katsaggelos, "Bayesian compressive sensing using Laplace priors," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 53 –63, Jan 2010.

[29] David Paul Wipf, *Bayesian Methods for Finding Sparse Representations*, Ph.D. thesis, University of California, San Diego, 2006.

[30] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM journal on computing*, vol. 24, no. 2, pp. 227–234, 1995.

[31] Robert Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[32] A. F. M. Smith and G. O. Roberts, "Bayesian computation via the Gibbs sampler and related Markov chain monte carlo methods," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 3–23, 1993.

[33] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence, 1999 (UAI'99)*, Kathryn Laskey and Henri Prade, Eds., Stockholm, Sweden, 1999, pp. 467–475, Morgan Kaufmann, San Francisco, CA.

[34] Matthew J. Beal, *Variational Algorithms for Approximate Bayesian Inference*, Ph.D. thesis, University College London, 2003.

[35] C. M. Bishop and M. E. Tipping, "Variational relevance vector machines," in *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence, 2000 (UAI'00)*, Craig Boutilier and Moises Goldszmidt, Eds., Stanford, CA, 2000, pp. 46–53, Morgan Kaufmann, San Francisco, CA.

[36] C. S. Raghavendra, K. M. Sivalingam, and T. Znati, Eds., *Wireless sensor networks*, Springer, New York, NY, 2006.

[37] F. L. Lewis, "Wireless sensor networks," in *Smart Environments: Technologies, Protocols, and Applications*, Albert Y. Zomaya, Ed., pp. 11–46. John Wiley & Sons, Hoboken, NJ, 2004.

[38] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug 2002.

[39] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 2001 (ICASSP'01)*, Salt Lake City, Utah, 2001, IEEE, vol. 4, pp. 2033–2036.

[40] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448–1453, Dec 2002.

[41] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *IEEE Computer*, vol. 37, no. 8, pp. 50–56, Aug 2004.

[42] C. Otto, A. Milenkovic, C. Sanders, and E. Jovanov, "System architecture of a wireless body area sensor network for ubiquitous health monitoring," *Journal of Mobile Multimedia*, vol. 1, no. 4, pp. 307–326, 2006.

[43] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proceedings of the 2nd international conference on embedded networked sensor systems*, Baltimore, MD, 2004, ACM, pp. 13–24.

[44] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, and M. Nixon, "WirelessHART: Applying wireless technology in real-time industrial process control," in *IEEE Real-Time and Embedded Technology and Applications Symposium, 2008 (RTAS'08)*, St. Louis, MO, 2008, pp. 377–386.

[45] H. Lhermet, C. Condemine, M. Plissonnier, R. Salot, P. Audebert, and M. Rosset, "Efficient power management circuit: From thermal energy harvesting to above-ic microbattery energy storage," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 246 –255, Jan 2008.

[46] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56 – 69, Jul 2006.

[47] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad hoc networks*, vol. 3, no. 3, pp. 325–349, 2005.

[48] S. Kumar, Feng Zhao, and D. Shepherd, "Collaborative signal and information processing in microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 13 –14, March 2002.

[49] J. Chou, D. Petrovic, and K. Ramachandran, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, San Francisco, CA, 2003, vol. 2, pp. 1054–1062.

[50] A. Swami, Q. Zhao, Y.-W. Hong, and L. Tong, Eds., *Wireless Sensor Networks: Signal Processing and Communications*, John Wiley & Sons, Chichester, West Sussex, 2007.

[51] Y. Xu and H. Qi, "Distributed computing paradigms for collaborative signal and information processing in sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, no. 8, pp. 945–959, 2004.

[52] Joel Benjamin Predd, *Topics in Distributed Inference*, Ph.D. thesis, Princeton University, New Jersey, USA, 2006.

[53] Yuanqing Lin, *"l1-norm Sparse Bayesian Learning: Theory and Applications"*, Ph.D. thesis, University of Pennsylvania, 2008.

[54] Joaquin Quiñonero Candela, *Learning with Uncertainty Gaussian Processes and Relevance Vector Machines*, Ph.D. thesis, Technical University of Denmark, Lyngby, Denmark, 2004.

[55] C. E. Rasmussen and Quinonero-Candela J., "Healing the relevance vector machine through augmentation," in *Proceedings of the 22nd international conference on machine learning*, Bonn, Germany, 2005, pp. 689–696.

[56] Carl E. Rasmussen and Christopher Williams, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006.

[57] D. G. Tzikas, A. C. Likas, and N. P. Galatsanos, "Sparse Bayesian modeling with adaptive kernel learning," *IEEE Transactions on Neural Networks*, vol. 20, pp. 926–937, 2009.

[58] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: An efficient framework for modelling sensor network data," in *Third International Symposium on Information Processing in Sensor Networks, 2004 (IPSN'04)*, Berkeley, CA, 2004, pp. 1–10.

[59] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 798–808, 2005.

[60] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Third International Symposium on Infomation Processing in Sensor Networks, 2004 (IPSN'04)*, Berkeley, CA, 2004, pp. 20–27.

[61] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Regression in sensor networks: Training distributively with alternating projections," in *Proc. SPIE Conf. Advanced Signal Processing Algorithms, Architectures, and Implementations XV*, San Diego, CA, 2005.

[62] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed kernel regression: An algorithm for training collaboratively," in *IEEE Information Theory Workshop, 2006 (ITW'06)*, Punta del Este, Uruguay, 2006, pp. 332–336.

[63] Dongbing Gu and Zongyao Wang, "Distributed regression over sensor networks: An support vector machine approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008 (IROS 2008)*, Nice, France, Sept 2008, pp. 3286 –3291.

[64] P. Honeine, M. Essoloh, C. Richard, and H. Snoussi, "Distributed regression in sensor networks with a reduced-order kernel model," in *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008*, New Orleans, LA, Dec 2008, pp. 1–5.

[65] Timothy Battisti, *Distributed Field Estimation in Wireless Sensor Networks*, Ph.D. thesis, Sapienza University, Rome, Italy, 2010.

[66] Günter Reise, *Distributed Field Reconstruction in Wireless Sensor Networks Based on Hybrid Shift-Invariant Spaces*, Ph.D. thesis, Technical University of Vienna, Austria, 2011.

[67] C. C. Moallemi and B. Van Roy, "Consensus propagation," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4753 –4766, Nov 2006.

[68] M. E. Tipping and A. C. Faul, "Fast marginal likelihood maximisation for sparse Bayesian models," in *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, Key West, FL, Jan 2003.

[69] D. Shutin, T. Buchgraber, S. R. Kulkarni, and H. V. Poor, "Fast variational sparse Bayesian learning with automatic relevance determination for superimposed signals," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6257–6261, Dec 2011.

[70] D. Shutin and T. Buchgraber, "Trading approximation quality versus sparsity within incremental automatic relevance determination frameworks," in *IEEE International Workshop on Machine Learning for Signal Processing 2012 (MLSP'12)*, Santander, Spain, 2012, pp. 1–6.

[71] T. Buchgraber and D. Shutin, "Distributed variational sparse Bayesian learning for sensor networks," in *IEEE International Workshop on Machine Learning for Signal Processing 2012 (MLSP'12)*, Santander, Spain, 2012, pp. 1–6, [Best Student Paper Award].

[72] T. Buchgraber and D. Shutin, "Sparse Bayesian consensus-based distributed field estimation," in *5th International Conference on Signal Processing and*

Communication Systems, 2011 (ICSPCS'11), Honolulu, Hawaii, Dec 2011, pp. 1–8.

[73] T. Buchgraber, D. Shutin, and H. V. Poor, "A sliding-window online fast variational sparse Bayesian learning algorithm," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2011 (ICASSP'11)*, Prague, Czech Republic, May 2011, IEEE, pp. 2128–2131.

[74] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug 2004.

[75] Stephen P. Brooks, "Markov chain monte carlo method and its application," *Journal of the Royal Statistical Society*, vol. 47, no. 1, pp. 69–100, 1998.

[76] D. G. Tzikas, A. C. Likas, and N. P. Galatsanos, "The variational approximation for Bayesian inference," *IEEE Signal Processing Magazine*, vol. 25, no. 6, pp. 131–146, Nov 2008.

[77] Thomas Minka, "Expectation propagation for approximate Bayesian inference," in *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence, 2001 (UAI'01)*, John Breese and Daphne Koller, Eds., Seattle, WA, 2001, pp. 362–369, Morgan Kaufmann, San Francisco, CA.

[78] H. Sagan, *Introduction to the Calculus of Variations*, Courier Dover Publications, Mineola, New York, 1992.

[79] Thomas Minka, "Divergence measures and message passing," Tech. Rep. MSR-TR-2005-173, Microsoft Research Cambridge, 2005.

[80] P. K. Giri and J. Bannerjee, *Introduction to Statistics*, Academic Publishers, Kolkata, India, 2001.

[81] Stephen Boyd and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, England, Mar 2004.

[82] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[83] Geoffrey J. McLachlan and Thriyambakam Krishnan, *The EM Algorithm and Extensions (Wiley Series in Probability and Statistics)*, John Wiley & Sons, Hoboken, NJ, 2nd edition, Mar 2008.

[84] Matthew J. Beal and Zoubin Ghahramani, "The variational Bayesian EM algorithm for incomplete data: With application to scoring graphical model structures," *Statistics*, vol. 7, no. 7, pp. 1–10, 2003.

[85] Michael E. Tipping, "Bayesian inference: An introduction to principles and practice in machine learning," in *Advanced Lectures on Machine Learning*, Olivier Bousquet, Ulrike Luxburg, and Gunnar Rätsch, Eds., vol. 3176 of *Lecture Notes in Computer Science*, pp. 41–62. Springer Berlin Heidelberg, 2004.

[86] J. L. Johnson, *Probability and Statistics for Computer Science*, John Wiley & Sons, Hoboken, NJ, 2008.

[87] Harold Jeffreys, "An invariant form for the prior probability in estimation problems," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 186, no. 1007, pp. 453–461, 1946.

[88] G. E. P. Box and G. C. Tiao, *Bayesian inference in statistical analysis*, Wiley classics library. John Wiley & Sons, Hoboken, NJ, 1992.

[89] J. M. Bernardo, "Reference analysis," in *Handbook of statistics*, Dipak Dey and C. R. Rao, Eds., vol. 25 of *Bayesian Thinking, Modeling and Computation*, pp. 17–90. Elsevier, Amsterdam, Netherlands, 2005.

[90] J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*, Wiley Series in Probability and Statistics. John Wiley & Sons, Sussex, England, 2009.

[91] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*, McGraw-Hill Series in Electrical and Computer Engineering. McGraw-Hill, New York, NY, 4th edition, 2002.

[92] B. D. Rao, K. Engan, S. F. Cotter, J. Palmer, and K. Kreutz-Delgado, "Subset selection in noise based on diversity measure minimization," *IEEE Transactions on Signal Processing*, vol. 51, no. 3, pp. 760–770, Mar 2003.

[93] David J. C. MacKay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.

[94] David Wipf, Jason Palmer, and Bhaskar Rao, "Perspectives on sparse Bayesian learning," in *Advances in Neural Information Processing Systems 16 (NIPS)*, Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, Eds., pp. 249–256. MIT Press, Cambridge, MA, 2004.

[95] M. A. T. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1150–1159, Sep 2003.

[96] David Wipf and Srikantan Nagarajan, "A new view of automatic relevance determination," in *Advances in Neural Information Processing Systems 20 (NIPS)*, J.C. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., pp. 1625–1632. MIT Press, Cambridge, MA, 2008.

[97] D. P. Wipf, B. D. Rao, and S. Nagarajan, "Latent variable Bayesian models for promoting sparsity," *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 6236–6255, Sep 2011.

[98] B. D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," *IEEE Transactions on Signal Processing*, vol. 47, no. 1, pp. 187–200, Jan 1999.

[99] Jason Palmer, David Wipf, Kenneth Kreutz-Delgado, and Bhaskar Rao, "Variational EM algorithms for non-Gaussian latent variable models," in *Advances in Neural Information Processing Systems 18 (NIPS)*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., pp. 1059–1066. MIT Press, Cambridge, MA, 2006.

[100] David Wipf and Srikantan Nagarajan, "Sparse estimation using general likelihoods and non-factorial priors," in *Advances in Neural Information Processing Systems 22 (NIPS)*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds., pp. 2071–2079. MIT Press, Cambridge, MA, 2009.

[101] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[102] John Winn and Christopher M. Bishop, "Variational message passing," *Journal of Machine Learning Research*, vol. 6, pp. 661–694, Dec 2005.

[103] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, Jun 2006.

[104] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Fourth International Symposium on Information Processing in Sensor Networks, 2005 (IPSN 2005)*, Los Angeles, CA, Apr 2005, pp. 63–70.

[105] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498 –519, Feb 2001.

[106] Dmitry M. Malioutov, Jason K. Johnson, and Alan S. Willsky, "Walk-sums and belief propagation in gaussian graphical models," *Journal of Machine Learning Research (JMLR)*, vol. 7, pp. 2031–2064, Dec 2006.

[107] Yair Weiss and William T. Freeman, "Correctness of belief propagation in gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, pp. 2173–2200, Oct 2001.

[108] J. E. Elson, *Time synchronization in wireless sensor networks*, Ph.D. thesis, University of California, Los Angeles, 2003.

[109] V. Schwarz and G. Matz, "Distributed averaging in wireless sensor networks under an aloha-like communication protocol," in *Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, CA, 2010, IEEE, pp. 1918–1922.

[110] Jaakko Luttinen, Alexander Ilin, and Tapani Raiko, "Transformations for variational factor analysis to speed up learning," in *Proceedings of the 17th European Symposium on Artificial Neural Networks, 2009 (ESANN 2009)*, Bruges, Belgium, Apr 2009.

[111] Y. Qi and T. S. Jaakkola, "Parameter expanded variational Bayesian methods," in *Advances in Neural Information Processing Systems 19 (NIPS)*, J. Platt B. Schölkopf and T. Hoffman, Eds., pp. 1097–1104. MIT Press, Cambridge, MA, 2007.

[112] D. Shutin and B. H. Fleury, "Sparse variational Bayesian SAGE algorithm with application to the estimation of multipath wireless channels," *IEEE Transactions on Signal Processing*, vol. 59, no. 8, pp. 3609 –3623, Aug 2011.

[113] G. H. Golub and C. F Van Loan, *Matrix Computations, 3rd ed.*, The Johns Hopkins University Press, Baltimore and London, 1996.

[114] Steven H. Strogatz, *Nonlinear Dynamics And Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, Studies in Nonlinearity. Westview Press, Perseus Books Group, Cambridge, MA, Dec 2000.

[115] Dmitriy Shutin, Sanjeev R. Kulkarni, and H. Vincent Poor, "A novel view of incremental sparse Bayesian learning with automatic relevance determination for parametric regression," *IEEE Transactions on Signal Processing, submitted*, Oct 2011, [in peer review].

[116] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," `http://www2.imm.dtu.dk/pubdb/p.php?3274`, October 2008, Version 20081110, [Online; accessed Feb-2009].

[117] D. Shutin, T. Buchgraber, S. R. Kulkarni, and H. V. Poor, "Fast adaptive variational sparse Bayesian learning with automatic relevance determination," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2011 (ICASSP'11)*, Prague, Czech Republic, May 2011, IEEE, pp. 2180–2183.

[118] I. C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete Research*, vol. 28, no. 12, pp. 1797–1808, 1998.

[119] A. Frank and A. Asuncion, "UCI machine learning repository," `http://archive.ics.uci.edu/ml/`, 2010, [Online; accessed Oct-2010].

[120] P. M. Djuric and Yunlong Wang, "Distributed Bayesian learning in multiagent systems: Improving our understanding of its capabilities and limitations," *IEEE Signal Processing Magazine*, vol. 29, no. 2, pp. 65–76s, Mar 2012.

[121] Reza Olfati-Saber, Elisa Franco, Emilio Frazzoli, and Jeff S. Shamma, "Belief consensus and distributed hypothesis testing in sensor networks," in *Proceedings of the Network Embedded Sensing and Control Workshop 2005 (NESC'05)*, P. J. Antsaklis and P. Tabuada, Eds., Notre Dame, Indiana, 2006, vol. 331 of *Lecture Notes in Control and Information Sciences*, pp. 169–182, Springer Verlag, Berlin Heidelberg.

[122] M. Lazaro-Gredilla, S. Van Vaerenbergh, and I. Santamaria, "A Bayesian approach to tracking with kernel recursive least-squares," in *Proceedings of*

the IEEE International Workshop on Machine Learning for Signal Processing, 2011 (MLSP'11), Beijing, China, Sep 2011, pp. 1–6.

[123] Jason K. Johnson, Danny Bickson, and Danny Dolev, "Fixing convergence of Gaussian belief propagation," in *Proceedings of the International Symposium on Information Theory 2009 (ISIT'09)*, Seoul, Korea, Jun 2009, IEEE, pp. 1674–1678.

[124] Carl D. Meyer, Ed., *Matrix analysis and applied linear algebra*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[125] Simon Haykin, *Adaptive Filter Theory (4th Edition)*, Prentice Hall, Upper Saddle River, NJ, Sep 2001.

[126] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Physical Review Letters*, vol. 8, no. 59, pp. 845–848, 1987.

[127] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, Apr 2004.