Diss. ETH No. 16170

A Factor Graph Approach to Signal Modelling, System Identification and Filtering

A dissertation submitted to the
Swiss Federal Institute of Technology, Zürich
for the degree of
Doctor of Technical Sciences

presented by

Sascha Korl

Dipl. Ing., TU Graz
born on August 8, 1974
citizen of the Republic of Austria

accepted on the recommendation of
Prof. Dr. Hans-Andrea Loeliger, examiner
Dr. Stefan Launer, co-examiner
Prof. Dr. Allen G. Lindgren, co-examiner

Hartung-Gorre Verlag, Konstanz, July 2005

Series in Signal and Information Processing **Vol. 15**

Editor: Hans-Andrea Loeliger

Bibliographic Information published by Die Deutsche Bibliothek

Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the internet at http://dnb.ddb.de.

First Edition 2005

HARTUNG-GORRE VERLAG KONSTANZ

ISSN 1616-671X
ISBN 3-86628-032-7

You are worthy, our Lord and God,
to receive glory and honour and power,
for you created all things, and by your will
they were created and have their being.
(The Bible, Revelation 4,11)

Seite Leer /
Blank leaf

Acknowledgements

I was very fortunate to have the opportunity to work under the guidance of my advisor Andi Loeliger. I very much enjoyed the working environment he provided, the freedom where my creativity could unfold. I will remember his excitement whensoever I reported a new finding about factor graphs ... He also changed my science vocabulary. Statements like 'That's impossible' he persistantly transmuted into 'I don't know how to do that—now'.

It is an honour to work with someone like Allen Lindgren, one of my co-advisors. His wealth of experience is stunning. He could point out important things even when I completely lost orientation amidst a pile of equations. Thank you for visiting Zurich and sharing your wide know-how with me.

I am indebted to my co-advisor Stefan Launer. He managed it fetching me back to earth when I tended to float into academic heights. As a researcher in industry he tortured me with questions like 'Can you give an example?' or 'Does that bring us closer to a working solution?'. I am also very grateful for his financial and logistic support.

I consider working at ISI to be a rewarding experience. My appreciation goes to every single person hanging around at this place. Especially my project partner and office neighbour Markus Hofbauer, Volker Koch for his unselfish 'Moin Chef!' and Justin Dauwels for all the creative "coffee breaks" where many of the ideas of this thesis were born (and many more were burried).

A warm thanks goes to the research and signal processing people at Phonak, especially Silvia Allegro and Hans-Ueli Röck. During my rare visits I always felt very welcome and I got the chance to experience a whiff of engineering reality. I vividly remember the situation, where I asked for a 1024 point FFT on the chip and got a 64 point one.

And last but not least my thanks are due to my dear family. My mother and father always gave me their loving support no matter what way in life I took. I'll never forget that day during my thesis when a very special person said yes to a very special question I asked her. I thank my fiancée Barbara for her love and encouragement.

Abstract

This thesis concerns model-based signal processing. In model-based signal processing a class of signals is described by a stochastic state-space model, in general with unknown parameters. The aim of signal estimation (filtering, denoising, parameter estimation, etc.) is to determine the 'best' signal (e.g. the most probable signal) out of that class given a set of observations.

Just a few years ago model-based signal processing was still limited to a restricted class of models. There are models with finite state-space (hidden Markov models) on the one hand and there are linear Gaussian state-space models (Kalman filter and related algorithms) on the other hand. For a multitude of applications those model types are insufficient.

Factor graphs open up new possibilities in this matter. First, factor graphs (and similar graphical models) provide a framework for the systematic and consistent derivation of classic model-based algorithms. Second, factor graphs permit and encourage the combination of different classic approaches for complex models with many unknown parameters; and third, factor graphs provide a framework for the systematic development of completely new algorithms.

A factor graph is used to represent the factorisation of the probability density function of the signal model. Inference is performed by passing messages along the edges of the graph. Messages can be interpreted as summaries of subgraphs, therefore the inference algorithm is called summary-propagation algorithm. We derive different message types justified by different representations of such summaries.

In a first step all messages in the graph are Gaussian. Several classic algorithms can be solely represented by Gaussian messages: Kalman filtering and smoothing, linear prediction and recursive least squares adaptive filters.

The more interesting and also more complicated case is when messages appear which are not Gaussian. Different techniques are proposed in this thesis to deal with such messages. For example, complicated messages can be represented with lists of samples of the exact message, which leads to particle-filter-type algorithms, or as gradients of the exact message, which leads to gradient descent (or hill climbing) methods.

The expectation maximisation (EM) algorithm is a powerful parameter estimation algorithm which has been used by many people in different applications. In this thesis we show how the EM algorithm can be stated as message passing on a factor graph. A simple message update rule is given, which allows the development of reusable building blocks. In varying the message update schedule, different new variants of the EM algorithm can be devised. Finally, a local message update rule arising from the combination of the ideas of summary-propagation and EM is given.

The utilisation of the proposed techniques is demonstrated by means of the autoregressive model with unknown coefficients, unknown input noise variance and unknown observation noise variance.

Keywords: Graphical models, factor graphs, summary-propagation algorithm, belief propagation, message passing, expectation maximisation, signal modelling, system identification, autoregressive model, Kalman filter.

Kurzfassung

Die vorliegende Arbeit befasst sich mit modellbasierter Signalverarbeitung. Bei der modellbasierten Signalverarbeitung wird eine Klasse von Signalen durch ein stochastisches Zustandsraummodell, im Allgemeinen mit unbekannten Parametern, beschrieben. Das Ziel der Signalschätzung (Filterung, Entrauschung, Parameterschätzung, etc.) ist es das 'beste' Signal aus der modellierten Signalklasse zu ermitteln, wobei alle verfügbaren Beobachtungen berücksichtigt werden sollten.

Noch vor wenigen Jahren war die modellbasierte Signalverarbeitung auf wenige Typen von Modellen beschränkt. Im Wesentlichen gab es einerseits Modelle mit endlichem Zustandsraum (Hidden Markov Modelle) und andererseits lineare Modelle mit Gauss'schem Rauschen (Kalmanfilter und verwandte Algorithmen). Für eine Vielzahl von Anwendungen sind diese Modelltypen aber nicht ausreichend.

Faktorgraphen eröffnen hier neue Horizonte. Erstens bieten Faktorgraphen (und ähnliche graphische Modelle) einen Rahmen für die systematische und einheitliche Herleitung klassischer modellbasierter Signalverarbeitungs-Algorithmen; zweitens ermöglichen und ermuntern Faktorgraphen zur kombinierten Verwendung verschiedener klassischer Verfahren für komplexe Modelle mit vielen unbekannten Parametern und drittens bieten Faktorgraphen einen Rahmen zur systematischen Entwicklung gänzlich neuer Verfahren.

Ein Faktorgraph wird verwendet um eine Faktorisierung der Wahrscheinlichkeitsdichtefunktion des Signalmodells zu präsentieren. Statistische Inferenz erfolgt durch das Versenden von Nachrichten entlang der Kanten des Graphen. Diese Nachrichten können als Zusammenfassungen

von Teilen des Graphen gesehen werden, daher der Name Summary-Propagation-Algorithmus. Es werden verschiedene Nachrichtentypen abgeleitet, welche sich durch verschiedene Repräsentationen dieser Zusammenfassungen ergeben.

In einer ersten Betrachtung sind alle Nachrichten gaussförmig. Der Kalmanfilter und -smoother, lineare Prädiktion und recursive least squares adaptive Filter können mit ausschliesslich gaussförmigen Nachrichten dargestellt werden.

Der interessantere und kompliziertere Fall tritt ein, wenn Nachrichten entstehen, die nicht mehr als Gauss-Funktion dargestellt werden können. Verschiedene Methoden mit solchen Nachrichten umzugehen werden in dieser Arbeit behandelt. Zum Beispiel können komplizierte Nachrichten als Liste von Samples der exakten Nachricht dargestellt werden, was zu Particle-Filter Algorithmen führt. Oder der Gradient wird zur Darstellung der Nachricht herangezogen, was zu gradient descent (oder hill climbing) Methoden führt.

Der Expectation-Maximisation-Algorithmus (EM) ist ein leistungsfähiger Algorithmus zur Parameterschätzung und wird von Vielen in verschiedenen Anwendungen eingesetzt. In dieser Dissertation wird gezeigt, wie der EM-Algorithmus als Message-Passing am Faktorgraphen dargestellt werden kann. Eine einfache Nachrichten-Aufdatierungsregel erlaubt die Entwicklung von wiederverwendbaren Bausteinen. Neue Varianten des EM-Algorithmus erhält man durch Variation der Reihenfolge der Nachrichten-Aktualisierung. Zum Schluss wird eine lokale Nachrichten-Aufdatierungsregel angegeben, welche Ideen von EM und Summary-Propagation kombiniert.

Die Anwendung der vorgestellten Methoden wird anhand des autoregressiven Modells mit unbekannten Koeffizienten, unbekanntem Eingangs- und Beobachtungsrauschen demonstiert.

Stichworte: Graphische Modelle, Faktorgraphen, Summary-Propagation Algorithms, Belief Propagation, Message Passing, Expectation Maximisation, Signalmodellierung, System-Indentifikation, Autoregressives Modell, Kalmanfilter.

Contents

Chapter 1

Introduction

1.1 Motivation

The primary motivation for this work has been the design of a speech enhancement algorithm for hearing aids. The reduction of speech intelligibility through interfering background noise has been a long-standing nuisance for hearing aid users (as well as for speech recognition programs and other applications).

This dissertation, rather than proposing yet another speech enhancement algorithm, deals instead with the general mathematical framework for the development of such algorithms.

All signal processing algorithms presented in this thesis rely on a mathematical description (i.e. the model) of the signals under consideration; therefore, our approach falls into the category of model-based signal processing. The signal models are formulated as probability-density functions (cf. Section 3.1).

Given the model and some observations, e.g. some measurements of a noisy speech signal, we aim at estimating some parameters or unknown variables. Filtering, denoising, parameter estimation etc. may be viewed as estimating some unkown variables in a model. Unfortunately, the estimation of variables in a complex model may necessitate multidimensional

integration which is either computational too expensive or even cannot be solved in closed form. For this reason only a limited class of models are practical, examples include hidden Markov models (HMM) for finite state-spaces or the Kalman filter algorithm and related algorithms for linear Gaussian state-space models.

The present dissertation proposes a graphical approach to signal modelling and estimation. Factor graphs are used to represent the structure of the problem at hand. Variables of the system are represented by edges and relation between variables are represented by nodes of the graph. This representation is similar to signal flow diagrams, which signal processing engineers are used to. Statistical inference is performed by passing messages along the edges. Different algorithms are obtained by choosing different representation of the messages or different message update schedules. Using message passing algorithms on factor graphs opens up new possibilities to solve the estimation problem in more complex models as has been possible till now.

Although the attempt of tackling the problem of speech enhancement from a new perspective has so far not led to a spectacular new enhancement algorithm, some nice advances could be achieved (detailed in Section 1.2). Of scientific interest are the findings about inference techniques in graphical models whereas some newly derived algorithms for parameter estimation are also of practical interest.

1.2 Contributions

Some of the contributions of this thesis are listed below:

- We have adapted and extended the theory and application of factor graphs to build continuous and mixed density models. Factor graphs were introduced in [41, 74]. We build upon this material and show in detail how time series, such as speech signals, can be modelled with factor graphs.

- In [83] it has been shown, that the Kalman filter recursion can be derived from the corresponding factor graph. In the same manner we have written out many classic signal processing and estimation

algorithms, such as many variants of the Kalman filter and Kalman smoother, adaptive filter algorithms or particle filters.

- Through the graphical representation, altered or extended versions of the algorithms have been derived in this thesis to enhance performance or reduce computational complexity of the existing algorithms.
- In the wake of the algorithm developments we present a set of building blocks from which algorithm designers can choose to develop signal processing algorithms that fit their needs without going through every detail themselves.
- A well known and often used parameter estimation algorithm is expectation maximisation (EM). In this thesis, we show that EM-type algorithms can be derived within the same framework of factor graphs.
- A mixture form of the EM algorithm and the summary-propagation algorithm has been devised. In simulations the new scheme has shown slightly faster convergence in the first iterations than the standard EM algorithm.

1.3 Suitability for Practical Applications

One important condition for the developments in this thesis has been their industrial application for the practical design of speech enhancement algorithms. Here we summarise salient attributes of the proposed approach with regard to its use in practical applications:

- The factor graph approach offers a common platform for the design of complex signal processing and estimation algorithms.
- Due to the conceptually simple graphical representation it helps the algorithm designer to comprehend the problem easier and therefore shorten the development cycle.
- An object-oriented software toolbox, which has been developed for this thesis, contains many building blocks, which can be combined in a modular way to allow rapid prototyping.

- Once a model is expressed in the factor graph language, modifications and extensions are easily carried out, because of the modular structure of factor graphs.
- It is possible to integrate this approach into available design tools such as Matlab and/or Simulink.

1.4 Outline

This thesis is structured as follows: Chapter 2 gives a brief introduction to factor graphs and the summary-propagation algorithm focused on continuous variables. In Chapter 3 the factor graph of the autoregressive (AR) model is given together with some explanations how to derive estimation algorithms from the factor graph representation of the model.

The message update rules take an appealing simple form if the messages are Gaussian. The main application of such messages is Kalman filtering which is treated in Chapter 4. Everything that goes beyond Gaussian techniques is collected in Chapter 5.

In Chapter 6 most of the proposed techniques are demonstrated by means of the autoregressive model. Chapter 7, finally, concludes this dissertation.

Chapter 2

Introduction to Factor Graphs

This chapter aims at giving an introduction to factor graphs and the summary-propagation algorithm on a generic level. The introduction is based on [84, 134]; here we focus on continuous variables. The main idea of the graphical representation of the factorisation of functions is presented in Section 2.1. One of the most important operations that can be performed on factor graphs is marginalisation, which can be computed by message passing on the graph. This generic algorithm is called summary-propagation algorithm (SPA) and is introduced in Section 2.2.

Literature

A factor graph tutorial is [84]. The first comprehensive paper about factor graphs and the sum-product algorithm is [74]. Normal graphs (or Forney-style factor graphs) are introduced in [41]. Least squares and Kalman filtering on factor graphs has been shown in [83]. The book by Pearl about Bayesian networks is [110]. A comparison of different graphical models is given in [43].

The mathematical foundation for factor graphs is the generalised distributive law, which is treated in [1]; a comment on the iterative appli-

cation of Bayes' rule is given in [124]. Some different message representations are discussed in [40, 135]. Coonvergence and optimality proofs of graphical models are given in [136, 137]. Gaussian graphical models are treated in [111, 114, 123, 129]; Bayesian networks in [59, 63, 102]. There are some publications about approximative inference in graphical models: [138] (linear response), [103] (loopy belief propagation), [91] (Quasi-Bayes), [20] (likelihood-weighting). A different message passing algorithm is expectation propagation, which is treated in [58, 99, 101]. Learning in graphical models is discussed in [15, 16, 42, 44, 46, 56, 73, 87]. Hidden Markov models in conjunction with graphical models are shown in [120, 121]. The application of graphical models to speech is shown in [11, 115] and classification with graphical models is shown in [45].

2.1 Factor Graphs

A thorough and comprehensible tutorial on factor graphs is given by Loeliger [84]. Originally introduced in [74], Forney used a refined notation [41], which we will use in this thesis. We will refer to this notation as Forney-style factor graphs, or short FFG.

Factor graphs belong to the family of graphical models. A graphical model in general represents dependencies among variables by a graph. Other types of graphical models include Markov random fields [139] or Bayesian networks [110]. In principle, most techniques presented in this thesis could have been performed on another type of graphical model but we prefer factor graphs due to the reasons given in [84].

A factor graph represents a factorisation of a multivariate function which is demonstrated in the following example.

Example 2.1. (Factorisation of a global function)

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = f_A(x_1, x_2) f_B(x_3, x_4) f_C(x_2, x_4, x_5) f_D(x_5, x_6) \tag{2.1}$$

is represented by the graph in Fig. 2.1 where every factor (or local function) is represented by a node and every variable is represented by an edge. □

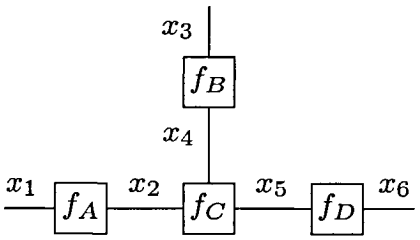


Figure 2.1: An example factor graph.

More formally a Forney-style factor graph (FFG) is defined as follows:

- **Factor graph:** An FFG represents a global function and consists of nodes and edges.
- **Nodes/local functions:** There is a node for every factor, also called local function.
- **Edges/variables:** There is an edge or half-edge for every variable. Half-edges are connected to one node only.
- **Connections:** An edge (or half-edge) representing some variable x is connected to a node representing some factor f if and only if f is a function of x.

Implicit in this definition is the assumption, that an edge is connected to no more than two nodes. This restriction is easily circumvented by introducing equality constraint nodes (cf. Section 4.2).

It should be emphasised that a factor graph can represent any multivariate function, not necessarily probability density functions. The application to probabilities is shown in Chapter 3.

2.2 Summary-Propagation Algorithm

A common task in model-based signal processing is the estimation of parameters of a stochastic model. In general Bayesian estimation [9,67], we are interested in marginal probabilities of the parameters (or any other variable, e.g. the hidden state of a state-space model).

Example 2.2. (Marginalisation of a factored function)
Coming back to Example 2.1, we might be interested in the marginal

$$f(x_5) = \int_{\mathcal{D}} \cdots \int f(x_1, x_2, x_3, x_4, x_5, x_6)\, dx_1 dx_2 dx_3 dx_4 dx_6 \tag{2.2}$$

with domain $\mathcal{D} = \mathbb{R}^5$. With the factorisation (2.1)

$$\begin{aligned} f(x_5) &= \int\int\int\int\int f_A(x_1, x_2) \cdot f_B(x_3, x_4) \cdot f_C(x_2, x_4, x_5) \cdot f_D(x_5, x_6) \\ &\qquad dx_1 dx_2 dx_3 dx_4 dx_6 \\ &= \underbrace{\int\int f_C(x_2, x_4, x_5) \underbrace{\int f_A(x_1, x_2) dx_1}_{\mu_{f_A \to x_2}(x_2)} \cdot \underbrace{\int f_B(x_3, x_4) dx_3}_{\mu_{f_B \to x_4}(x_4)} dx_2 dx_4 \cdot}_{\mu_{f_C \to x_5}(x_5)} \\ &\quad \underbrace{\int f_D(x_5, x_6) dx_6}_{\mu_{f_D \to x_5}(x_5)} . \end{aligned} \tag{2.3}$$

□

The trick is to pull the factors which do not depend on the integration out of the integral, e.g.

$$\int f(x) f(y) dx = f(y) \int f(x) dx. \tag{2.4}$$

For example, when integrating (2.3) w.r.t. x_6, we can pull out every factor except $f_D(x_5, x_6)$, which depends on x_6. Instead of solving a high dimensional integral, it suffices to solve simpler ones (one and two dimensional in our example). Intermediate terms, $\mu_{f \to x}(x)$, can be interpreted as messages flowing along the edges of the graph with the subscript indicating the direction of the messages. Their meaning becomes obvious when looking at Fig. 2.2. For example, $\mu_{f_A \to x_2}(x_2)$ as the result of the integral $\int f_A(x_1, x_2) dx_1$ can be interpreted as message coming out of node f_A towards edge x_2. If both $\mu_{f_A \to x_2}(x_2)$ and $\mu_{f_B \to x_4}(x_4)$ are available, $\mu_{f_C \to x_5}(x_5)$ can be computed as the output message of node f_C towards edge x_5. The final result of (2.3) is

$$f(x_5) = \mu_{f_C \to x_5}(x_5) \cdot \mu_{f_D \to x_5}(x_5) \tag{2.5}$$

and thus the product of the two messages along the same edge.

A different interpretation can be seen in Fig. 2.3. In this view, solving one intermediate integral in (2.3) means boxing the corresponding part of the graph. The details inside such a box are "integrated out"; only a summary is propagated (therefore the name summary-propagation). In the first step the dark shaded areas in Fig. 2.3 are boxed. Afterwards the lighter shaded box is closed, until we arrive at (2.5).

Half-edges (such as x_1) do not carry a message towards the connected node; alternatively, the edge may be thought of as carrying a message representing a neutral factor 1. With this in mind, every message (i.e. every intermediate result of (2.3)) is computed in the same way. Imagine the generic node depicted in Fig. 2.4. We want to compute the message towards edge y, given the messages impinging on all other edges $x_1, \ldots, x_N$:

Sum-product rule:

$$\mu_{f\to y}(y) = \int_{\mathcal{D}} \cdots \int f(y, x_1, \ldots, x_N) \cdot \mu_{x_1\to f}(x_1) \cdots \mu_{x_N\to f}(x_N)\, dx_1 \cdots dx_N \qquad (2.6)$$

In words: The message out of a factor node $f(\ldots)$ along edge y is the product of the function $f(\ldots)$ and all messages towards f along all other edges, summarised (integrated) over all variables except y. This is the **sum-product rule** and serves as the starting point for many of the node update derivations in this thesis. In general, messages are computed out of any edge; there is no preferential direction.

The integral operators in (2.3) and (2.6) can be replaced by any summary operator, e.g. the sum operator for discrete-valued variables or the max operator for computing maximisations. It can also be a more abstract summary operation, as we will see in Chapter 5.6, for instance.

Finally, the marginal $f(y)$ of a certain variable y is the multiplication of the two messages on the corresponding edge, such as in (2.5).

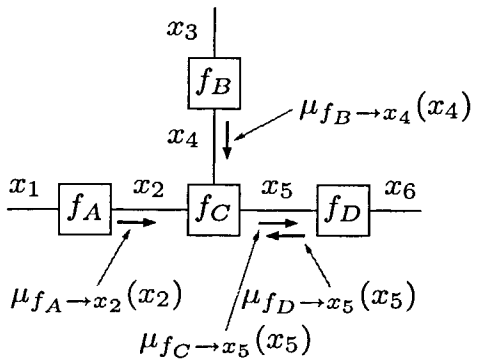


Figure 2.2: An example factor graph. Intermediate results of the marginalisation can be interpreted as messages flowing along the edges of the graph.

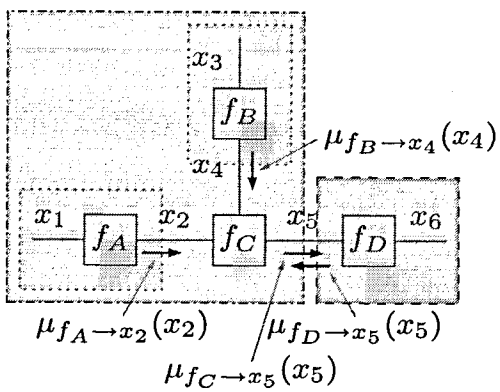


Figure 2.3: An example factor graph. The computation of the messages can also be interpreted as boxing the corresponding part of the graph.

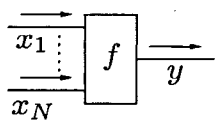


Figure 2.4: Message out of a generic node.

In general it is

$$f(y) = \mu_{\text{tot}}(y) = \mu_{f_A \to y}(y) \cdot \mu_{f_B \to y}(y) \tag{2.7}$$

where f_A and f_B are the two nodes attached to edge y. For half edges, the message coming from the open end carries a neutral factor "1". Thus, the message from the node towards the half-edge is already the marginal of the corresponding variable.

In its general form, the **summary-propagation algorithm (SPA)** computes two messages on every edge. For graphs without loops (polytrees) it is efficient to start the message computation from the leaves and proceed with nodes whose input messages become available. In this way, each message is computed exactly once. When the algorithm stops, exact marginals, such as (2.7), are available for all variables simultaneously.

A completely different situation arises, when the graph has loops. In this case, messages have to be updated recursively, since a new output message at one node can have influence on the inputs of the same node over another path through the graph. In this case, the SPA computes approximated marginals rather than exact marginals. Even worse, there is no guarantee that the algorithm converges. In practice, however, in many cases the algorithm reaches a stable point and the decisions based on the marginals are good enough.

In many applications only the mode or mean of the required marginal (2.7) is needed. This gives us the freedom to arbitrarily scale (2.3) and the individual terms therein. Consequently, (2.6) needs only be evaluated up to a scaling constant:

$$\begin{aligned} \mu_{f \to y}(y) \quad \propto \quad & \int_{\mathcal{D}} \cdots \int f(y, x_1, \ldots, x_N) \cdot \\ & \mu_{x_1 \to f}(x_1) \cdots \mu_{x_N \to f}(x_N)\, dx_1 \cdots dx_N \end{aligned} \tag{2.8}$$

Especially when working with probability distributions, it makes sense to normalise them after each node to prevent numerical underflows. Thus,

$$\mu_{f \to y}(y) = \frac{\tilde{\mu}_{f \to y}(y)}{\int_{\mathcal{D}_y} \tilde{\mu}_{f \to y}(y) dy} \tag{2.9}$$

where $\tilde{\mu}_{f \to y}(y)$ is the unnormalised message.

Seite Leer /
Blank leaf

Chapter 3

Factor Graph of the Autoregressive Model

We use the autoregressive (AR) model to illustrate the application of factor graphs for signal modelling, system identification and filtering. On the one hand the AR model is simple enough to develop an intuition about the resulting algorithms, and on the other hand it is complex enough to demonstrate the versatility of the factor graph approach.

In Section 3.1 the concept of a probabilistic model is reviewed. Section 3.2 aims at specifying the autoregressive model used and introduces the factor graph for this specific model, whereas in Section 3.3 the way of using factor graphs for estimation in general time-series is explained. Section 3.4 finally summarises the factor graph approach by means of the example of speech enhancement.

Literature

An introduction to the autoregressive model is given in [90]; The AR model in noise is treated in [14, 66]; the subspace approach in [24], the LMS algorithm in [122]. Multichannel AR models are shown in [53]. Deterministic signals in AR noise [68]; AR parameter estimation with missing observations [94,95]; RLS in state-space formulation [92,93]; Uni-

versal linear prediction [119]; Robust recursive AR [72]; Unknown noise statistics [17, 47, 78, 105, 142]; autoregressive moving average (ARMA) models: ML [62], Maximum a posteriori [28], Support vector method [113]; How to deal with coloured noise is shown in [50, 145]; Smoothing [31]; LPC speech spectrum time evolution [108]; Variance estimation in AR models is treated in [8, 75, 97, 98]; AR estimation with factor graphs [69, 86].

3.1 Probabilistic Modelling with Factor Graphs

A model in the probabilistic sense is described by a probability density function

$$p(\mathbf{z}, \mathbf{y}, \mathbf{x} \mid \boldsymbol{\theta}, \boldsymbol{\vartheta}). \tag{3.1}$$

The arguments of the function (3.1) can be classified according to three properties. First, we distinguish between variables, which are in front of the separator, and parameters, which are behind the separator. Second, we distinguish between known and unknown variables or parameters. Third, we distinguish between variables or parameters that are of interest and that are auxiliary (also called nuisance variables or hidden variables).

The specific arguments of (3.1) are classified as follows. The vector $\mathbf{z}$ includes all variables that are observed and therefore known, $\mathbf{y}$ includes unknown variables we are interested in, and $\mathbf{x}$ includes all unknown auxiliary variables. The parameters, either known or unknown, are divided into parameters of interest $\boldsymbol{\theta}$ and auxiliary parameters $\boldsymbol{\vartheta}$.

Many models in signal processing have structure; their probability density function can be factorised as will be shown in the following section by means of the autoregressive (AR) model.

3.2 System Model

An autoregressive (AR) process is defined as follows: Let $\{X_n, n \in \mathbb{Z}\}$[1] be a real-valued stochastic process defined by

$$X_n = \sum_{\ell=1}^{M} a_\ell \cdot X_{n-\ell} + U_n \tag{3.2}$$

with $a_\ell \in \mathbb{R}$ and where $\{U_n\}$ is white Gaussian noise with zero mean and variance σ_U^2. Commonly, all a_ℓ's are compiled into a vector

$$\mathbf{a} \triangleq (a_1, \ldots, a_M)^\mathsf{T}. \tag{3.3}$$

We observe the process $\{Z_n\}$ with

$$Z_n = X_n + W_n \tag{3.4}$$

where $\{W_n\}$ is white Gaussian noise with zero mean and variance σ_W^2. The complete parameter vector comprises $(\mathbf{a}, \sigma_U^2, \sigma_W^2)$. Depending on the problem at hand individual parameters are known or unknown. Throughout this thesis we assume the parameters are constant, except for Section 3.2.2, where the extension to time-varying parameters is considered.

It is convenient to write (3.2) and (3.4) in state-space form as

$$\mathbf{X}_n = \mathbf{A}\mathbf{X}_{n-1} + \mathbf{b}U_n \tag{3.5}$$

$$Y_n = \mathbf{c}^\mathsf{T}\mathbf{X}_n \tag{3.6}$$

$$Z_n = Y_n + W_n \tag{3.7}$$

with

$$\mathbf{X}_n \triangleq (X_n, \ldots, X_{n-M+1})^\mathsf{T} \tag{3.8}$$

$$\mathbf{b} \triangleq \mathbf{c} \triangleq (1, 0, \ldots, 0)^\mathsf{T} \tag{3.9}$$

$$\mathbf{A} \triangleq \left[\begin{array}{c|c} \mathbf{a}^\mathsf{T} & \\ \hline \mathbf{I} & \mathbf{0} \end{array}\right] \tag{3.10}$$

The $M \times M$ matrix $\mathbf{A}$ is the state transition matrix where $\mathbf{I}$ is the $(M-1)\times(M-1)$ identity matrix and $\mathbf{0}$ is a zero vector of dimension

[1] In the following we omit the specification of the index set and denote stochastic processes by $\{X_n\}$.

$M-1$. The state-space model defined by (3.5)-(3.10) has scalar input U_n and scalar output Y_n. This is not a real restriction; all algorithms in the following sections also work for vector input and vector output.

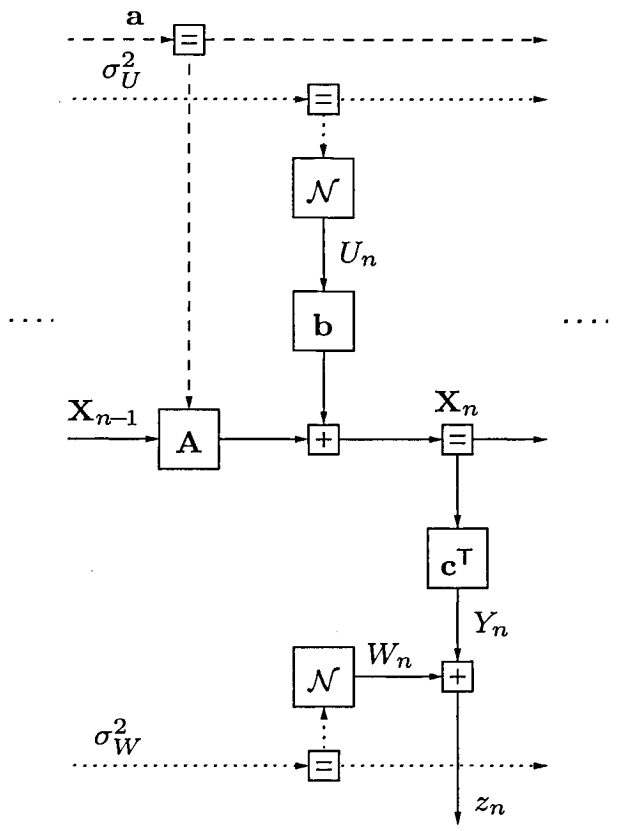


Figure 3.1: Factor graph corresponding to the state-space model defined by (3.5)-(3.10).

One section of the factor graph of this model is depicted in Fig. 3.1; the complete factor graph is built by concatenating such sections. The basic state-space model is represented by the part consisting of solid lines in the middle. The (unknown) AR coefficient vector $\mathbf{a}$ is represented by the dashed edges, and the dotted edges represent the variance of the innovation σ_U^2 and the variance of the observation noise σ_W^2, respectively. If any of these parameters is known, the corresponding part of the graph may be removed. Although sometimes it may be intended to show the dependence on a known parameter explicitely.

The arrows on the edges of the graph in Fig. 3.1 solely express causality relations. There is no fundamental difference whether the message flows in the direction of the arrow or against it.

The graph represents the probability density function

$$p(z_1,..,z_N,y_1,..,y_N,x_0,..,x_N,u_1,..,u_N,w_1,..,w_N \mid \mathbf{a},\sigma_U^2,\sigma_W^2) = p(\mathbf{x}_0)\prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1},u_n,\mathbf{a})p(u_n|\sigma_U^2)p(z_n|y_n,w_n)p(y_n|\mathbf{x}_n)p(w_n|\sigma_W^2). \tag{3.11}$$

In (3.11) the *parameters* $\mathbf{a}$, σ_U^2 and σ_W^2 are either known or unknown; there is no prior probability distribution assigned to them. The factor graph of Fig. 3.1 may also represent the joint density function

$$p(x_0,..,x_N,z_1,..,z_N,u_1,..,u_N,w_1,..,w_N,\mathbf{a},\sigma_U^2,\sigma_W^2) = p(\mathbf{x}_0)\prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1},u_n,\mathbf{a})p(\mathbf{a})p(u_n|\sigma_U^2)p(\sigma_U^2) p(z_n|y_n,w_n)p(y_n|\mathbf{x}_n)p(w_n|\sigma_W^2)p(\sigma_W^2) \tag{3.12}$$

where $\mathbf{a}$, σ_U^2 and σ_W^2 are *variables* having a prior distribution. In that case the factor graph is extended by additional nodes for each prior $p(\mathbf{a})$, $p(\sigma_U^2)$ and $p(\sigma_W^2)$ (not shown in Fig. 3.1).

3.2.1 Message update schedule

The summary propagation algorithm is performed by passing message along the edges of the factor graph of Fig. 3.1 as described in Section 2.2. Since the graph has cycles there is no general recipe to define an appropriate message update schedule.

Fig. 3.2 shows one possible update schedule. The figure shows one section only; the complete factor graph is build by concatenating many such sections. The summary propagation algorithm starts at the left-most section and updates the messages in the order given by the numbers in circles. It then proceeds with the next section traversing from left to right through the whole graph. Afterwards, the graph is traversed from right to left updating the messages in every section according to the numbers given in the right-hand graph of Fig. 3.2. These steps are iterated until convergence. We refer to this scheme as *iterative schedule.*

In contrast, the backward pass and the additional iterations may be skipped. In other words, the graph is only once traversed from left to

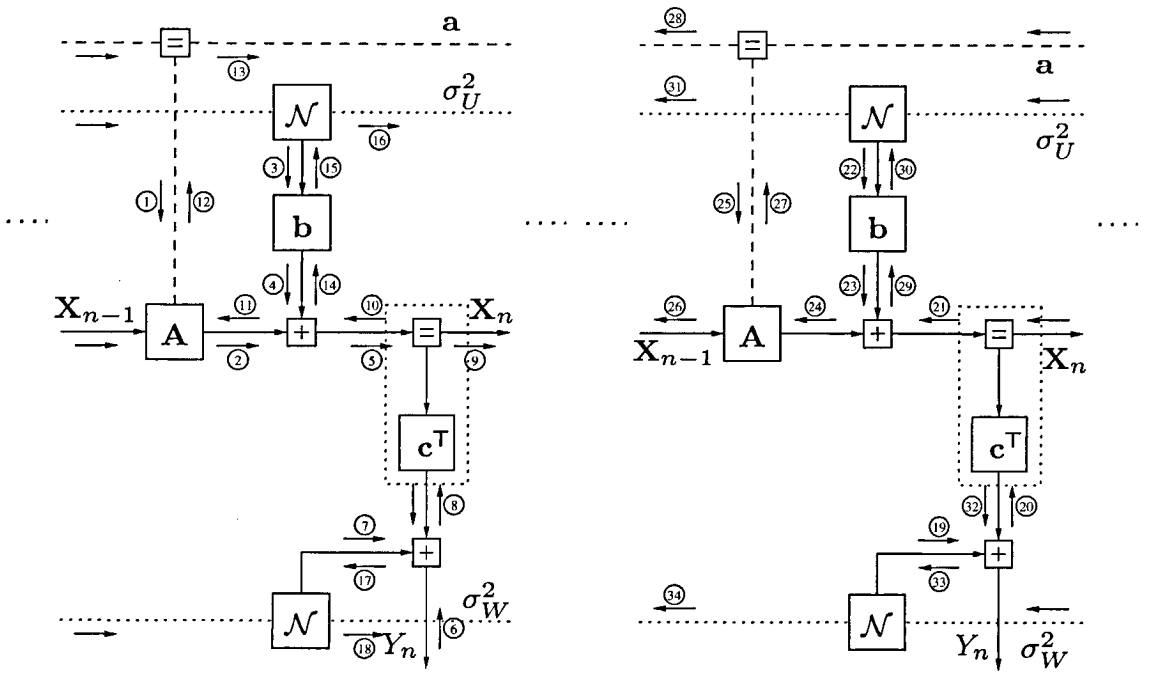


Figure 3.2: Example of a message update schedule. The message updates during the forward pass are shown in the left-hand graph, where the message updates during the backward pass are shown in the right-hand graph.

right without going backwards, which sometimes leads to acceptable results (cf. Chapter 6). This scheme is referred to as *forward-only schedule.*

3.2.2 Extension to time-varying parameters

The model of Fig. 3.1 can easily be extended to time-varying parameters by introducing additional nodes into the graph, which model the parameter variation. The altered graph is shown in Fig. 3.3. Note that the parameters $\mathbf{a}_n$, $(\sigma_U^2)_n$ and $(\sigma_W^2)_n$ are now time-dependent, thus the subscript n.

The variation in the parameters is modelled as random deviation between successive time-steps. The deviation is Gaussian with zero mean and variance $\alpha\mathbf{I}$ for the coefficient vector $\mathbf{a}$, variance β for the parameter σ_U^2 and variance γ for the parameter σ_W^2, respectively. This can be seen as process noise for the parameter process. In fact, the parameters are modelled themselves as state-space models and are computed by recursions equivalent to the Kalman filter.

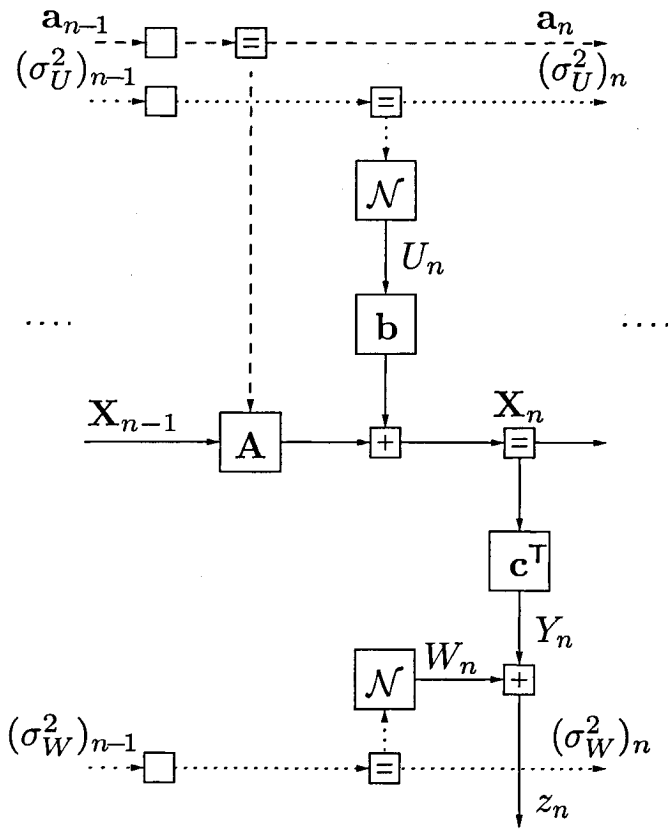


Figure 3.3: Factor graph for joint state/parameter estimation with time-varying parameters. Every parameter is itself modelled by a state-space model.

3.3 Estimation and Filtering

In this section a brief review of estimation in general is given (Section 3.3.1). In time-series estimation one can distinguish three different estimation tasks depending on what observations are available when deciding about the value of a parameter or variable. Estimation with time-series is elaborated on in Section 3.3.2.

3.3.1 Estimation

A common task in model-based signal processing is to estimate the value of a variable or parameter based on the model and some observations. In speech enhancement, for instance, we may be interested in the values of the clean speech signal $\mathbf{y}$ based on noisy measurements $\mathbf{z}$ of the speech signal and the model $p(\mathbf{z}, \mathbf{y}, \mathbf{x} \mid \boldsymbol{\theta}, \boldsymbol{\vartheta})$. We look for the signal $\mathbf{y}$ that best

explains the measurements $\mathbf{z}$, i.e. that maximises $p(\mathbf{z}, \mathbf{y}, \mathbf{x} \mid \boldsymbol{\theta}, \boldsymbol{\vartheta})$. But before we can maximise we have to get rid of the unknown variables $\mathbf{x}$ and unknown parameters $\boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$. This can be done in different ways; it is always a kind of summation operation. The generic summation operation is expressed by the integral sign here:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \int_{\mathbf{x}} \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\vartheta}} p(\mathbf{z}, \mathbf{y}, \mathbf{x} \mid \boldsymbol{\theta}, \boldsymbol{\vartheta}) \tag{3.13}$$

Solving (3.13) has traditionally been possible for a limited class of models only. Such models essentially include state-space models with discrete state-space (Hidden Markov Models, HMMs) or linear Gaussian models with continuous state-space (Kalman filter and related algorithms). For most practical applications, however, HMMs and Kalman filters are insufficient.

Factor graphs provide new opportunities in this context. As presented in Section 2.2 the summary-propagation algorithm computes exactly such summaries as needed in (3.13).

In the following we review the most important estimators. The *maximum a posteriori* or MAP-estimator is defined as

$$\hat{y}_{\text{MAP}} = \underset{y}{\operatorname{argmax}}\, p(y|z) = \underset{y}{\operatorname{argmax}}\, p(z, y) \tag{3.14}$$

since $p(y|z) = p(z, y)/p(z) \propto p(z, y)$ where z is observed. The marginal

$$p(z, y) = \int_{x} \int_{\theta} \int_{\vartheta} p(z, y, x \mid \theta, \vartheta) \tag{3.15}$$

is in our case computed by summary-propagation on the corresponding factor graph.

The *minimum mean squared error* of MMSE-estimator (Bayes estimator with quadratic cost function) is defined as

$$\hat{y}_{\text{MMSE}} = \mathsf{E}[Y|Z{=}z] = \int_{\mathcal{D}_y} y\, p(y|z)\, dy = \frac{1}{K} \int_{\mathcal{D}_y} y\, p(z, y)\, dy \tag{3.16}$$

where $K = \int p(z, y) dy$ is a proper normalisation constant and the marginal $p(z, y)$ is again computed by summary-propagation.

The *maximum likelihood* or ML-estimator is defined for parameter estimation only:

$$\hat{\theta}_{\mathrm{ML}} = \underset{\theta}{\operatorname{argmax}}\, p(z|\theta) \tag{3.17}$$

where

$$p(z|\theta) = \int_y \int_x \int_\vartheta p(z, y, x \mid \theta, \vartheta) \tag{3.18}$$

is computed by the summary-propagation algorithm.

In conclusion, the summary-propagation algorithm on the factor graph computes the needed marginals. The desired estimates are then extracted from these marginals, for instance as the mean or mode of the marginal.

3.3.2 Estimation of time-series

The signals modelled in this thesis are time-series. The special structure of time-series leads to factor graphs that have a repetitive structure. In Fig. 3.1, for example, one section of the factor graph of the autoregressive model is shown; the complete graph is made by concatenating many of such sections.

There are three different estimation tasks in time-series models depending on what variable we want to estimate and what observations are available at that time. Fig. 3.4 depicts the three cases schematically, where $x_1, \ldots, x_N$ represent the (hidden) state and $z_1, \ldots, z_N$ represent the observations. Assume we want to estimate the state x_n at time-instance n. If the observations lie solely in the past, we talk about *prediction*; we talk about *filtering*, if we observe the signal y up to time n, and about *smoothing* if observations from the future are available.

In the factor graph framework, all three cases are treated the same way. The arrows in Fig. 3.4 indicate where in the graph the messages need to be updated. In the case of prediction and filtering only the messages in the left part of the graph need to be updated.

In general, the signals we deal with are of infinite length. The usual way to deal with infinitely long signals is frame-processing. The signals are partitioned into frames of the same length and the estimation or

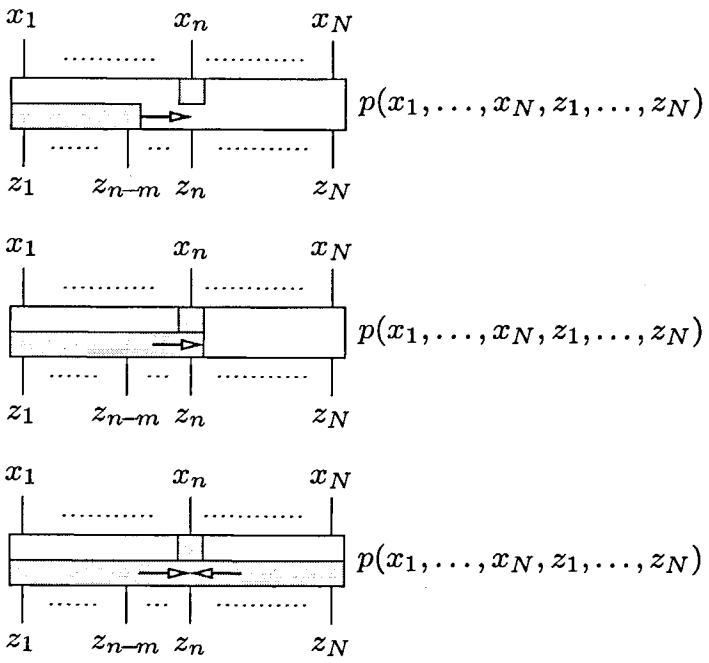


Figure 3.4: Prediction, filtering, smoothing. The arrows symbolise the information flow through the factor graph.

filtering algorithm is applied to every frame anew. The frames are usually overlapping to suppress boundary effects.

Of course it is possible to emulate this kind of frame-processing in the factor graph framework. But obviously, data about state and parameters in the overlapping part are discarded when building a completely new graph for every frame. Instead we can use the same graph with the messages already computed and extend it to include the new observations. In the same manner we can prune it on the side of old observations to keep it at equal length. In this approach, already computed messages need only be updated with information from the new observations instead of computing all messages from scratch.

A similar approach in the Bayesian network literature is called dynamic Bayesian networks [104, 146].

3.4 Example: Speech Enhancement

All concepts of the foregoing sections are demonstrated by means of an example. Speech signals are by nature very stochastic. So it is self-

evident to use a probabilistic approach to deal with such signals as for the problem of speech enhancement [26, 52]. We partition the design process for the speech enhancement algorithm into five design steps.

I. Generative model

The block diagram of Fig. 3.5 is used as a starting point for the development of a speech enhancement algorithm based on factor graphs. A

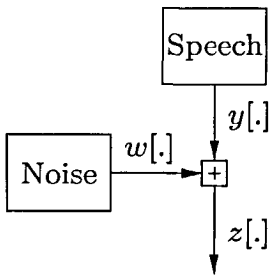


Figure 3.5: Block diagram of the speech enhancement problem as starting point for the development of an algorithm based on factor graphs.

speech signal $y[.]$ is corrupted by an unknown noise signal $w[.]$ through linear addition. Such a model is sometimes called generative model, since it explains the way the measured signal is generated. We do not know the exact way the speech and noise signals are generated, but we generally know some statistics about the signals. These statistics are captured by the corresponding boxes in Fig. 3.5.

II. Factor graph

To make the correlation over time explicit, a line is drawn for every output sample of the speech and noise model and its addition. This is depicted in Fig. 3.6. This diagram can already be seen as a factor graph. The boxes 'Speech' and 'Noise' are factor graph nodes which capture the statistics of the corresponding signals. The nodes 'Speech' and 'Noise' are of course structured themselves; such signals can be modelled by a state-space model for instance:

$$p(y_1, \ldots, y_N, x_0, \ldots, x_N) = p(x_0) \prod_{n=1}^{N} p(y_n, x_n | x_{n-1}) \tag{3.19}$$

where intermediate variables $x_0, \ldots, x_N$ for the (hidden) state are introduced. The factor graph of (3.19) is shown in Fig. 3.7. A detailed description of the model used in this thesis is given in Section 3.2.

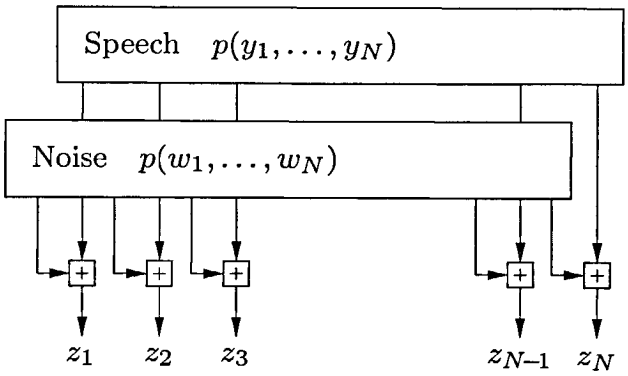


Figure 3.6: Unrolled block diagram of the speech enhancement problem, where every output sample is modelled explicitly.

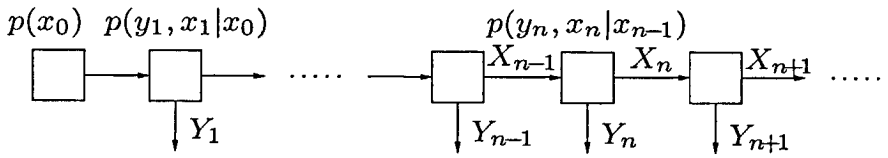


Figure 3.7: Factor graph of a state-space model given by (3.19).

A simple model for the noise is white Gaussian noise for instance.

$$p(w_1, \ldots, w_N) = \prod_{n=1}^{N} p(w_n) = \prod_{n=1}^{N} \mathcal{N}(w_n \mid 0, \sigma_W^2) \tag{3.20}$$

Since (3.20) comprises independent factors of one variable only, the corresponding factor graph is trivial and consists of separated nodes as shown in Fig. 3.8. Because the individual factors are independent and represent the same distribution such stochastic processes are called *independent and identically distributed* (i.i.d.).

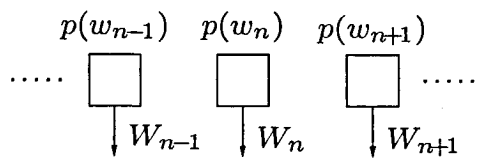


Figure 3.8: Factor graph of white Gaussian noise.

Putting together the speech model of Fig. 3.7 and the noise model of Fig. 3.8 we obtain the factor graph of Fig. 3.9.

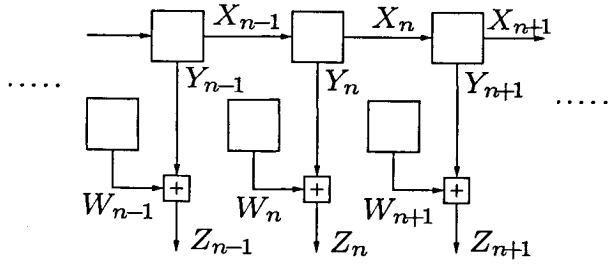


Figure 3.9: Factor graph of the model for speech enhancement.

III. Message update rules

The next step in designing a speech enhancement algorithm is to define how to compute the individual messages. This strongly depends on the specific structure of the nodes and is shown in detail in Section 2.2 in general and in Chapter 6 for some concrete examples. Once the update

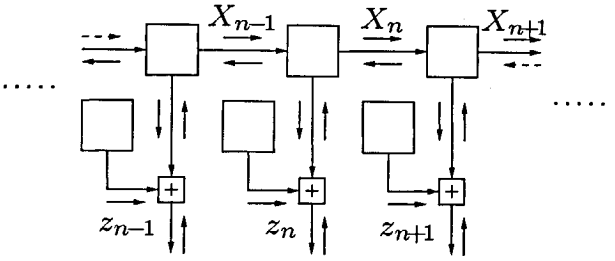


Figure 3.10: Factor graph of the model for speech enhancement with messages.

rule for every message is specified (Fig. 3.10) an update schedule should be defined. Thus, we have to define the order in which the messages are updated. Since most graphs have cycles, a message may be computed twice or more often.

IV. Message update schedule

The documentation of an update schedule is explained by means of Fig. 3.11. The graph consists of concatenated sections with the same structure. Hence, to specify the factor graph it suffices to show only one slice. The schedule is defined as described in Section 3.2.1. The outermost sections are either open (=half-edges carrying a neutral message) or terminated by a node, e.g. $p(x_0)$, representing an a-priori density.

V. Result

Depending on the task we are interested in different results. Here, we are

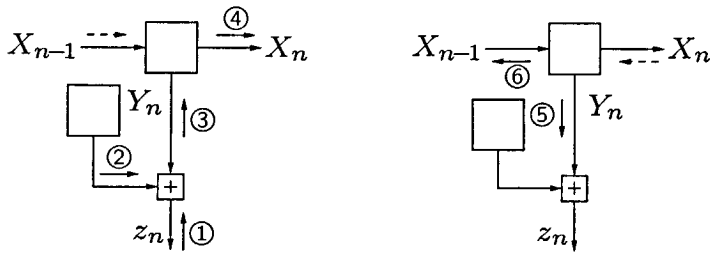


Figure 3.11: Factor graph of the model for speech enhancement with message update schedule. Left-hand side: forward-pass; right-hand side: backward pass.

not interest in any internal parameter of the model, but in an estimate of the clean speech signal. The individual marginals of this signal are available at the edges for $Y_1, \ldots, Y_N$. The a posteriori density (or its approximation, if the graph has cycles) at a single edge is the multiplication of the two messages flowing along this edge (μ_{tot}).

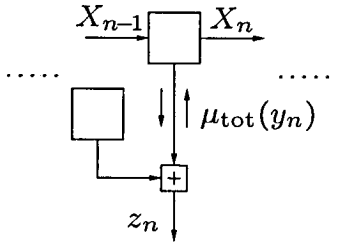


Figure 3.12: Reading off the results for the speech enhancement algorithm.

The MMSE estimate is obtained by the expected value of the corresponding variable given that density (cf. Section 3.3):

$$\hat{y}_{n,\text{MMSE}} = \mathsf{E}[Y_n | z_1, \ldots, z_N] = \int_{\mathcal{D}_{Y_n}} y_n \cdot \mu_{\text{tot}}(y_n)\, dy_n \tag{3.21}$$

Alternatively, the MAP estimate is obtained by taking the maximum:

$$\hat{y}_{n,\text{MAP}} = \underset{y_n}{\text{argmax}}\, \mu_{\text{tot}}(y_n) \tag{3.22}$$

3.5 Summary

In this chapter we have considered the following points:

- We have given a brief introduction into probabilistic modelling. The use of factor graphs has been shown by means of the autoregressive model in this context. The factor graph of the autoregressive model with unkown parameters has been explained.
- The concept of prediction, filtering and smoothing in time-series has been introduced together with the factor graph view.
- The process of designing algorithms with the help of factor graphs has been demonstrated by means of an example.

Seite Leer /
Blank leaf

Chapter 4

Gaussian Models: Kalman Filtering and Related Algorithms

In this chapter the application of factor graphs and the summary–propagation algorithm is restricted to Gaussian models, where all variables and messages are Gaussian distributed. In that case, the message update rules take on an appealing simple form.

Different representations of Gaussian messages are reviewed in Section 4.1. Using these representations update rules for often used building blocks are tabulated in Section 4.2. The most salient application of the tabulated update rules are Kalman filtering algorithms as shown in Section 4.3. With linear prediction (Section 4.4) and recursive least squares (Section 4.5) two further applications of Gaussian graphs are given. The chapter concludes with Section 4.6.

4.1 Representation of Gaussian messages

This section reviews different ways to represent Gaussian messages. Messages flow along edges and are functions of the variable corresponding to this edge. Most often, these functions can be specified by parameters, as in the case of Gaussian distributions. Several different parameterisations are possible (cf. Appendix A.1.1):

- From the basic definition of the n-dimensional real Gaussian distribution

$$\mu(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{W}|^{-1}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^\mathsf{T}\mathbf{W}(\mathbf{x}-\mathbf{m})\right) \quad (4.1)$$

$$= \mathcal{N}_W(\mathbf{x} \mid \mathbf{m}, \mathbf{W}) \quad (4.2)$$

  where $\mathbf{W}$ is a $n \times n$ positive semidefinite matrix and $\mathbf{m}$ a column vector of size n. $\mathbf{W}$ is called weight matrix. The distribution is fully specified by the pair $(\mathbf{m}, \mathbf{W})$. The notation $\mathcal{N}_W$ is used shorthand for this representation.

- If the inverse $\mathbf{V} = \mathbf{W}^{-1}$ exists, (4.1) can also be written in the following form

$$\mu(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^\mathsf{T}\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m})\right) \quad (4.3)$$

$$= \mathcal{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{V}) \quad (4.4)$$

  In this case, the distribution is fully specified by the pair $(\mathbf{m}, \mathbf{V})$ and $\mathbf{V}$ is called covariance matrix. The symbol $\mathcal{N}$ is used for this representation.

- Often it is not necessary to compute $\mathbf{m}$ explicitly; if suffices to use the weighted mean $\boldsymbol{\xi} \triangleq \mathbf{W}\mathbf{m}$ as a parameter vector. The message is therefore

$$\mu(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}\boldsymbol{\xi}^\mathsf{T}\mathbf{W}^{-1}\boldsymbol{\xi}\right)}{\sqrt{(2\pi)^n |\mathbf{W}|^{-1}}} \exp\left(-\frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{W}\mathbf{x} + \mathbf{x}^\mathsf{T}\boldsymbol{\xi}\right) \quad (4.5)$$

$$= \mathcal{N}_{\xi,W}(\mathbf{x} \mid \boldsymbol{\xi}, \mathbf{W}) \quad (4.6)$$

  Hence, also the pair $(\boldsymbol{\xi}, \mathbf{W})$ can fully specify a Gaussian distribution; in that case the symbol $\mathcal{N}_{\xi,W}$ is used.

In most practical cases, the weight matrix $\mathbf{W}$ or the covariance matrix $\mathbf{V}$ have a special structure. Hence, it may be advantageous to utilise special matrix representations, such as the square-root form [54] or making use of the displacement structure [64]. Although this has not been used in this thesis, a considerable reduction in computational complexity of the devised algorithms may be obtained by consistently capitalising on the structure of the matrices during computation [40,135].

4.2 Update Rules for Linear Building Blocks

Gaussian distributions have a beneficial property. If the operands of a linear operation are Gaussian distributed the distribution of the result is again in the same family. Equivalently, if the input messages to a linear function node is Gaussian, the output message is Gaussian as well. Therefore, the whole message update operation can be described by means of the parameters of the messages, e.g. mean and covariance matrix.

A summary of those update rules can be found in Table 4.1 on page 32. The detailed derivation is given in [83].

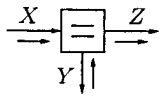
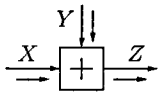
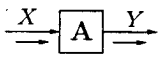
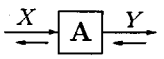
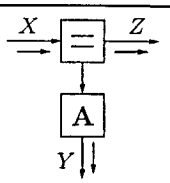
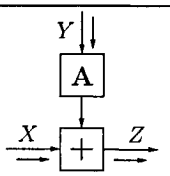
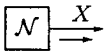
Example 4.1. (Equality constraint node)
Here we give an example of how such rules are derived. The equality node (Table 4.1-1) represents the factor $f(\mathbf{x},\mathbf{y},\mathbf{z}) = \delta(\mathbf{x}-\mathbf{z})\delta(\mathbf{y}-\mathbf{z})$. It can also be viewed as the constraint that configurations are valid only if all variables take on the same value.

$$\mu_{f\to\mathbf{Z}}(\mathbf{z}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\mathbf{x},\mathbf{y},\mathbf{z})\mu_{\mathbf{X}\to f}(\mathbf{x})\mu_{\mathbf{Y}\to f}(\mathbf{y})d\mathbf{x}d\mathbf{y} \tag{4.7}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(\mathbf{x}-\mathbf{z})\delta(\mathbf{y}-\mathbf{z})\mu_{\mathbf{X}\to f}(\mathbf{x})\mu_{\mathbf{Y}\to f}(\mathbf{y})d\mathbf{x}d\mathbf{y} \tag{4.8}$$

$$= \mu_{\mathbf{Z}\to f}(\mathbf{z}) \int_{-\infty}^{\infty} \delta(\mathbf{y}-\mathbf{z})\mu_{\mathbf{Y}\to f}(\mathbf{y})d\mathbf{y} \tag{4.9}$$

$$= \mu_{\mathbf{Z}\to f}(\mathbf{z}) \cdot \mu_{\mathbf{Z}\to f}(\mathbf{z}) \tag{4.10}$$

	Node	Update rule
1	 Equality	$m_Z = (W_X + W_Y)^{\#}(W_X m_X + W_Y m_Y)$ $V_Z = V_X(V_X + V_Y)^{\#}V_Y$ $W_Z = W_X + W_Y$ $\xi_Z = \xi_X + \xi_Y$
2	 Addition	$m_Z = m_X + m_Y$ $V_Z = V_X + V_Y$ $W_Z = W_X(W_X + W_Y)^{\#}W_Y$ $\xi_Z = (V_X + V_Y)^{\#}(V_X\xi_X + V_Y\xi_Y)$
3	 Forward multiplication	$m_Y = Am_X$ $V_Y = AV_XA^{\mathsf{T}}$ $W_Y \overset{3}{=} A^{-\mathsf{T}}W_XA^{-1}$ $\xi_Y = (AV_XA^{\mathsf{T}})^{\#}AV_X\xi_X \overset{1}{=} A^{-\mathsf{T}}\xi_X$
4	 Backward multiplication	$m_X = (A^{\mathsf{T}}W_YA)^{\#}A^{\mathsf{T}}W_Ym_Y \overset{2}{=} A^{-1}m_Y$ $V_X \overset{3}{=} A^{-1}V_YA^{-\mathsf{T}}$ $W_X = A^TW_YA$ $\xi_X = A^T\xi_Y$
5	 Equ.-mult. composite	$m_Z = m_X + V_XA^{\mathsf{T}}G(m_Y - Am_X)$ $V_Z = V_X - V_XA^{\mathsf{T}}GAV_X$ $W_Z = W_X + A^{\mathsf{T}}W_YA$ $\xi_Z = \xi_X + A^{\mathsf{T}}\xi_Y$ with $G = (V_Y + AV_XA^{\mathsf{T}})^{-1}$
6	 Mult.-add composite	$m_Z = m_X + Am_Y$ $V_Z = V_X + AV_YA^{\mathsf{T}}$ $W_Z = W_X - W_XAHA^{\mathsf{T}}W_X$ $\xi_Z = \xi_X + W_XAH(\xi_Y - A^{\mathsf{T}}\xi_X)$ with $H = (W_Y + A^{\mathsf{T}}W_XA)^{-1}$
7	 A priori	$m_x = m$ $V_x = V$ $W_x = W$

$^{\#}$ denotes the Moore-Penrose pseudoinverse

[1] if A and V_x are positive definite

[2] if A and W_x are positive definite

[3] if A is invertible

Table 4.1: Update equations for standard building blocks.

(4.10) is valid for any type of message. For the Gaussian case it becomes

$$\mu_{f\to\mathbf{z}}(\mathbf{z}) = \frac{|\mathbf{W}_X|}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{m}_X)^{\mathsf{T}}\mathbf{W}_X(\mathbf{z}-\mathbf{m}_X)\right) \cdot \tag{4.11}$$

$$\frac{|\mathbf{W}_Y|}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{m}_Y)^{\mathsf{T}}\mathbf{W}_Y(\mathbf{z}-\mathbf{m}_Y)\right) \tag{4.12}$$

$$= \frac{|\mathbf{W}_Z|}{\sqrt{2\pi)}} \exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{m}_Z)^{\mathsf{T}}\mathbf{W}_Z(\mathbf{z}-\mathbf{m}_Z)\right) \tag{4.13}$$

where

$$\mathbf{m}_Z = (\mathbf{W}_X + \mathbf{W}_Y)^{\#}(\mathbf{W}_X\mathbf{m}_X + \mathbf{W}_Y\mathbf{m}_Y) \tag{4.14}$$

$$\mathbf{W}_Z = \mathbf{W}_X + \mathbf{W}_Y \tag{4.15}$$

When using $\mathbf{V}$ instead of $\mathbf{W}$, $\mathbf{V}_Z$ is

$$\mathbf{V}_Z = \mathbf{V}_X(\mathbf{V}_X + \mathbf{V}_Y)^{\#}\mathbf{V}_Y. \tag{4.16}$$

Because we allow $\mathbf{V}$ or $\mathbf{W}$ to have reduced rank, the exact inverse is replaced by the Moore-Penrose pseudoinverse, which is symbolised by the $^{\#}$. Additionally, Gaussian messages may be represented by the weighted mean $\boldsymbol{\xi}_Z = \mathbf{W}_Z\mathbf{m}_Z$, so

$$\boldsymbol{\xi}_Z = \boldsymbol{\xi}_X + \boldsymbol{\xi}_Y. \tag{4.17}$$

□

The application of those nodes/update rules is quite universal; almost every graph in this thesis makes use of them. The main application, though, is linear Gaussian state-space estimation as demonstrated in the following section.

4.3 Kalman Filtering and Smoothing

In this section the Gaussian message update rules for linear building blocks, tabulated in Section 4.2, are applied to derive the classic Kalman filter recursions as message passing on the corresponding factor graph.

4.3.1 Kalman filter

The Kalman filter is a linear Gaussian estimator for the state of a linear state-space model given noisy observations [65]. The control input (or process noise) $\{U_n\}$ and measurement noise (or observation noise) $\{W_n\}$ are modelled as i.i.d. Gaussian noise sequences with zero mean and variance σ_U^2 and σ_W^2, respectively. Without loss of generality we assume scalar input and output.

The corresponding factor graph for the Kalman filter is shown in Fig. 4.1(a). The arrows represent the messages sent along the edges of the graph. The update schedule is defined by the circled numbers. In the following, all messages for one section are defined. The algorithm computes messages ① to ⑨ for every time step, which results in a forward-only update schedule (cf. Section 3.2.1), i.e. information is only propagated from the past to the future.

All messages are Gaussian functions and are indexed by their respective number as shown in Fig. 4.1(a). To increase readability we have not labelled every edge in the graph. A dot is given in the formulas as a replacement for the variable name.

① State estimate at time $n-1$ (propagated from the section to the left):

$$\mu_1(\mathbf{x}_{n-1}) = \mathcal{N}\Big(\mathbf{x}_{n-1} \;\Big|\; \hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}, \mathbf{V}_{n-1|\overrightarrow{n-1}}\Big) \quad [1]$$

② State transition (Table 4.1-3):

$$\mu_2(.) = \mathcal{N}\Big(\,.\;\Big|\; \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}, \mathbf{A}\mathbf{V}_{n-1|\overrightarrow{n-1}}\mathbf{A}^{\mathsf{T}}\Big)$$

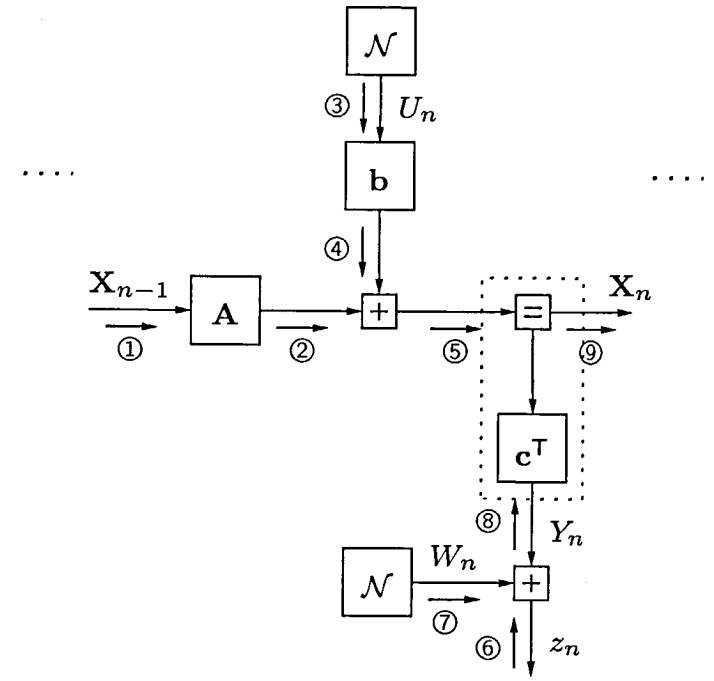
③ Control input is modelled as i.i.d. Gaussian noise (Table 4.1-7):

$$\mu_3(u_n) = \mathcal{N}\big(u_n \mid 0, \sigma_U^2\big)$$

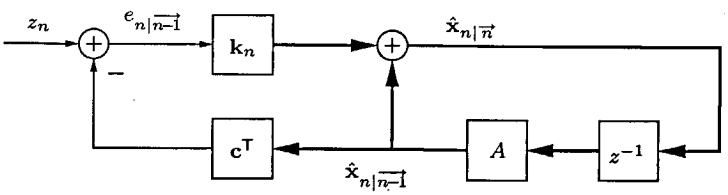
④ Multiplication (Table 4.1-3):

$$\mu_4(.) = \mathcal{N}\big(\,.\mid \mathbf{0}, \sigma_U^2\mathbf{b}\mathbf{b}^{\mathsf{T}}\big)$$

[1] Notation: $\hat{\mathbf{x}}_{n|\overrightarrow{n-1}}$ denotes the state estimate at time n given all observations up to time $n-1$, i.e. $\ldots, z_{n-2}, z_{n-1}$. Contrary, $\hat{\mathbf{x}}_{n|\overleftarrow{n+1}}$ denotes the (backward) state estimate at time n given all observations $z_{n+1}, z_{n+2}, \ldots$



(a) Factor graph with messages.



(b) Recursive estimator.

Figure 4.1: The Kalman filter. Computing all messages in the given order in (a) for every step in time amounts to the well known recursive estimator (b) for the state $\mathbf{X}_n$ given all observations $(Z_1, \ldots, Z_n) = (z_1, \ldots, z_n)$ up to time n.

⑤ Adding input and state (Table 4.1-2):

$$\mu_5(.) = \mathcal{N}\Big(. \,\Big|\, \hat{\mathbf{x}}_{n|\overrightarrow{n-1}}, \mathbf{V}_{n|\overrightarrow{n-1}}\Big)$$

$$\hat{\mathbf{x}}_{n|\overrightarrow{n-1}} = \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}} \tag{4.18}$$

$$\mathbf{V}_{n|\overrightarrow{n-1}} = \mathbf{A}\mathbf{V}_{n-1|\overrightarrow{n-1}}\mathbf{A}^{\mathsf{T}} + \sigma_U^2\mathbf{b}\mathbf{b}^{\mathsf{T}} \tag{4.19}$$

Here, we introduce the new symbols $\hat{\mathbf{x}}_{n|\overrightarrow{n-1}}$ for the estimate and $\mathbf{V}_{n|\overrightarrow{n-1}}$ for the variance, because they also appear in the classic Kalman filter recursion. They define the prediction distribution.

⑥ Incorporating the observation. This edge could have been removed because the value z_n of the variable Z_n is known. Alternatively, a message expressing complete certainty can be sent along the edge (Table 4.1-7):

$$\mu_6(.) = \mathcal{N}(. \mid z_n, 0)$$

⑦ Measurement noise (Table 4.1-7):

$$\mu_7(w_n) = \mathcal{N}(w_n \mid 0, \sigma_W^2)$$

⑧ Adding the measurement noise (Table 4.1-2):

$$\mu_8(.) = \mathcal{N}\Big(. \mid z_n, \sigma_W^2\Big)$$

⑨ Update the state with information from new observation. Here, the composite-block rule (Table 4.1-5) is used. Literal application of the rules for the backward matrix multiplication (Table 4.1-4) and the equality node (Table 4.1-1) would require a matrix inversion, which can be circumvented by application of the matrix inversion lemma in the composite block [83].

$$\mu_9(\mathbf{x}_n) = \mathcal{N}(\mathbf{x}_n \mid \hat{\mathbf{x}}_{n|\overrightarrow{n}}, \mathbf{V}_{n|\overrightarrow{n}})$$

$$\hat{\mathbf{x}}_{n|\overrightarrow{n}} = \hat{\mathbf{x}}_{n|\overrightarrow{n-1}} + \frac{\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}\,(z_n - \mathbf{c}^{\mathsf{T}}\hat{\mathbf{x}}_{n|\overrightarrow{n-1}})}{\sigma_W^2 + \mathbf{c}^{\mathsf{T}}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}} \tag{4.20}$$

$$\mathbf{V}_{n|\overrightarrow{n}} = \mathbf{V}_{n|\overrightarrow{n-1}} - \frac{\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}\mathbf{c}^{\mathsf{T}}\mathbf{V}_{n|\overrightarrow{n-1}}}{\sigma_W^2 + \mathbf{c}^{\mathsf{T}}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}} \tag{4.21}$$

Combining (4.18)-(4.21) leads to the well known Kalman recursion:

$$\hat{\mathbf{x}}_{n|\overrightarrow{n}} = \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}} + \mathbf{k}_n e_{n|\overrightarrow{n-1}} \tag{4.22}$$

$$\mathbf{V}_{n|\overrightarrow{n-1}} = \mathbf{A}\mathbf{V}_{n-1|\overrightarrow{n-1}}\mathbf{A}^{\mathsf{T}} + \sigma_U^2 \mathbf{b}\mathbf{b}^{\mathsf{T}} \tag{4.23}$$

$$\mathbf{V}_{n|\overrightarrow{n}} = (\mathbf{I} - \mathbf{k}_n \mathbf{c}^{\mathsf{T}})\mathbf{V}_{n|\overrightarrow{n-1}} \tag{4.24}$$

with prediction error and Kalman gain

$$e_{n|\overrightarrow{n-1}} = z_n - \mathbf{a}^{\mathsf{T}}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}} \tag{4.25}$$

$$\mathbf{k}_n = \frac{\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}}{\sigma_W^2 + \mathbf{c}^{\mathsf{T}}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}}. \tag{4.26}$$

The messages are computed in the given order for one section after the other in a forward-only (i.e. left-to-right) manner (message ⑨ of the current section is message ① in the subsequent section). This leads to the recursive estimator as shown in Fig. 4.1(b).

4.3.2 Information filter

Gaussian messages can not only be represented by the mean vector $\mathbf{m}$ and the covariance matrix $\mathbf{V}$, but also by the mean vector $\mathbf{m}$ and the inverse covariance or weight matrix $\mathbf{W} = \mathbf{V}^{-1}$. Additionally, the mean can be replaced by the product $\boldsymbol{\xi} = \mathbf{W}\mathbf{m}$ of weight matrix and mean. The notation $\mathcal{N}_W$ indicates that the weight matrix is used to represent the Gaussian function. Computing messages ① to ⑨ similar to Section 4.3.1 but using $\mathbf{W}$ instead of $\mathbf{V}$ results in a recursion in the weight matrix $\mathbf{W}$:

① State estimate at time $n-1$ (propagated from the section to the left):

$$\mu_1(\mathbf{x}_{n-1}) = \mathcal{N}_W\left(\mathbf{x}_{n-1} \;\middle|\; \hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}, \mathbf{W}_{n-1|\overrightarrow{n-1}}\right)$$

② State transition (Table 4.1-3):

$$\mu_2(.) = \mathcal{N}_W(\,.\,|\,\mathbf{m}_2, \mathbf{W}_2)$$

$$\mathbf{m}_2 = \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}} \tag{4.27}$$

$$\mathbf{W}_2 = \mathbf{A}^{-\mathsf{T}}\mathbf{W}_{n-1|\overrightarrow{n-1}}\mathbf{A}^{-1} \tag{4.28}$$

$$\boldsymbol{\xi}_2 = \mathbf{W}_2\mathbf{m}_2 = \mathbf{A}^{-\mathsf{T}}\boldsymbol{\xi}_{n-1|\overrightarrow{n-1}} \tag{4.29}$$

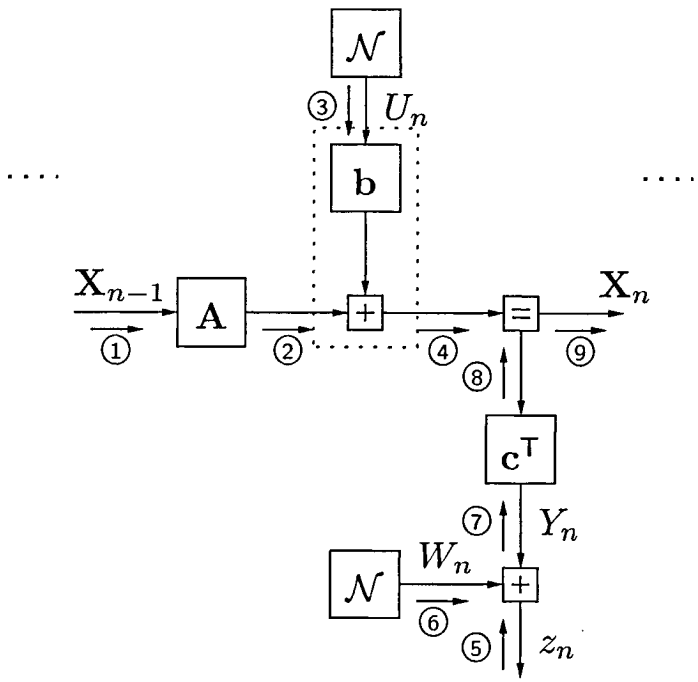


Figure 4.2: Factor graph for the information filter. Messages are Gaussian represented by mean vector $\mathbf{m}$ (or $\boldsymbol{\xi}$, see text) and information matrix $\mathbf{W}$.

where the subscript of $\mathbf{m}_2$ and $\mathbf{W}_2$ indicates, that these are parameters of message ②.
Two notes are convenient here. First, it suffices to propagate either the pair $(\mathbf{m}, \mathbf{W})$ or the pair $(\boldsymbol{\xi}, \mathbf{W})$. Second, for the systems under consideration in this thesis the transition matrix $\mathbf{A}$ is invertible.

③ Control input (Table 4.1-7):

$$\mu_3(u_n) = \mathcal{N}\big(u_n \mid 0, \sigma_U^2\big)$$

④ Adding input and state using the composite node for forward matrix multiplication and addition (Table 4.1-6):

$$\mu_4(.) = \mathcal{N}_W\Big(. \,\Big|\, \hat{\mathbf{x}}_{n|\overrightarrow{n-1}}, \mathbf{W}_{n|\overrightarrow{n-1}}\Big)$$

$$\hat{\mathbf{x}}_{n|\overrightarrow{n-1}} = \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}} \tag{4.30}$$

$$\mathbf{W}_{n|\overrightarrow{n-1}} = \mathbf{W}_2 - \frac{\mathbf{W}_2\mathbf{b}\mathbf{b}^\mathsf{T}\mathbf{W}_2}{1/\sigma_U^2 + \mathbf{b}^\mathsf{T}\mathbf{W}_2\mathbf{b}} \tag{4.31}$$

$$\boldsymbol{\xi}_{n|\overrightarrow{n-1}} = \boldsymbol{\xi}_2 - \frac{\mathbf{W}_2\mathbf{b}\mathbf{b}^\mathsf{T}\boldsymbol{\xi}_2}{1/\sigma_U^2 + \mathbf{b}^\mathsf{T}\mathbf{W}_2\mathbf{b}} \tag{4.32}$$

⑤ Observation (Table 4.1-7):

$$\mu_5(.) = \mathcal{N}(\,.\mid z_n, 0)$$

⑥ Measurement noise (Table 4.1-7):

$$\mu_6(w_n) = \mathcal{N}(w_n \mid 0, \sigma_W^2)$$

⑦ Adding the measurement noise (Table 4.1-2):

$$\mu_7(.) = \mathcal{N}_W(\,.\mid z_n, 1/\sigma_W^2)$$

⑧ Backward multiplication (Table 4.1-4):

$$\mu_8(.) = \mathcal{N}_W\left(\,.\,\middle|\, \mathbf{c}z_n, \frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2}\right)$$

⑨ Update with information from new observation (Table 4.1-1):

$$\mu_9(\mathbf{x}_n) = \mathcal{N}_W(\mathbf{x}_n \mid \hat{\mathbf{x}}_{n|\overrightarrow{n}}, \mathbf{W}_{n|\overrightarrow{n}})$$

$$\hat{\mathbf{x}}_{n|\overrightarrow{n}} = \mathbf{W}^{\#}_{n|\overrightarrow{n}}\left(\mathbf{W}_{n|\overrightarrow{n-1}}\hat{\mathbf{x}}_{n|\overrightarrow{n-1}} + \frac{\mathbf{c}z_n}{\sigma_W^2}\right) \tag{4.33}$$

$$\mathbf{W}_{n|\overrightarrow{n}} = \mathbf{W}_{n|\overrightarrow{n-1}} + \frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2} \tag{4.34}$$

$$\boldsymbol{\xi}_{n|\overrightarrow{n}} = \boldsymbol{\xi}_{n|\overrightarrow{n-1}} + \frac{\mathbf{c}z_n}{\sigma_W^2} \tag{4.35}$$

One can see that the update of the mean $\hat{\mathbf{x}}$ involves a matrix inversion, which can be circumvented by using $\boldsymbol{\xi}$ instead of $\hat{\mathbf{x}}$.

Combining (4.30)-(4.35) leads to the information filter recursion:

$$\mathbf{W}_{n|\overrightarrow{n-1}} = (\mathbf{I} - \mathbf{g}_n\mathbf{b}^\mathsf{T})\mathbf{A}^{-\mathsf{T}}\mathbf{W}_{n-1|\overrightarrow{n-1}}\mathbf{A}^{-1} \tag{4.36}$$

$$\mathbf{W}_{n|\overrightarrow{n}} = \mathbf{W}_{n|\overrightarrow{n-1}} + \frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2} \tag{4.37}$$

$$\boldsymbol{\xi}_{n|\overrightarrow{n}} = (\mathbf{I} - \mathbf{g}_n\mathbf{b}^\mathsf{T})\mathbf{A}^{-\mathsf{T}}\boldsymbol{\xi}_{n-1|\overrightarrow{n-1}} + \frac{\mathbf{c}z_n}{\sigma_W^2} \tag{4.38}$$

with Kalman gain

$$\mathbf{g}_n = \frac{\mathbf{b}\mathbf{A}^{-\mathsf{T}}\mathbf{W}_{n-1|\overrightarrow{n-1}}\mathbf{A}^{-1}}{1/\sigma_U^2 + \mathbf{b}^T\mathbf{A}^{-\mathsf{T}}\mathbf{W}_{n-1|\overrightarrow{n-1}}\mathbf{A}^{-1}\mathbf{b}} \tag{4.39}$$

If the state error covariance matrix is not needed explicitly, other recursions based on Chandrasekhar-type equations [3] can save computational effort. They can also be derived as message passing in a similar way. The difference $\mathbf{V}_{n+1|\overrightarrow{n}} - \mathbf{V}_{n|\overrightarrow{n-1}}$ is decomposed into $\mathbf{y}_n m_n \mathbf{y}_n^{\mathsf{T}}$ with rank 1. $\mathbf{y}_n$, m_n, the Kalman gain $\mathbf{g}_n$ and $\omega_n = \mathbf{c}^{\mathsf{T}}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c} + \sigma_U^2$ are then recursively computed. In this case, messages do not comprise mean vector and covariance matrix, but the decomposition variables ω_n, $\mathbf{g}_n$, $\mathbf{y}_n$ and m_n.

4.3.3 Kalman smoother

When at time n observations up to time $n+M$ are available, one can incorporate those observations to improve the estimate of the state $\mathbf{X}_n$. Therefore, information from time $n+M$ must be propagated back to section n. This backward pass is independent of the forward pass and can be carried out separately. Once the estimate has to be determined, forward and backward information must be combined.

The factor graph and the messages for the backward pass are shown in Fig. 4.3(a). For an explanation of the notation see the footnote on page 34. The forward pass is computed as in Section 4.3.1 or 4.3.2.

① Backward state estimate at time $n+1$:

$$\mu_1(\mathbf{x}_{n+1}) = \mathcal{N}\left(\mathbf{x}_{n+1} \;\middle|\; \hat{\mathbf{x}}_{n+1|\overleftarrow{n+2}}, \mathbf{V}_{n+1|\overleftarrow{n+2}}\right)$$

② Observation (Table 4.1-7):

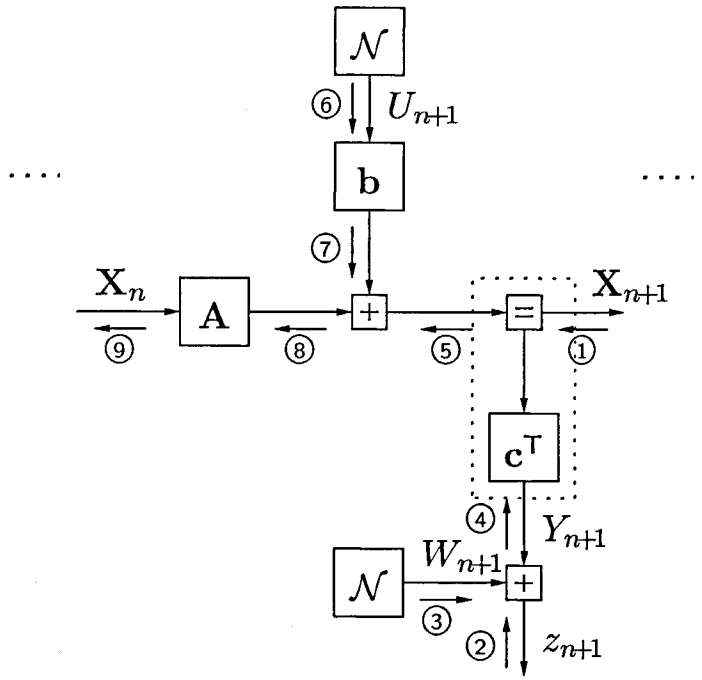
$$\mu_2(.) = \mathcal{N}(\,.\mid z_{n+1}, 0)$$

③ Measurement noise (Table 4.1-7):

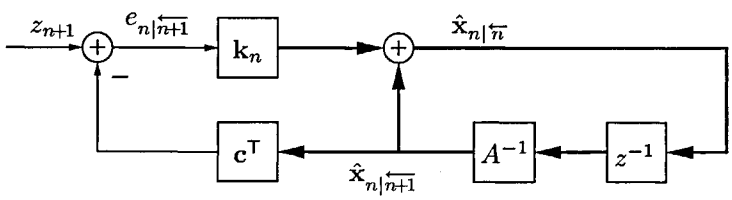
$$\mu_3(w_{n+1}) = \mathcal{N}(w_{n+1} \mid 0, \sigma_W^2)$$

④ Adding the measurement noise (Table 4.1-2):

$$\mu_4(.) = \mathcal{N}(\,.\mid z_{n+1}, \sigma_W^2)$$



(a) Factor graph and messages for the backward pass.



(b) Backward recursion.

Figure 4.3: The Kalman smoother. The backward pass consists of all messages as shown in (a) which leads to the recursive (backward in time) estimator in (b). To obtain smoothed estimates for the state, the forward and the backward messages need to be combined.

⑤ Update with information from new observation (Table 4.1-5):

$$\mu_5(.) = \mathcal{N}\Big(. \;\Big|\; \hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}}, \mathbf{V}_{n+1|\overleftarrow{n+1}} \Big)$$

$$\hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}} = \hat{\mathbf{x}}_{n+1|\overleftarrow{n+2}} + \frac{\mathbf{V}_{n+1|\overleftarrow{n+2}}\mathbf{c}\left(z_{n+1} - \mathbf{c}^{\mathsf{T}}\hat{\mathbf{x}}_{n+1|\overleftarrow{n+2}}\right)}{\sigma_W^2 + \mathbf{c}^{\mathsf{T}}\mathbf{V}_{n+1|\overleftarrow{n+2}}\mathbf{c}} \tag{4.40}$$

$$\mathbf{V}_{n+1|\overleftarrow{n+1}} = \mathbf{V}_{n+1|\overleftarrow{n+2}} - \frac{\mathbf{V}_{n+1|\overleftarrow{n+2}}\mathbf{c}\mathbf{c}^{\mathsf{T}}\mathbf{V}_{n+1|\overleftarrow{n+2}}}{\sigma_W^2 + \mathbf{c}^{\mathsf{T}}\mathbf{V}_{n+1|\overleftarrow{n+2}}\mathbf{c}} \tag{4.41}$$

⑥ Input (Table 4.1-7):

$$\mu_6(u_{n+1}) = \mathcal{N}(u_{n+1} \mid 0, \sigma_U^2)$$

⑦ Multiplication (Table 4.1-3):

$$\mu_7(.) = \mathcal{N}\big(. \mid \mathbf{0}, \sigma_U^2\mathbf{b}\mathbf{b}^{\mathsf{T}} \big)$$

⑧ Adding input (Table 4.1-2):

$$\mu_8(.) = \mathcal{N}\Big(. \;\Big|\; \hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}}, \mathbf{V}_{n+1|\overleftarrow{n+1}} + \sigma_U^2\mathbf{b}\mathbf{b}^{\mathsf{T}} \Big)$$

⑨ Backward matrix multiplication (Table 4.1-4):

$$\mu_9(\mathbf{x}_n) = \mathcal{N}\Big(\mathbf{x}_n \;\Big|\; \hat{\mathbf{x}}_{n|\overleftarrow{n+1}}, \mathbf{V}_{n|\overleftarrow{n+1}} \Big)$$

$$\hat{\mathbf{x}}_{n|\overleftarrow{n+1}} = \mathbf{A}^{-1}\hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}} \tag{4.42}$$

$$\mathbf{V}_{n|\overleftarrow{n+1}} = \mathbf{A}^{-1}\left(\mathbf{V}_{n+1|\overleftarrow{n+1}} + \sigma_U^2\mathbf{b}\mathbf{b}^{\mathsf{T}}\right)\mathbf{A}^{-\mathsf{T}} \tag{4.43}$$

Combining (4.40)-(4.43) leads to the following recursion:

$$\hat{\mathbf{x}}_{n|\overleftarrow{n}} = \mathbf{A}^{-1}\mathbf{x}_{n+1|\overleftarrow{n+1}} + \mathbf{k}_n e_{n|\overleftarrow{n+1}} \tag{4.44}$$

$$\mathbf{V}_{n|\overleftarrow{n+1}} = \mathbf{A}^{-1}\left(\mathbf{V}_{n+1|\overleftarrow{n+1}} + \sigma_U^2\mathbf{b}\mathbf{b}^{\mathsf{T}}\right)\mathbf{A}^{-\mathsf{T}} \tag{4.45}$$

$$\mathbf{V}_{n|\overleftarrow{n}} = (\mathbf{I} - \mathbf{k}_n\mathbf{c}^{\mathsf{T}})\mathbf{V}_{n|\overleftarrow{n+1}} \tag{4.46}$$

with backward prediction error and backward Kalman gain:

$$e_{n|\overleftarrow{n+1}} = z_n - \mathbf{c}^{\mathsf{T}}\mathbf{A}^{-1}\hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}} \tag{4.47}$$

$$\mathbf{k}_n = \frac{\mathbf{V}_{n|\overleftarrow{n+1}}\mathbf{c}}{\sigma_W^2 + \mathbf{c}^{\mathsf{T}}\mathbf{V}_{n|\overleftarrow{n+1}}\mathbf{c}} \tag{4.48}$$

The resulting backward estimator can be seen in Fig. 4.3(b).

To obtain smoothed state estimates, the forward and the backward messages need to be combined. The quantity $\mu_{\text{tot}}(\mathbf{x}_n)$ denotes the total function for edge $\mathbf{X}_n$ and is computed as a multiplication of the forward and the backward message (cf. (2.7)). Because in the equality node the incoming messages are combined in the same way, we can use the update formulas of Table 4.1-1:

$$\begin{aligned}
\mu_{\text{tot}}(\mathbf{x}_n) &= \mathcal{N}(\mathbf{x}_n \mid \hat{\mathbf{x}}_{n,\text{tot}}, \mathbf{V}_{n,\text{tot}}) \\
\hat{\mathbf{x}}_{n,\text{tot}} &= \mathbf{V}_{n,\text{tot}} \left(\mathbf{V}^{\#}_{n|\overrightarrow{n}} \hat{\mathbf{x}}_{n|\overrightarrow{n}} + \mathbf{V}^{\#}_{n|\overleftarrow{n+1}} \hat{\mathbf{x}}_{n|\overleftarrow{n+1}} \right) \\
\mathbf{V}_{n,\text{tot}} &= \mathbf{V}_{n|\overrightarrow{n}} \left(\mathbf{V}_{n|\overrightarrow{n}} + \mathbf{V}_{n|\overleftarrow{n+1}} \right)^{\#} \mathbf{V}_{n|\overleftarrow{n+1}}
\end{aligned}$$

4.3.4 Information smoother

Again, using the information matrix instead of the covariance matrix for the state (Fig. 4.4) leads to a recursion in $(\mathbf{m}, \mathbf{W})$ or $(\boldsymbol{\xi}, \mathbf{W})$.

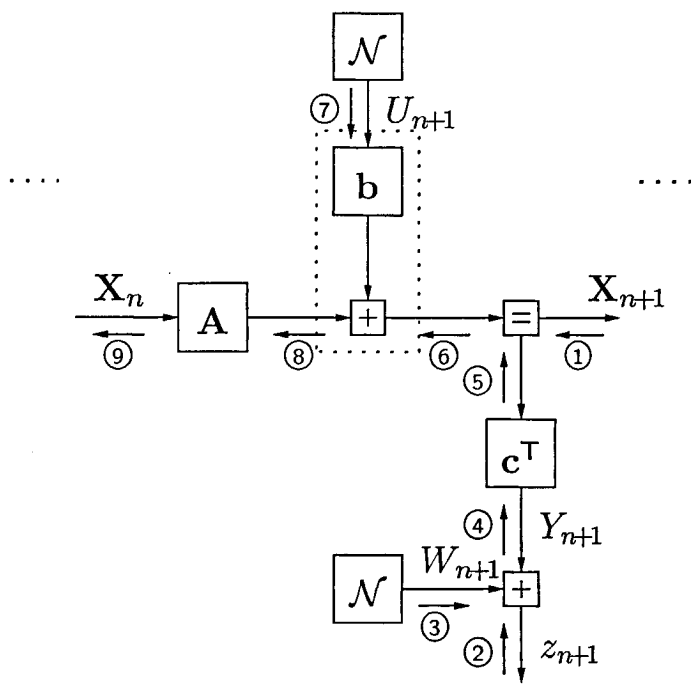


Figure 4.4: Factor graph and messages for the backward pass for the information smoother.

① Backward state estimate at time $n+1$:

$$\mu_1(\mathbf{x}_{n+1}) = \mathcal{N}_W\left(\mathbf{x}_{n+1} \;\middle|\; \hat{\mathbf{x}}_{n+1|\overleftarrow{n+2}}, \mathbf{W}_{n+1|\overleftarrow{n+2}}\right)$$

② Observation (Table 4.1-7):

$$\mu_2(.) = \mathcal{N}(\,.\mid z_{n+1}, 0)$$

③ Measurement noise (Table 4.1-7):

$$\mu_3(w_{n+1}) = \mathcal{N}(w_{n+1} \mid 0, \sigma_W^2)$$

④ Adding the measurement noise (Table 4.1-2):

$$\mu_4(.) = \mathcal{N}_W(\,.\mid z_{n+1}, 1/\sigma_W^2)$$

⑤ Backward multiplication (Table 4.1-4):

$$\mu_5(.) = \mathcal{N}_W\left(\,. \;\middle|\; \mathbf{c}y_{n+1}, \frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2}\right)$$

⑥ Update with information from new observation (Table 4.1-1):

$$\mu_6(.) = \mathcal{N}\left(\,.\mid \hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}}, \mathbf{W}_{n+1|\overleftarrow{n+1}}\right)$$

$$\hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}} = \mathbf{W}^{\#}_{n+1|\overleftarrow{n+1}}\left(\mathbf{W}_{n+1|\overleftarrow{n+2}}\hat{\mathbf{x}}_{n+1|\overleftarrow{n+2}} + \frac{\mathbf{c}z_n}{\sigma_W^2}\right) \tag{4.49}$$

$$\mathbf{W}_{n+1|\overleftarrow{n+1}} = \mathbf{W}_{n+1|\overleftarrow{n+2}} + \frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2} \tag{4.50}$$

$$\boldsymbol{\xi}_{n+1|\overleftarrow{n+1}} = \boldsymbol{\xi}_{n+1|\overleftarrow{n+2}} + \frac{\mathbf{c}z_n}{\sigma_W^2} \tag{4.51}$$

⑦ Input (Table 4.1-7):

$$\mu_7(u_{n+1}) = \mathcal{N}(u_{n+1} \mid 0, \sigma_U^2)$$

⑧ Adding input (Table 4.1-6):

$$\mu_8(.) = \mathcal{N}_W\left(\,. \;\middle|\; \hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}}, \mathbf{W}_8\right)$$

$$\mathbf{W}_8 = \mathbf{W}_{n+1|\overleftarrow{n+1}} - \frac{\mathbf{W}_{n+1|\overleftarrow{n+1}}\mathbf{b}\mathbf{b}^\mathsf{T}\mathbf{W}_{n+1|\overleftarrow{n+1}}}{1/\sigma_U^2 + \mathbf{b}\mathbf{W}_{n+1|\overleftarrow{n+1}}\mathbf{b}} \tag{4.52}$$

$$\boldsymbol{\xi}_8 = \mathbf{W}_{n+1|\overleftarrow{n+1}} - \frac{\mathbf{W}_{n+1|\overleftarrow{n+1}}\mathbf{b}\mathbf{b}^\mathsf{T}\boldsymbol{\xi}_{n+1|\overleftarrow{n+1}}}{1/\sigma_U^2 + \mathbf{b}\mathbf{W}_{n+1|\overleftarrow{n+1}}\mathbf{b}} \tag{4.53}$$

⑨ (Table 4.1-4):

$$\mu_9(\mathbf{x}_n) = \mathcal{N}_W\left(\mathbf{x}_n \;\middle|\; \hat{\mathbf{x}}_{n|\overleftarrow{n+1}}, \mathbf{W}_{n|\overleftarrow{n+1}}\right)$$

$$\hat{\mathbf{x}}_{n|\overleftarrow{n+1}} = \mathbf{A}^{-1}\hat{\mathbf{x}}_{n+1|\overleftarrow{n+1}} \tag{4.54}$$

$$\mathbf{W}_{n|\overleftarrow{n+1}} = \mathbf{A}^{\mathsf{T}}\mathbf{W}_8\mathbf{A} \tag{4.55}$$

$$\boldsymbol{\xi}_{n|\overleftarrow{n+1}} = \mathbf{A}^{\mathsf{T}}\boldsymbol{\xi}_8 \tag{4.56}$$

Combining (4.49)-(4.56) leads to the following recursion:

$$\mathbf{W}_{n|\overleftarrow{n+1}} = \mathbf{A}^{-1}(\mathbf{I} - \mathbf{g}_n\mathbf{b}^{\mathsf{T}})\mathbf{W}_{n+1|\overleftarrow{n+1}}\mathbf{A}^{-\mathsf{T}} \tag{4.57}$$

$$\mathbf{W}_{n|\overleftarrow{n}} = \mathbf{W}_{n|\overleftarrow{n+1}} + \frac{\mathbf{c}\mathbf{c}^{\mathsf{T}}}{\sigma_W^2} \tag{4.58}$$

$$\boldsymbol{\xi}_{n|\overleftarrow{n}} = \mathbf{A}^{\mathsf{T}}(\mathbf{I} - \mathbf{g}_n\mathbf{b}^{\mathsf{T}})\boldsymbol{\xi}_{n+1|\overleftarrow{n+1}} + \frac{\mathbf{c}z_n}{\sigma_W^2} \tag{4.59}$$

To get smoothed state estimates forward and backward messages have to be combined:

$$\begin{aligned}
\mu_{\text{tot}}(\mathbf{x}_n) &= \mathcal{N}_W(\mathbf{x}_n \mid \hat{\mathbf{x}}_{n,\text{tot}}, \mathbf{W}_{n,\text{tot}}) \\
\hat{\mathbf{x}}_{n,\text{tot}} &= \mathbf{W}^{\#}_{n,\text{tot}}\left(\mathbf{W}_{n|\overrightarrow{n}}\hat{\mathbf{x}}_{n|\overrightarrow{n}} + \mathbf{W}_{n|\overleftarrow{n+1}}\hat{\mathbf{x}}_{n|\overleftarrow{n+1}}\right) \\
&= \mathbf{W}^{\#}_{n,\text{tot}}\left(\boldsymbol{\xi}_{n|\overrightarrow{n}} + \boldsymbol{\xi}_{n|\overleftarrow{n+1}}\right) \\
\mathbf{W}_{n,\text{tot}} &= \mathbf{W}_{n|\overrightarrow{n}} + \mathbf{W}_{n|\overleftarrow{n+1}}
\end{aligned}$$

There are different possibilities how both the forward and the backward messages are represented. In both directions either the covariance matrix $\mathbf{V}$ or the information matrix $\mathbf{W}$ or both can be used. Neither recursion involves a matrix inversion; only when computing the a posteriori probability for the state (i.e. the combination of forward and backward messages) at least one matrix inversion has to be computed. In addition, when estimates of the control input and the output are needed, a minimum of three matrix inversions per section are needed, which is the main cost factor in all the algorithms in this section.

4.4 Linear Prediction

In this section the problem of estimating the coefficients of a linear predictor is considered. This problem again demonstrates the use of Gaussian messages and the tabulated update rules for linear building blocks (Table 4.10).

The classic linear predictive coding (LPC) algorithm solves the following estimation problem. Let $a_1, a_2, \ldots, a_M$ (with $a_\ell \in \mathbb{R}$) be the coefficients of a linear predictor of order M. Let $U_1, U_2, \ldots$ with $U_n \in \mathbb{R}$, the model of the prediction error, be white Gaussian noise with zero mean and variance one.

$$X_n = \sum_{\ell=1}^{M} a_\ell X_{n-\ell} + U_n \tag{4.60}$$

Assume that the predictor coefficients $\mathbf{a} = (a_1, \ldots, a_M)^\mathsf{T}$ are not known; instead, we know the signal $X_1, X_2, \ldots = x_1, x_2, \ldots$
Based on $x_1, x_2, \ldots$ we wish to estimate $\mathbf{a}$.

Since the unknown predictor coefficients will appear as variables in the factor graph, we will denote them by capital letters:

$$\mathbf{A} = (A_1, \ldots, A_M)^\mathsf{T}. \tag{4.61}$$

We also define

$$\mathbf{x}_n \triangleq (x_n, x_{n-1}, \ldots, x_{n-M+1})^\mathsf{T}. \tag{4.62}$$

With this notation (4.60) becomes

$$x_n = \mathbf{A}^\mathsf{T} \mathbf{x}_{n-1} + U_n. \tag{4.63}$$

A factor graph of this system is shown in Fig. 4.5. The predictor coefficients $\mathbf{A}$ can be estimated by forward-only Gaussian message passing. The initial message for $\mathbf{A}$, which corresponds to $n = 0$, may be set to $\boldsymbol{\xi}_{A,0} = \mathbf{0}$ the zero vector and $\mathbf{W}_{A,0} = \mathbf{0}$ the zero matrix. The rest of the messages are computed according to the following scheme. Since every message is a Gaussian specified by (m, V) or (ξ, W) the update rules of Table 4.1 apply.

① $m_1 = x_n$, $V_1 = 0$.

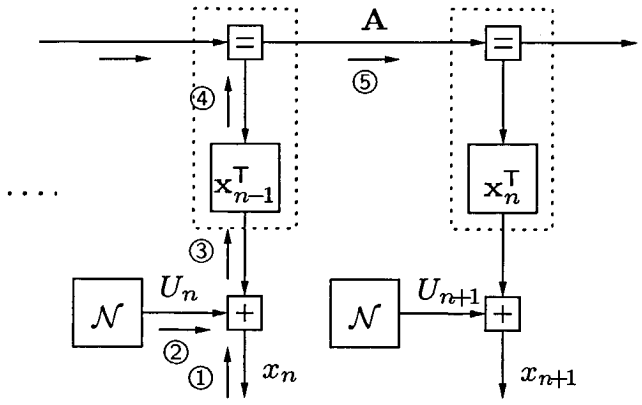


Figure 4.5: Factor graph for linear prediction.

② (Table 4.1-7): $m_2 = 0$, $V_2 = 1$.

③ (Table 4.1-2): $m_3 = x_n$, $V_3 = 1$,
which can also be expressed as $\xi_3 = x_n$, $W_3 = 1$.

There are two ways of computing the messages for the coefficient vector. If the Gaussian messages are represented by the weight matrix $\mathbf{W}$ messages ④ and ⑤ are computed as follows:

④ (Table 4.1-4): $\boldsymbol{\xi}_4 = \mathbf{x}_{n-1}x_n$, $\mathbf{W}_4 = \mathbf{x}_{n-1}\mathbf{x}_{n-1}^{\mathsf{T}}$.

⑤ (Table 4.1-1):

$$\boldsymbol{\xi}_{A,n} = \boldsymbol{\xi}_{A,n-1} + \mathbf{x}_{n-1}x_n = \sum_{k=1}^{n} \mathbf{x}_{k-1}x_k$$

$$\mathbf{W}_{A,n} = \mathbf{W}_{A,n-1} + \mathbf{x}_{n-1}\mathbf{x}_{n-1}^{\mathsf{T}} = \sum_{k=1}^{n} \mathbf{x}_{k-1}\mathbf{x}_{k-1}^{\mathsf{T}}$$

The estimate of the coefficient vector at section n is

$$\hat{\mathbf{a}}_n = \mathbf{W}_{A,n}^{-1}\boldsymbol{\xi}_{A,n} = \left(\sum_{k=1}^{n} \mathbf{x}_{k-1}\mathbf{x}_{k-1}^{\mathsf{T}}\right)^{-1} \left(\sum_{k=1}^{n} \mathbf{x}_{k-1}x_k\right) = \mathbf{R}_n^{-1}\mathbf{r}_n \quad (4.64)$$

where $\mathbf{R}_n$ is the (auto-)correlation matrix and $\mathbf{r}_n$ is the correlation vector in linear prediction literature [26]. The solution of (4.64) is referred to as the classic autocorrelation solution of linear prediction analysis. Since

$\mathbf{R}_n$ is Toeplitz (4.64) may be solved by the Levinson-Durbin recursion with only $O(M^2)$ operations.

Alternatively, the Gaussian messages for the coefficients can be represented by the covariance matrix $\mathbf{V}$. In that case message ④ is not computed explicitely, but the update rule for the composite block (dotted boxes in Fig. 4.5) is applied:

⑤ (Table 4.1-5)

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \mathbf{V}_{A,n-1}\mathbf{x}_{n-1}\frac{x_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}}\mathbf{x}_{n-1}}{1 + \mathbf{x}_{n-1}^{\mathsf{T}}\mathbf{V}_{n-1}\mathbf{x}_{n-1}} \tag{4.65}$$

$$= \hat{\mathbf{a}}_{n-1} + \mathbf{k}_n e_n \tag{4.66}$$

$$\mathbf{V}_{A,n} = \mathbf{V}_{A,n-1} - \frac{\mathbf{V}_{A,n-1}\mathbf{x}_{n-1}\mathbf{x}_{n-1}^{\mathsf{T}}\mathbf{V}_{A,n-1}}{1 + \mathbf{x}_{n-1}^{\mathsf{T}}\mathbf{V}_{n-1}\mathbf{x}_{n-1}} \tag{4.67}$$

$$= (\mathbf{I} - \mathbf{k}_n\mathbf{x}_{n-1}^{\mathsf{T}})\mathbf{V}_{A,n-1} \tag{4.68}$$

with

$$e_n = x_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}}\mathbf{x}_{n-1} \tag{4.69}$$

$$\mathbf{k}_n = \frac{\mathbf{V}_{A,n-1}\mathbf{x}_{n-1}}{1 + \mathbf{x}_{n-1}^{\mathsf{T}}\mathbf{V}_{n-1}\mathbf{x}_{n-1}} \tag{4.70}$$

As becomes obvious from (4.66) and (4.68) the estimate $\hat{\mathbf{a}}_n$ is available at every section n. Additionally, no matrix inversion is involved in the update of the estimate.

It is noteworthy that the factor graph of Fig. 4.5 with update equations (4.66) and (4.68) is a degenerate Kalman filter. The equivalence of general recursive least squares algorithms and the Kalman filter has been pointed out before [55,81,116]; in the factor graph this equivalence is eye-catching.

4.5 Recursive Least Squares

The problem of estimating the coefficients of an adaptive filter is considered in this section. Similar to the linear prediction problem in Section 4.4 this problem can be solved with Gaussian messages. It is also a special case of Kalman filtering.

The classic recursive least squares (RLS) algorithm solves the following adaptive filtering problem. Let $h_0, \ldots, h_M$ (with $h_\ell \in \mathbb{R}$) be the coefficients of an FIR filter of order M. Let $U_1, U_2, \ldots$ with $U_n \in \mathbb{R}$ be the input signal to this filter and let $Y_1, Y_2, \ldots$ be the corresponding output signal:

$$Y_n = \sum_{\ell=1}^{M} h_\ell U_{n-\ell}. \tag{4.71}$$

Now assume that the filter coefficients $\mathbf{h} = (h_0, \ldots, h_M)^\mathsf{T}$ are not known; instead, we know both the input signal $U_1, U_2, \ldots = u_1, u_2, \ldots$ and a noisy output signal $Z_1, Z_2, \ldots = z_1, z_2, \ldots$ with

$$Z_n = Y_N + W_n, \tag{4.72}$$

where $W_1, W_2, \ldots$ is white Gaussian noise with variance σ^2. Based on $u_1, u_2, \ldots$ and $y_1, y_2, \ldots$ we wish to estimate $\mathbf{h}$.

Since the unknown filter coefficients will appear as variables in the factor graph, we will denote them by capital letters: $\mathbf{H} = (H_0, \ldots, H_M)^\mathsf{T}$. We also define

$$\mathbf{u}_n \triangleq (u_n, \ldots, u_{n-M})^\mathsf{T} \tag{4.73}$$

With this notation, (4.71) becomes

$$Z_n = \mathbf{u}_n^\mathsf{T} \mathbf{H}. \tag{4.74}$$

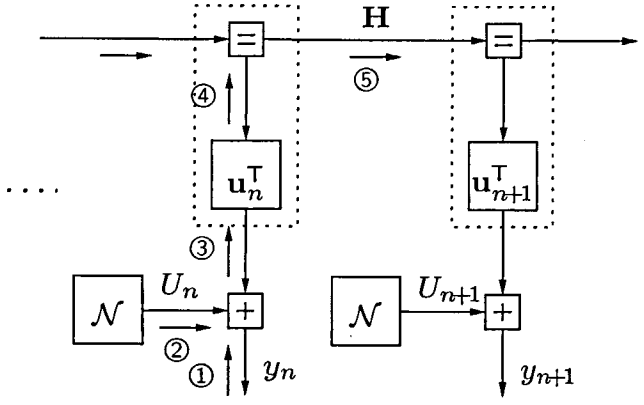


Figure 4.6: Factor graph of an FIR adaptive filter.

A factor graph of this system model is shown in Fig. 4.6. The filter coefficients $\mathbf{H}$ can be estimated by forward-only Gaussian message passing. The initial message for $\mathbf{H}$, which corresponds to $n = 0$, may be set

to $\mathbf{m}_{H,0} = \mathbf{0}$ and $\mathbf{V} =$. The rest of the messages are computed according to the following scheme.

① $m_1 = x_n$, $V_1 = 0$.

② (Table 4.1-7): $m_2 = 0$, $V_2 = 1$.

③ (Table 4.1-2): $m_3 = x_n$, $V_3 = 1$,
which can also be expressed as $\xi_3 = x_n$, $W_3 = 1$.

There are again two ways of computing the messages for the coefficient vector. One may use the weight matrix $\mathbf{W}$ or the covariance matrix $\mathbf{V}$ to represent the Gaussian messages similar to the messages in Section 4.4. When using the covariance matrix and the update rule of the composite block (Table 4.1-5) the recursion for the coefficient estimation is

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \mathbf{V}_{A,n-1}\mathbf{u}_{n-1}\frac{y_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}}\mathbf{u}_{n-1}}{1 + \mathbf{u}_{n-1}^{\mathsf{T}}\mathbf{V}_{n-1}\mathbf{u}_{n-1}} \tag{4.75}$$

$$= \hat{\mathbf{a}}_{n-1} + \mathbf{k}_n e_n \tag{4.76}$$

$$\mathbf{V}_{A,n} = \mathbf{V}_{A,n-1} - \frac{\mathbf{V}_{A,n-1}\mathbf{u}_{n-1}\mathbf{u}_{n-1}^{\mathsf{T}}\mathbf{V}_{A,n-1}}{1 + \mathbf{u}_{n-1}^{\mathsf{T}}\mathbf{V}_{n-1}\mathbf{u}_{n-1}} \tag{4.77}$$

$$= (\mathbf{I} - \mathbf{k}_n\mathbf{u}_{n-1}^{\mathsf{T}})\mathbf{V}_{A,n-1} \tag{4.78}$$

with

$$e_n = y_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}}\mathbf{u}_{n-1} \tag{4.79}$$

$$\mathbf{k}_n = \frac{\mathbf{V}_{A,n-1}\mathbf{u}_{n-1}}{1 + \mathbf{u}_{n-1}^{\mathsf{T}}\mathbf{V}_{n-1}\mathbf{u}_{n-1}}. \tag{4.80}$$

If the RLS algorithm is applied to very long blocks, it is common to multiply the covariance matrix $\mathbf{V}_{H,n}$ by some scale factor $\lambda > 1, \lambda \approx 1$, before using it as input to section $n+1$. The algorithm is thus forced to forget what it has learned in the distant past, thus it can also be used for (slowly) time-varying coefficients. Of course, a more principled approach to cope with time-varying $\mathbf{H}$ is to model such changes explicitely in the factor graph, as is sketched in Section 3.2.2.

4.6 Summary

This section summarises the conclusions about the factor graph approach for Gaussian models where all variables are jointly Gaussian distributed and, therefore, all messages are Gaussian densities.

- Different representations of **Gaussian messages** have been reviewed. Besides the representations used in [83] we introduced the representation based on the weighted mean $\boldsymbol{\xi} = \mathbf{W}\mathbf{m}$.
- **Linear building blocks** such as the equality constraint node, the addition node or the matrix multiplication node have message update rules which are manipulations of the corresponding mean vectors and covariance matrices. The existing table [83] has been extended by the update rules for the weighted mean.
- The application of the tabulated update rules (Table 4.1) has been demonstrated by means of a linear state space model. In the literature it has been shown, that the **Kalman filter** can be derived by message passing on the factor graph of a state-space model [74,83]. In this section of the thesis also the Information filter and the Kalman smoothers have been written out.
- In addition to the Kalman filter the building blocks have proved useful in two other applications. We could show that the **linear predictive coding** algorithm and the **recursive least squares** algorithm can also be expressed as Gaussian message passing on the corresponding factor graph. Above that, the similarity to the Kalman filter is eye-catching in the factor graph approach.
- Besides the application of the linear building blocks in purely Gaussian models, these building blocks may also be used in models which reach beyond Gaussianity. This is shown in the following chapter.

Seite Leer /
Blank leaf

Chapter 5

Beyond Gaussian Models

If the variables in a factor graph are jointly Gaussian, every message is Gaussian and simple update rules for linear building blocks exist (cf. Chapter 4). But practical applications of linear Gaussian models are limited.

Often the solution of the sum-product update rule (2.6) leads to functions which are non-Gaussian; or even worse, a solution to the integral in (2.6) may not exist at all. In that case, we have to resort to approximations or completely different techniques, such as gradient methods or expectation maximisation type methods. In this chapter it is shown, that many different techniques can be interpreted as message passing on the factor graph and can also be combined with one another.

In section Section 5.1 we show how tractable messages can be derived when making a tentative decision about one or more input messages to the node. In some cases it may be possible to parameterise messages which are not Gaussian, as exemplified in Section 5.2. Section 5.3 shows how an approximation of the mode of the marginal may lead to tractable messages. As mentioned, gradient methods can be incorporated into the factor graph framework, which is shown in Section 5.4. How messages can be computed by numerical methods is shown in Section 5.5. Section 5.6 gives a new view onto the expectation maximisation algorithm as message passing. Some conclusions are made in Section 5.7.

5.1 Sum-Product with Tentative Decision

In some cases, the sum-product update rule (2.6) cannot be solved analytically or the result is too complicated to be useful. In addition, messages are multiplied when propagated through an equality node (cf. (4.10)). For the outgoing message to be of the same form as the incoming messages, the messages need to be closed under multiplication. Otherwise messages become more and more complicated when propagated through equality nodes (and other nodes).

In such cases it might be appropriate to approximate the computation of the integral in (2.6) with some intuition. A reasonable remedy to make the integral in (2.6) solvable is to replace one incoming message by its mean, i.e. to ignore the variance of that message.

Zero variance approximation (tentative decision):

$$\mu_{X\to f}(x) \stackrel{!}{=} \delta(x-m_X) \tag{5.1}$$

This approach is related to the method of iterative conditional modes (ICM) [43]. In the ICM-technique every variable except one is set to its MAP-estimate, while the remaining variable is re-estimated based on the fixed values of its neighbours. In this extreme case, the algorithm often gets stuck in a local optimum. In our case, we apply the rule to set one variable temporarily to its MAP-estimate only where necessary. This combines the advantages of simple message updates with the ability to escape a local optimum.

Example 5.1. (Multiplication with unknown multiplier)

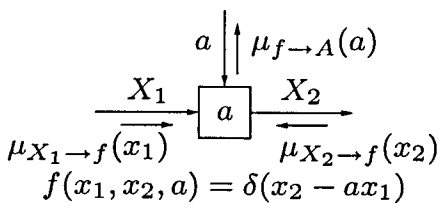


Figure 5.1: Factor graph node for the (scalar) AR coefficient estimation.

Consider the situation depicted in Fig. 5.1 where the variable X_1 is mul-

tiplied by an unknown multiplier a. We want to compute the outgoing message along edge a. The messages arriving at this node are

$$\mu_{X_1 \to f}(x_1) = \mathcal{N}(x_1 \mid m_1, v_1), \tag{5.2}$$
$$\mu_{X_2 \to f}(x_2) = \mathcal{N}(x_2 \mid m_2, v_2). \tag{5.3}$$

The exact message towards edge a would be (applying (2.6)):

$$\mu_{f \to a}(a) \propto \iint_{-\infty}^{\infty} \delta(x_2 - a x_1) \exp\left(-\frac{(x_1 - m_1)^2}{v_1}\right) \exp\left(-\frac{(x_2 - m_2)^2}{v_2}\right) dx_1 dx_2 \tag{5.4}$$

$$\propto \int_{-\infty}^{\infty} \exp\left(-\frac{(x_1 - m_1)^2}{v_1} - \frac{(a x_1 - m_2)^2}{v_2}\right) dx_1 \tag{5.5}$$

$$\propto \exp\left(-\frac{(a - \frac{m_2}{m_1})^2}{\frac{v_2 + a^2 v_1}{m_1^2}}\right). \tag{5.6}$$

Because the denominator in (5.6) depends itself on a, the message $\mu_{f \to a}(a)$ is non-Gaussian. Here it is reasonable to apply the zero variance approximation: Ignore the uncertainty in the message $\mu_{X_1 \to f}(x_1)$, i.e. set its variance v_1 to zero. The message is thus $\mu_{X_1 \to f}(x_1) = \delta(x_1 - m_1)$ and the computation of the outgoing message changes to

$$\mu_{f \to a}(a) \propto \iint_{-\infty}^{\infty} \delta(x_2 - a x_1) \delta(x_1 - m_1) \exp\left(-\frac{(x_2 - m_2)^2}{s_2}\right) dx_1 dx_2 \tag{5.7}$$

$$\propto \exp\left(-\frac{(a - \frac{m_2}{m_1})^2}{\frac{s_2}{m_1^2}}\right) \tag{5.8}$$

$$\propto \mathcal{N}\left(a \,\middle|\, \frac{m_2}{m_1}, \frac{s_2}{m_1^2}\right) \tag{5.9}$$

□

In the problem of AR coefficient estimation, the zero variance approximation works reasonably well because the variance of the state estimate (i.e. the variance of the message $\mu_{X_1 \to f}(x_1)$) is considerably smaller than the variance of the observation noise (appearing in the message $\mu_{X_2 \to f}(x_2)$ of the example), cf. Section 6.1.2.

A summary of update rules for the multiplication node derived with the technique of tentative decision is given in Table 5.1; their derivation is given in Appendix B.3. The algorithms of Section 6.1 make use of these update rules for coefficient estimation problems.

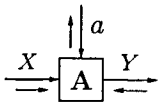
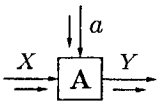
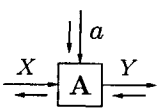
Node	Update rule
	$\mu_a(a) = \mathcal{N}(a \mid m_a, W_a)$ $m_a = m_Y^{[1]} m_X / \|m_X\|^2$ $W_a = m_X m_X^{\mathsf{T}} / V_Y^{[1,1]}$
	$m_Y = \hat{A} m_X$ $V_Y = \hat{A} V_X \hat{A}^{\mathsf{T}}$ $W_Y \overset{3}{=} \hat{A}^{-\mathsf{T}} W_X \hat{A}^{-1}$ $\xi_Y = (\hat{A} V_X \hat{A}^{\mathsf{T}})^{\#} \hat{A} V_X \xi_X \overset{1}{=} \hat{A}^{-\mathsf{T}} \xi_X$
	$m_X = (\hat{A}^{\mathsf{T}} W_Y \hat{A})^{\#} \hat{A}^{\mathsf{T}} W_Y m_Y \overset{2}{=} \hat{A}^{-1} m_Y$ $V_X \overset{3}{=} \hat{A}^{-1} V_Y \hat{A}^{-\mathsf{T}}$ $W_X = \hat{A}^{\mathsf{T}} W_Y \hat{A}$ $\xi_X = \hat{A}^{\mathsf{T}} \xi_Y$
where $\hat{A} \triangleq \begin{bmatrix} m_a^{\mathsf{T}} \\ \mathbf{I} \quad \mathbf{0} \end{bmatrix}$	
$^{\#}$ denotes the Moore-Penrose pseudoinverse ^{1} if A and V_X are positive definite ^{2} if A and W_X are positive definite ^{3} if A is invertible	

Table 5.1: Update equations for the AR state transition node.

5.2 Inverted Gamma Messages

In this section we treat the case where messages can be expressed in closed form. An example of such a message type is the inverted gamma message, which appears in variance estimation for example.

Example 5.2. (Gaussian distribution with unknown variance) Consider the situation depicted in Fig. 5.2. This graph models a Gaussian distribution over Y with unknown variance S. The sum-product

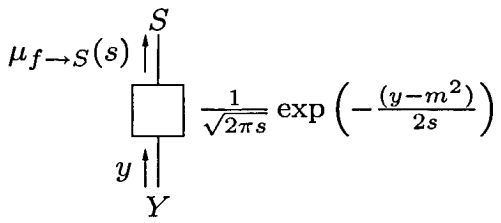


Figure 5.2: Factor graph node of a Gaussian distribution with unknown variance.

rule in the case where we observe the variable $Y = y$ is

$$\mu_{f\to S}(s) = \int_{-\infty}^{\infty} f(y,s)\, \mu_{Y\to f}(.)dy \tag{5.10}$$

$$= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(y-m)^2}{2s}\right) \delta(.-y)dy \tag{5.11}$$

$$= \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(y-m)^2}{2s}\right) \tag{5.12}$$

$$\propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{(y-m)^2}{2}\right) \tag{5.13}$$

which is an inverted gamma distribution (cf. Appendix A.1.2). □

Unfortunately, the inverted gamma distribution (and most other non-Gaussian distribution) is not very versatile. At least it is closed under multiplication, that means multiplying two inverted gamma functions yields another (scaled) inverted gamma function. This property is utilised by the equality node for Ig-messages shown in Table 5.2.

Example 5.3. (Variance estimation)

Fig. 5.3 shows the factor graph for the problem of estimating an unknown variance. For the message passing through the equality nodes, the equations of Table 5.2 are applied repeatedly. After n slices this leads to (assuming a prior $p_0(s) = \mathrm{Ig}(s \mid -1, 0)$):

$$\mu_n(s) = \mathrm{Ig}(s \mid \alpha_n, \beta_n) \tag{5.14}$$

$$\alpha_n = \sum_{\ell=1}^{n} \alpha_\ell + n - 1 = \frac{n}{2} - 1 \tag{5.15}$$

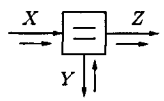
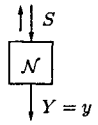
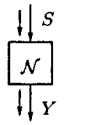
Node	Update rule
 Equality	$\mu_Z(z) = \mathrm{Ig}(z \mid \alpha_Z, \beta_Z)$ $\alpha_Z = \alpha_X + \alpha_Y + 1$ $\beta_Z = \beta_X + \beta_Y$
 Gaussian, unknown variance	$\mu_S(s) = \mathrm{Ig}(s \mid \alpha_S, \beta_S)$ $\alpha_S = -1/2$ $\beta_S = (y-m)^2/2$
 Gaussian, unknown variance	$\mu_Y(y) = \mathcal{N}(y \mid m_Y, v_Y)$ $m_Y = m$ $v_Y = \beta_S/(\alpha_S - 1)$

Table 5.2: Update equations involving inverted gamma messages.

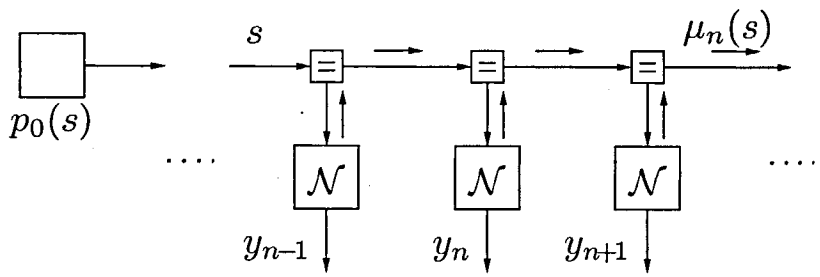


Figure 5.3: Factor graph for the variance estimation problem.

$$\beta_n = \sum_{\ell=1}^{n} \beta_\ell = \frac{1}{2} \sum_{\ell=1}^{n} (y_\ell - m)^2 \tag{5.16}$$

and the MMSE and MAP estimates after n observations are

$$\begin{aligned} \hat{s}_{\mathrm{MMSE}} &= \mathsf{E}[S|y_1, \ldots, y_n] \\ &= \frac{\beta_n}{\alpha_n - 1} = \frac{1}{n-4} \sum_{\ell=1}^{n} (y_\ell - m)^2 \qquad n > 4 \end{aligned} \tag{5.17}$$

$$\begin{aligned} \hat{s}_{\mathrm{MAP}} &= \mathsf{M}[S] \\ &= \frac{\beta_n}{\alpha_n + 1} = \frac{1}{n} \sum_{\ell=1}^{n} (y_\ell - m)^2 \end{aligned} \tag{5.18}$$

□

It is interesting, that a scaling factor $n-4$ appears in the estimator (5.17). From classic estimation theory we are used to factors n as appears in the ML estimator or the factor $n-1$ as appears in the unbiased estimator. In [125] the biased estimator with minimum MSE (not to be confused with the Bayesian MSE) has a factor $n+2$. The different estimators for variance estimation are summarised in Appendix D.

The application of inverted gamma messages for estimating the input noise variance of an AR process is presented in Section 6.2.1.

5.3 Approximated Mode

In factor graph modelling it often happens that the result of the sum-product update rule (2.6) is too complex. In some cases it is possible to approximate the mode of the posterior such that the approximation can be computed by message passing. This especially applies to constant parameters. The trick is to find the mode $\hat{\theta}$ of the a posteriori density for the parameter θ by setting

$$\left.\frac{\partial p(\theta|y)}{\partial \theta}\right|_{\theta=\hat{\theta}} \stackrel{!}{=} 0 \qquad \text{or} \qquad \left.\frac{\partial \log p(\theta|y)}{\partial \theta}\right|_{\theta=\hat{\theta}} \stackrel{!}{=} 0. \tag{5.19}$$

In the resulting equation sufficient statistics are identified and messages are designed to carry these sufficient statistics. This procedure shall be clarified in the following example.

Example 5.4. (Gaussian distribution with unknown variance)

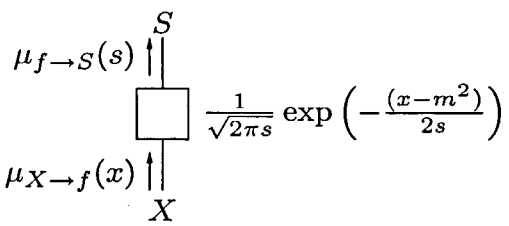


Figure 5.4: Factor graph node for the variance estimation.

The node shown in Fig. 5.4 represents a Gaussian distribution over X with unknown variance S. Computing the sum-product update rule

(2.6) for the message towards S, assuming a Gaussian input message $\mu_{X\to f}(x) = \mathcal{N}(x \mid m_X, v_X)$:

$$\mu_{f\to S}(s) = \int_{-\infty}^{\infty} f(x,s)\mu_{X\to f}(x)dx \tag{5.20}$$

$$= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x-m)^2}{2s}\right) \cdot \tag{5.21}$$

$$\frac{1}{\sqrt{2\pi v_X}} \exp\left(-\frac{(x-m_X)^2}{2v_X}\right) dx \tag{5.22}$$

$$= \frac{1}{\sqrt{2\pi(s+v_X)}} \exp\left(-\frac{(m-m_X)^2}{2(s+v_X)}\right) \tag{5.23}$$

$$= \mathrm{Ig}\left(s+v_X \;\middle|\; -\frac{1}{2}, \frac{(m-m_X)^2}{2}\right) \tag{5.24}$$

Unfortunately, (5.24) is not closed under multiplication, because of the displacement v_X. That means, multiplying functions of the form (5.24) with different parameters v_X and m_X leads to functions with exponentially increasing complexity:

$$p(s|y) = \prod_{n=1}^{N} \mathrm{Ig}\left(s+v_{X_n} \;\middle|\; -\frac{1}{2}, -\frac{(m-m_{X_n})^2}{2}\right) \tag{5.25}$$

So, trying to find sufficient statistics for (5.25) by computing the mode $\hat{s} = \operatorname{argmax} \log p(s|y)$:

$$\frac{\partial \log p(s|y)}{\partial s} = \frac{\partial}{\partial s}\left(-\frac{N}{2}\log(2\pi) - \frac{1}{2}\sum_{n=1}^{N}\log(s+v_{X_n})\right.$$

$$\left. -\frac{1}{2}\sum_{n=1}^{N}\frac{(m-m_{X_n})^2}{s+v_{X_n}}\right) \tag{5.26}$$

$$= -\frac{1}{2}\sum_{n=1}^{N}\frac{1}{s+v_{X_n}} + \frac{1}{2}\sum_{n=1}^{N}\frac{(m-m_{X_n})^2}{(s+v_{X_n})^2} \stackrel{!}{=} 0 \tag{5.27}$$

The problem in (5.27) is, that there is no simple solution for $\hat{s}$. But when we approximate all v_{X_n} by the average $v_X = \frac{1}{N}\sum_{n=1}^{N} v_{X_n}$:

$$-\frac{1}{2}\sum_{n=1}^{N}\frac{1}{s+v_X} + \frac{1}{2}\sum_{n=1}^{N}\frac{(m-m_{X_n})^2}{(s+v_X)^2} \stackrel{!}{=} 0 \tag{5.28}$$

then there exists the solution

$$\hat{s} = \frac{1}{N} \sum_{n=1}^{N} \left[(m - m_{X_n})^2 - v_X \right]. \tag{5.29}$$

Because the quantity v_X is not available in the factor graph, we replace v_X with the individual quantities v_{X_n}:

$$\hat{s} = \frac{1}{N} \sum_{n=1}^{N} \left[(m - m_{X_n})^2 - v_X \right]. \tag{5.30}$$

The sufficient statistics for this approximation are the number of terms N and the difference $(m - m_{X_n})^2 - v_{X_n}$ for every n. □

This technique is used in the algorithm for joint coefficient/variance estimation presented in Section 6.3.2.

5.4 Gradient Methods as Message Passing

Gradient methods are often used to search for an extremum of a function to optimise, such as the likelihood function of a probabilistic model. In this section we show that gradient methods can be smoothly integrated into the factor graph framework. The material presented in this section is based on joint work with Dauwels et. al. [23].

Instead of computing the complete a posteriori distribution of some parameter, one may by interested in the mode of the posterior distribution only. The mode of the posterior distribution corresponds to a maximum a posteriori (MAP) estimate. If a closed-form solution for the MAP estimate does not exist, it might be found iteratively by a gradient method. The gradient method (gradient descent or hill climbing) can be viewed as message passing on the corresponding factor graph.

Fig. 5.5 shows one node which depends on the parameter θ which we want to estimate via hill climbing. The message from the node f towards edge θ is

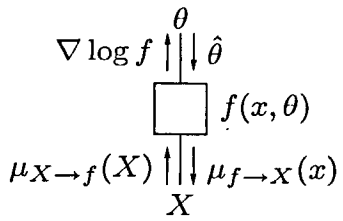


Figure 5.5: Factor graph node for the gradient update rule.

Gradient Node Update Rule:

$$\nabla \log f(\hat{\theta}) = \nabla_\theta \log \mu_{f\to\theta}(\theta)\Big|_{\theta=\hat{\theta}} \tag{5.31}$$

$$= \nabla_\theta \log \int_{\mathcal{D}_x} f(x,\hat{\theta})\, \mu_{X\to f}(x)\, dx \tag{5.32}$$

where $\mu_{f\to\theta}(\theta)$ is the standard sum-product message (2.6) out of node f towards edge θ. Contrary to the sum-product message, the gradient message $\nabla \log f(\hat{\theta})$ is a value and not a function.

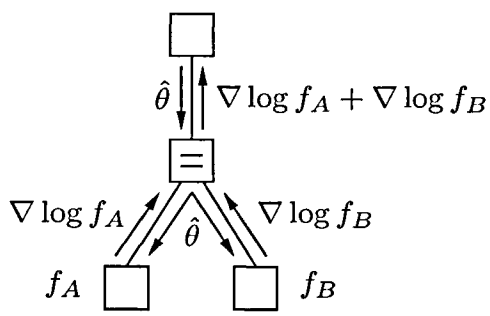


Figure 5.6: Factor graph node for the generic gradient descent.

Gradient messages from different nodes arriving at an equality node are just added as can be seen in Fig. 5.6. The new estimate $\hat{\theta}^{(k+1)}$ is found by updating with an appropriate step size λ:

Global Gradient Descent Step:

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda \cdot \sum_{n=1}^{N} \nabla \log f_n(\hat{\theta}) \tag{5.33}$$

The proofs of (5.31)-(5.33) can be found in Appendix B.1.

Example 5.5. (Variance estimation)
We want to estimate the variance of a Gaussian source. The function of the node of Fig. 5.5 is

$$f(x,v) = \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{(x-m)^2}{2v}\right) \tag{5.34}$$

where $m \in \mathbb{R}$ is the constant mean and the parameter $\theta = v$, the unknown variance. Assuming a Gaussian message $\mu_{X\to f}(x) = \mathcal{N}(x \mid m_X, v_X)$ and applying (5.32) gives the upward message

$$\begin{aligned}\nabla \log f(\hat{v}) = \nabla_v \log \int_{\mathcal{D}_v} & \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{(x-m)^2}{2v}\right) \\ & \frac{1}{\sqrt{2\pi v_X}} \exp\left(-\frac{(x-m_X)^2}{2v_X}\right) dx\Big|_{v=\hat{v}} \qquad (5.35)\end{aligned}$$

$$= \frac{(m-m_X)^2 - (\hat{v}+v_X)}{2(\hat{v}+v_X)^2}. \tag{5.36}$$

Note that (5.36) is real number; it is not a function as in the case of standard sum-product messages. □

Table 5.3 summarises the gradient update rules for nodes used in Section 6.1.4.

5.5 Particle Methods

The key idea of the techniques introduced in this section is to represent a function by a list of (weighted) samples of that function. Messages in the factor graph are represented as such lists.

Literature

Tutorials for particle methods are [5, 7, 30, 32–34, 88]. Some improvements to the standard particle filter are given in [18,35]. Online particle filter are treated in [6, 51, 76, 79]; The estimation of static parameters

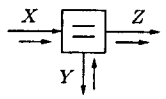
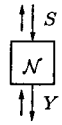
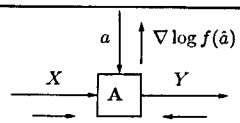
Node	Update rule
 Equality	$\nabla \log f(\hat{z}) = \nabla \log f(\hat{x}) + \nabla \log f(\hat{y})$
 Gaussian, unknown variance	$\nabla \log f(\hat{s}) = \dfrac{(m - m_Y)^2 - (\hat{v} + v_Y)}{2(\hat{v} + v_Y)^2}$
 AR state transition	$\nabla \log f(\hat{a}) = -2m_X(m_Y^{[1]} - \hat{a}^{\mathsf{T}} m_X)$

Table 5.3: Update equations involving gradient messages.

in [82, 127]; The concept of nonparametric belief propagation is introduced in [60, 61, 128, 131]. For Gaussian particle filtering see [70, 71]. Particle filter are applied to speech signals in [39, 117, 133]. Gaussian sum approximation is shown [2]. Markov chain Monte Carlo methods are treated in [19, 106, 112, 132].

When messages are (scaled) probability distributions, they have a cumulative function

$$F_X(x) = P_X(X \leq x) = \int_{-\infty}^{x^+} p_X(x)\, dx. \tag{5.37}$$

The main idea of particle methods is to approximate the cumulative of a message by discretisation:

$$F_X(x) \approx \int_{-\infty}^{x^+} \sum_{i=1}^{M} \rho^{(i)} \delta(x - x^{(i)})\, dx \tag{5.38}$$

where the weights $\rho^{(i)}$ are normalised, i.e. $\sum_{i=1}^{M} \rho^{(i)} = 1$. A particle message comprises the discretisation points $x^{(i)}$ (also called particles)

and their weights $\rho^{(i)}$, hence

$$\mu_X(x) = \frac{d}{dx} F_X(x) \approx \left\{ x^{(i)}, \rho^{(i)} \right\}_{i=1}^{M} \tag{5.39}$$

is a list of weighted samples of the corresponding distribution function. Such particle messages can propagate through different factor graph nodes, as the following example demonstrates.

Example 5.6. (Equality node)

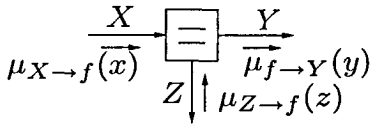


Figure 5.7: Factor graph node for the demonstration of particle messages; Equality.

Given the graph of Fig. 5.7 we want to compute the outgoing message $\mu_{f\to Y}(y) = \frac{d}{dy} F_Y(y)$. We assume the message $\mu_{X\to f}(x)$ is a particle message and the message $\mu_{Z\to f}(z)$ may be arbitrary.

$$F_Y(y) = \int_{-\infty}^{y^+} \mu_{f\to Y}(y)\, dy \tag{5.40}$$

$$= \int_{-\infty}^{y^+} \iint_{-\infty}^{\infty} f(x,y,z)\mu_{Z\to f}(z)\mu_{X\to f}(x)\, dzdx\, dy \tag{5.41}$$

$$= \int_{-\infty}^{y^+} \iint_{-\infty}^{\infty} \delta(x-y)\delta(z-y)\mu_{Z\to f}(z)\frac{d}{dx}F_X(x)\, dzdx\, dy \tag{5.42}$$

$$= \int_{-\infty}^{y^+} \mu_{Z\to f}(y)\frac{d}{dx}F_X(y)\, dy \tag{5.43}$$

$$= \int_{-\infty}^{y^+} \mu_{Z\to f}(y)\sum_{i=1}^{M}\rho_X^{(i)}\delta(y-x^{(i)})\, dy \tag{5.44}$$

$$= \int_{-\infty}^{y^+} \sum_{i=1}^{M}\rho_Y^{(i)}\delta(y-y^{(i)})\, dy. \tag{5.45}$$

The message $\mu_{f\to Y}(y)$ is again of the form (5.38) and (5.39) with parameters

$$y^{(i)} = x^{(i)} \qquad \rho_Y^{(i)} = \rho_X^{(i)} \cdot \mu_{Z\to f}(y^{(i)}). \tag{5.46}$$

□

Another building block for particle messages is the addition node as depicted in Fig. 5.8, where $\mu_{X\to f}(x)$ is a particle message and $\mu_{U\to f}(u)$ is arbitrary.

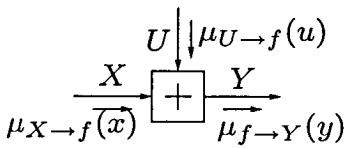


Figure 5.8: Factor graph node for the demonstration of particle messages; Addition.

Applying the sum-product update rule (2.6) for the addition node in Fig. 5.8 leads to the output message

$$\mu_{f\to Y}(y) = \sum_{i=1}^{M} \rho_X^{(i)} \mu_{U\to f}(y + x^{(i)}) \tag{5.47}$$

which is a mixture density and not simply a list of particles. In practice, we sample from $\mu_{U\to f}(y)$ in (5.47). The resulting update rules are summarised in Table 5.4.

Computing the particle messages of a chain of nodes as shown in Fig. 5.9 amounts to sequential importance sampling, a standard technique in particle filtering [32]:

$$x_{n+1}^{(i)} = x_n^{(i)} + u_n \tag{5.48}$$

$$\rho_{n+1}^{(i)} = \rho_n^{(i)} \cdot \mu_{Z_n}(x_n^{(i)}) \tag{5.49}$$

with

$$u_n \sim \mathcal{N}(u_n \mid 0, \sigma_U^2) \tag{5.50}$$

Repeated application of the update rules given in Table 5.4 may lead to messages where all but one particle have zero weight. This phenomenon is called degeneracy [30]. A remedy proposed in the literature

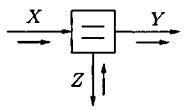
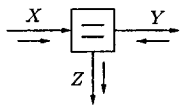
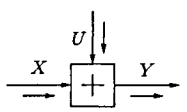
Node	Update rule
 Equality particle message/ arbitrary message	$F_Y(y) = \int_{-\infty}^{y^+} \sum_{i=1}^{M} \rho_Y^{(i)} \delta(y - y^{(i)})\, dy$ $y^{(i)} = x^{(i)}$ $\rho_Y^{(i)} = \rho_X^{(i)} \cdot \mu_Z(y^{(i)})$.
 Equality particle message/ particle message	$\mu_Z(z) = \mathcal{N}(z \mid m_Z, v_Z)$ $m_Z = v_Z\,(m_X/v_X + m_Y/v_Y)$ $v_Z = v_X v_Y/(v_X + v_Y)$ $m_X = \sum_{i=1}^{M} \rho_X^{(i)} x^{(i)}$ $v_X = \sum_{i=1}^{M} \rho_X^{(i)} (m_X - x^{(i)})^2$ m_Y, v_Y likewise
 Addition particle message/ arbitrary message	$F_Y(y) = \int_{-\infty}^{y^+} \sum_{i=1}^{M} \rho_Y^{(i)} \delta(y - y^{(i)})\, dy$ $y^{(i)} \sim \mu_U(u) + x^{(i)}$ $\rho_Y^{(i)} = \rho_X^{(i)}$

Table 5.4: Update equations involving particle messages.

is to include a resampling step after every particle message update. During resampling a new list is created where particles from the old list are transfered randomly according to their weight.

Another problem appears when the noise U_n added to the particle list is zero or very small. During resampling some particles may be chosen more often and others may never be chosen. This leads to another problem called sample impoverishment where all particles will eventually coincide. Sample impoverishment is especially inevitable when estimating constant parameters by particle messages.

Strategies to overcome the problems of degeneracy and sample impoverishment used in this thesis are resampling only when necessary [7,34], auxiliary resampling [32] and the shrinking technique proposed in [82].

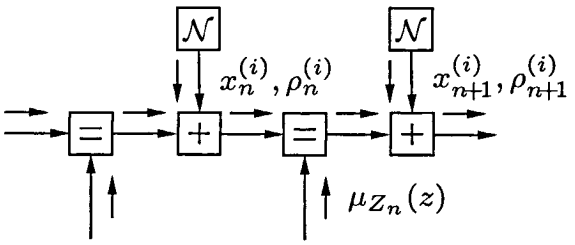


Figure 5.9: Factor graph of the random walk model.

5.6 Expectation Maximisation as Message Passing

The expectation maximisation (EM) algorithm is a powerful technique for maximum likelihood estimation in complex models. The problem in parameter estimation in complex models is that there usually are many unknown variables. Integrating them out, as is done in a Bayesian approach, is most often not feasible. Instead, in the EM algorithm, those hidden variables are replaced by their expected values. By alternating between the estimation of the parameters and the computation of the expectations it is guaranteed to find a local optimum.

In this chapter it is shown how the EM algorithm can be computed as message passing on the corresponding factor graph. Moreover, non-trivial a priori models for the parameters can easily be incorporated. The message passing view opens up the freedom of choosing message types and update schedules. Thus, variants of the EM algorithm, such as Gradient-EM or Online-EM, can be derived in a straightforward way. Finally, a local EM update rule is proposed, which combines ideas from sum-product and EM.

Literature

The EM algorithm was originally introduced in [29] (sometimes referred to as "DLR"-paper according to the initials of the authors); For tutorials refer to [10, 96, 100]; Convergence properties are elaborated on in [140, 141]; For different interpretations of the EM algorithm consult: [25] (lower bound maximisation), [109] (double minimisation), [126] (majorisation), [27] (iterative conditional estimation); EM in connection

with graphical models: [12,22,36,57,77,130]; An application to time-series modelling can be found in [118], and an application to speech enhancement in [48, 49] and [80]; For online EM look into [4, 37] exemplarily. Incremental EM, i.e. incomplete E-step [107], SAGE [38].

5.6.1 Introduction to classic EM

The maximum-likelihood (ML) estimator is found by maximising the (log-)likelihood function with respect to the parameters $\boldsymbol{\theta}$ given the observations $\mathbf{z}$.

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \log p(\mathbf{z}|\boldsymbol{\theta}). \tag{5.51}$$

For most stochastic models considered here, not every variable is known. We, therefore, have to get rid of the unknown variables, which is usually done by integration. The ML estimator for the parameter vector $\boldsymbol{\theta}$ is then

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \log \int_{\mathcal{D}_{\mathbf{x}}} p(\mathbf{z}, \mathbf{x} \mid \boldsymbol{\theta})\, d\mathbf{x} \tag{5.52}$$

where $p(\mathbf{z}, \mathbf{x} \mid \boldsymbol{\theta})$ is called the complete-data likelihood function in comparison to the incomplete-data likelihood function $p(\mathbf{z}|\boldsymbol{\theta})$.

The maximisation of the log of the integral in (5.52) is often hard to compute. In the EM algorithm the log-integral in (5.52) is replaced by an expectation of the log, which is easier to compute:

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathsf{E}\left[\log p(\mathbf{z}, \mathbf{x} \mid \boldsymbol{\theta}) \,\middle|\, \mathbf{z}, \boldsymbol{\theta}^{(k)}\right] \tag{5.53}$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \int_{\mathcal{D}_{\mathbf{x}}} p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}^{(k)}) \log p(\mathbf{z}, \mathbf{x} \mid \boldsymbol{\theta})\, d\mathbf{x} \tag{5.54}$$

The expectation in (5.53) is a lower bound to the log-likelihood function $p(\mathbf{z}|\boldsymbol{\theta})$ [25]. In the maximisation step only the mode of this bound is computed; to find the maximum of the likelihood function itself the expectation step and the maximisation step have to be carried out alternatingly, which leads to the classic formulation of the EM algorithm:

Expectation maximisation (EM) algorithm:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k)}) = \mathsf{E}\left[\log p(\mathbf{z}, \mathbf{x} \mid \boldsymbol{\theta}) \,\middle|\, \mathbf{z}, \boldsymbol{\theta}^{(k)}\right] \tag{5.55}$$

$$\boldsymbol{\theta}^{(k+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k)}) \tag{5.56}$$

The different roles of $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{(k)}$ in (5.55) and (5.56) may be confusing at first sight. The first step (5.55) of the EM algorithm (E-step) is to find the expected value of the complete-data log-likelihood function $\log p(\mathbf{z}, \mathbf{x} \mid \boldsymbol{\theta})$, which is a function of $\boldsymbol{\theta}$, with respect to the unknown data $\mathbf{x}$ given the observed data $\mathbf{z}$ and the current parameter estimates $\boldsymbol{\theta}^{(k)}$. The result of the E-step is function of $\boldsymbol{\theta}$. The second step (5.56) of the EM algorithm (M-step) is the maximisation of the expectation of the log-likelihood function with respect to $\boldsymbol{\theta}$. The algorithm proceeds by iterating the E- and M-step until convergence or the allotted time is over. In each iteration the log-likelihood is increased and the algorithm is guaranteed to converge to a local maximum of the likelihood function. Convergence properties are elaborated on in [140, 141].

5.6.2 Introduction to messages passing EM

In the literature, there have been two connections between the Expectation Maximisation (EM) algorithm and graphical models. First, the EM algorithm has been used for estimating the parameters of a graphical model in a stand-alone way. Second, graphical models have been used to compute the E-step of the EM algorithm.

In this thesis we show how the EM algorithm (both the E-step and the M-step) can be expressed as message passing on the corresponding factor graph. Earlier work on this topic was done by Eckford [36]. We build on this work by showing how the EM algorithm can be computed by local message updates on the factor graph. The material presented in this section is based on joint work with Dauwels et. al. [22].

Assume a stochastic model with the following factorised probability density function:

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \prod_i p_i(\mathbf{x}_i|\mathbf{z}, \boldsymbol{\theta}). \tag{5.57}$$

where the $\mathbf{x}_i$'s are subsets of $\mathbf{x}$. In the E-step of the EM algorithm (5.55) the following quantity is computed:

$$
\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k)}) &= \mathsf{E}\left[\log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})\right] && (5.58)\\
&= \mathsf{E}\left[\log \prod_i p_i(\mathbf{x}_i|\mathbf{z}, \boldsymbol{\theta})\right] && (5.59)\\
&= \mathsf{E}\left[\sum_i \log p_i(\mathbf{x}_i|\mathbf{z}, \boldsymbol{\theta})\right] && (5.60)\\
&= \sum_i \mathsf{E}\left[\log p_i(\mathbf{x}_i|\mathbf{z}, \boldsymbol{\theta})\right] && (5.61)\\
&= \sum_i h_i(\boldsymbol{\theta}) && (5.62)
\end{aligned}
$$

where the expectation is with respect to the density $p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}^{(k)})$. The individual terms of the sum in (5.61) can be viewed as messages $h_i(\boldsymbol{\theta})$ computed by individual nodes locally. Section 5.6.3 gives the formal definition of those messages.

The M-step comprises the maximisation of (5.61) w.r.t. $\boldsymbol{\theta}$ and leads to a new estimate $\boldsymbol{\theta}^{(k+1)}$ which is sent back to the nodes for the next iteration. The formal definition can be found in Section 5.6.4.

5.6.3 Computing the h-message

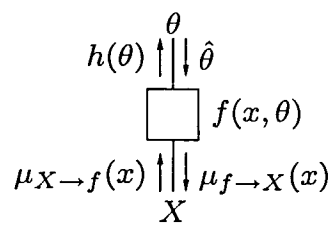


Figure 5.10: Factor graph node for the generic EM update rule.

The update rule for a generic node $f(x, \theta)$ (Fig. 5.10) is

Message passing EM update rule: **"E-step"**

$$\begin{aligned} h(\theta) &= \mathsf{E}_p[\log f(X,\theta)] \\ &= \int_{\mathcal{D}_X} p(x|\hat{\theta}^{(k)}) \log f(x,\theta) dx \end{aligned} \tag{5.63}$$

where E_p denotes the expectation with respect to the distribution

$$p(x|\hat{\theta}^{(k)}) = \gamma \mu_{X\to f}(x)\mu_{f\to X}(x) = \gamma \mu_{X\to f}(x) f(x,\hat{\theta}^{(k)}) \tag{5.64}$$

with a proper scaling factor γ. Note that the observation y does not appear in (5.63) since it is implied in the computation of the message $\mu_{X\to f}(x)$.

Example 5.7. (Estimating the mean of a Gaussian source)
We want to estimate the mean of a scalar Gaussian source. The node function for this case is

$$f(x,m) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x-m)^2}{2s}\right) \tag{5.65}$$

where $s \in \mathbb{R}^+$ denotes the (fixed) variance and $m \in \mathbb{R}$ is the parameter to be estimated. The h-message (5.63) is computed as follows

$$h(m) = \int_{\mathcal{D}_X} p(x|\hat{m}^{(k)}) \log f(x,m) dx \tag{5.66}$$

$$= \int_{\mathcal{D}_X} p(x|\hat{m}^{(k)}) \left(-\frac{1}{2}\log(2\pi s) - \frac{(x-m)^2}{2s}\right) dx \tag{5.67}$$

$$= C - \frac{1}{2s}\left(m^2 - 2m\mathsf{E}_p[X]\right) \tag{5.68}$$

with

$$C = -\frac{1}{2}\log(2\pi s) - \frac{\mathsf{E}_p[X^2]}{2s}. \tag{5.69}$$

(5.68) is a quadratic form in m, so it is convenient to use the message $e^{h(m)}$ instead of $h(m)$

$$e^{h(m)} \propto \mathcal{N}\Big(m \,\Big|\, \mathsf{E}_p[X], s\Big) \tag{5.70}$$

hence, $e^{h(m)}$ itself becomes Gaussian. □

In this example, the h-message (or its exponential form) has a nice form and can easily be processed by the Gaussian sum-product nodes of Table 4.1.

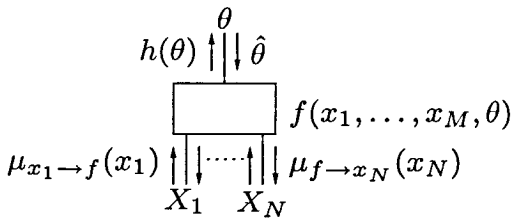


Figure 5.11: Factor graph node for the EM update rule for nodes connected to more than one hidden node.

In general, the node can be connected to more than one hidden variable as in Fig. 5.11. In that case, the message passing EM update rule (5.63) is extended in a straightforward way:

$$h(\theta) = \mathsf{E}_p[\log f(X_1, \ldots, X_N, \theta)] \tag{5.71}$$

$$= \int_{\mathcal{D}_\mathcal{X}} \cdots \int p(x_1, \ldots, x_N | \hat{\theta}^{(k)}) \log f(x_1, \ldots, x_N, \theta) dx_1 \cdots dx_N \tag{5.72}$$

$$= \frac{1}{\gamma} \int_{\mathcal{D}_\mathcal{X}} \cdots \int f(x_1, \ldots, x_N, \hat{\theta}^{(k)}) \cdot \mu_{X_1 \to f}(x_1) \cdots \mu_{X_N \to f}(x_N) \log f(x_1, \ldots, x_N, \theta) dx_1 \cdots dx_N \tag{5.73}$$

with a proper scaling factor

$$\gamma = \int_{\mathcal{D}_\mathcal{X}} \cdots \int f(x_1, \ldots, x_N, \hat{\theta}^{(k)}) \cdot \mu_{X_1 \to f}(x_1) \cdots \mu_{X_N \to f}(x_N) dx_1 \cdots dx_N. \tag{5.74}$$

5.6.4 Computing the $\hat{\theta}$-message

The message downwards (Fig. 5.10) represents the new estimate $\hat{\theta}$ of the parameter θ. (In the EM literature the iteration index $^{(k)}$ is always given. In the message passing view it looses its meaning. The freedom of choosing a certain schedule gives us much more flexibility to design the parameter update.)

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmax}}\, h(\theta) = \underset{\theta}{\operatorname{argmax}}\, e^{h(\theta)} \tag{5.75}$$

Example 5.8. (Mean of a Gaussian Source continued)
Solving (5.75) for the message (5.70) gives

$$\hat{m} = \operatorname*{argmax}_m e^{h(m)} = \operatorname*{argmax}_m \mathcal{N}\Big(m \;\Big|\; \mathsf{E}_p[X], s\Big) = \mathsf{E}_p[X] \tag{5.76}$$

which is intuitive. □

5.6.5 Non-trivial a priori models

In contrast to most standard accounts of the EM algorithm, we permit more general a priori models for the parameters. In this case the maximisation includes the a priori model $f_A(\theta_1, \ldots, \theta_n)$:

Maximisation: **"M-step"**

$$\begin{aligned}(\hat{\theta}_1, \ldots, \hat{\theta}_n)^{\mathsf{T}} &= \operatorname*{argmax}_{\hat{\theta}_1,\ldots,\hat{\theta}_n} \left(\log f_A(\theta_1, \ldots, \theta_n) + \sum_{\ell=1}^{n} h_\ell(\theta_\ell) \right) \\ &= \operatorname*{argmax}_{\hat{\theta}_1,\ldots,\hat{\theta}_n} \left(f_A(\theta_1, \ldots, \theta_n) \cdot \prod_{\ell=1}^{n} e^{h_\ell(\theta_\ell)} \right) \end{aligned} \tag{5.77}$$

If f_A itself has a nice factorisation, then (5.77) can be computed by the standard max-sum or max-product algorithm (which are certain instances of the summary-propagation algorithm). In addition, if all $e^{h_\ell(\theta_\ell)}$-terms are Gaussian, the max-product and the sum-product scheme are equivalent [83]. Therefore, the Kalman filter can be used to solve (5.77), not to be confused with the Kalman filter of the E-step. We can use Gaussian sum-product message passing and the standard update rules given in table 4.1 for this purpose. Two different models for f_A are discussed in the following sections.

Constant parameter

The situation at hand is depicted in Fig. 5.12. In this case $f_A(\theta)$ is uniform and drops out of the maximisation. The equality constraints

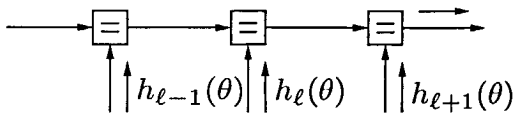


Figure 5.12: Chain of equality nodes, which models a constant parameter.

make sure, that every θ_n for time n is in fact the same. Hence, (5.77) reduces to

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \left(\sum_{\ell=1}^{n} h_\ell(\theta) \right) = \operatorname*{argmax}_{\theta} \left(\prod_{\ell=1}^{n} e^{h_\ell(\theta)} \right) \tag{5.78}$$

Example 5.9. (Mean estimation continued)
The h-messages for the mean estimation problem are

$$e^{h_\ell(m)} \propto \mathcal{N}(m \mid \mathsf{E}[X_\ell], s) \tag{5.79}$$

and the maximisation is therefore

$$\hat{m} = \operatorname*{argmax}_{m} \prod_{\ell=1}^{n} \mathcal{N}(m \mid \mathsf{E}[X_\ell], s) \tag{5.80}$$

$$= \operatorname*{argmin}_{m} \sum_{\ell=1}^{n} (\mathsf{E}[X_\ell] - m)^2 \tag{5.81}$$

$$= \frac{1}{n} \sum_{\ell=1}^{n} \mathsf{E}[X_\ell] \tag{5.82}$$

The product in (5.80) can be computed by standard Gaussian equality nodes with update rules from Table 4.1. □

Note that the maximisation need not be done at the very end (either to the left or right of the graph of Fig. 5.12). When having computed sum-product messages in both directions along every edge, the maximum $\hat{m}$ is available on every edge as the mode of the total message at this edge.

Random walk model

The situation at hand is depicted in Fig. 5.13. The a priori function f_A

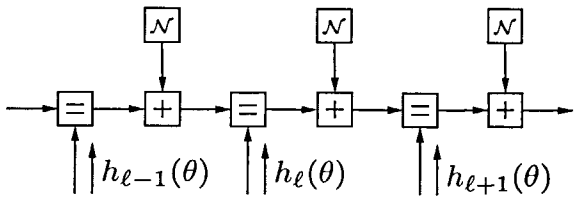


Figure 5.13: A priori model for random walk.

is here

$$f_A(\theta_1, \ldots, \theta_n) = \prod_{\ell=2}^{n} \mathcal{N}(\theta_\ell - \theta_{\ell-1} \mid 0, s)\,. \tag{5.83}$$

where s is the variance of the Gaussian random deviation between two sucessive parameters. The maximisation is therefore

$$(\hat{\theta}_1, \ldots, \hat{\theta}_n)^{\mathsf{T}} = \operatorname*{argmax}_{\hat{\theta}_1, \ldots, \hat{\theta}_n} \left(f_A(\theta_1, \ldots, \theta_n) \cdot \prod_{\ell=1}^{n} e^{h_\ell(\theta_\ell)} \right) \tag{5.84}$$

$$= \operatorname*{argmax}_{\hat{\theta}_1, \ldots, \hat{\theta}_n} \left(\prod_{\ell=2}^{n} f(\theta_{\ell-1}, \theta_\ell) \cdot \prod_{\ell=1}^{n} e^{h_\ell(\theta_\ell)} \right) \tag{5.85}$$

$$= \operatorname*{argmax}_{\hat{\theta}_1, \ldots, \hat{\theta}_n} \left(\prod_{\ell=2}^{n} e^{\left(-\frac{(\theta_{\ell-1} - \theta_\ell)^2}{2s}\right)} \cdot \prod_{\ell=1}^{n} e^{h_\ell(\theta_\ell)} \right) \tag{5.86}$$

The term in brackets in (5.86) is a product of (scaled) Gaussian functions. This is such a case where maximisation and integration leads to the same result (cf. Appendix A.3). Hence, max-product and sum-product are equivalent here. Consequently, the graph in Fig. 5.13 comprises only Gaussian nodes of table Table 4.1.

5.6.6 Schedule

When choosing a specific schedule, as shown in Section 6.3.3, the resulting algorithm resembles standard EM. One advantage of representing EM as message passing is the freedom in choosing the schedule differently from the standard processing. First simulations using a per-section scheme showed somewhat faster convergence during the first five to ten iterations. Afterwards, both schedules are comparable. The effects of different schedules has to be analysed in further endeavours.

5.6.7 Clustering

Computing the message passing EM update rule (5.63) for a node which function is a Dirac delta function leads to a singularity problem. The expectation of the $\log \delta(.)$ is again a delta function. Hence, the h-message leaving such a node expresses complete certainty about the parameter, which is not what we want.

To circumvent the appearance of h-messages which are delta functions one may group different nodes and derive the EM update rule for this combined node. The idea is to include nodes which have softer node functions.

For example, in autoregression coefficient estimation (cf. Fig. 5.14) the multiplication node has a node function which is a Dirac delta function. Grouping the multiplication node with the node representing the input noise and the addition node leads to a useful h-message which is shown in Table 5.5. A detailed derivation is given in Appendix E.

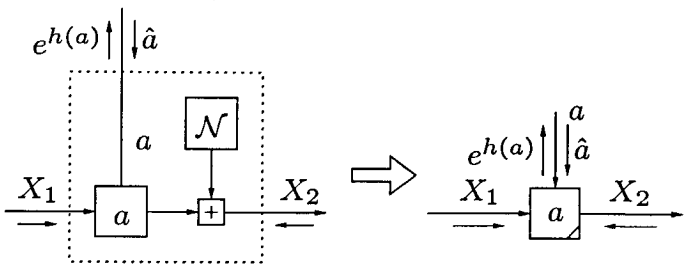


Figure 5.14: Factor graph of the state transition node for the autoregressive model made by clustering.

Building even bigger clusters may also improve the convergence rate of the message passing EM algorithm as proven in [38].

5.6.8 Table of EM update rules

In this section, the message passing EM update rule (5.63) is worked out as in Example 5.7 for the most common nodes and groupings of nodes. All h-messages (or e^h-messages) are collected in Table 5.5, the corresponding derivations can be found in Appendix E.

Table 5.5: EM Update equations for standard building blocks.

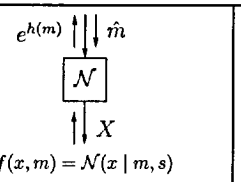
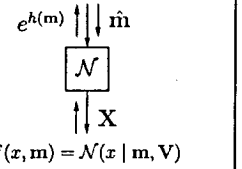
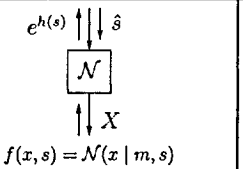
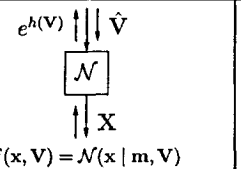
	Graph	Node	EM update rule	
1		Gaussian, unknown mean, scalar	$e^{h(m)} \propto \mathcal{N}\left(m \,\middle\|\, \mathsf{E}[X], s\right)$	$X, m \in \mathbb{R}$ $s \in \mathbb{R}_0^+$
2		Gaussian, unknown mean	$e^{h(\mathbf{m})} \propto \mathcal{N}\left(\mathbf{m} \,\middle\|\, \mathsf{E}[\mathbf{X}], \mathbf{V}\right)$	$\mathbf{X}, \mathbf{m} \in \mathbb{R}^n$ $\mathbf{V} \in \mathbb{R}^{n \times n}$ $\mathbf{V} \succeq 0$
3		Gaussian, unknown variance	$e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle\|\, -\frac{1}{2}, \frac{\mathsf{E}[X^2]-2m\mathsf{E}[X]+m^2}{2}\right)$	$X \in \mathbb{R}$ $s \in \mathbb{R}_0^+$
4		Gaussian, unknown variance	$e^{h(\mathbf{V})} \propto \exp\left(-\mathsf{E}\left[(\mathbf{X}-\mathbf{m})^{\mathsf{T}}\mathbf{V}^{-1}(\mathbf{X}-\mathbf{m})\right]\right)$	$\mathbf{X}, \mathbf{m} \in \mathbb{R}^n$ $\mathbf{V} \in \mathbb{R}^{n \times n}$ $\mathbf{V} \succeq 0$
5		Identity covariance matrix	$e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle\|\, \frac{n-2}{2}, \frac{1}{2}\mathsf{E}\left[(\mathbf{X}-\mathbf{m})^{\mathsf{T}}(\mathbf{X}-\mathbf{m})\right]\right)$	$\mathbf{X}, \mathbf{m} \in \mathbb{R}^n$ $\mathbf{V} = \mathbf{I}s$ $s \in \mathbb{R}_0^+$

Table 5.5: (continued)

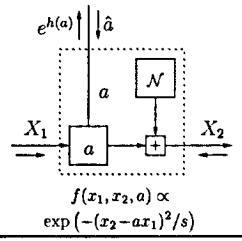
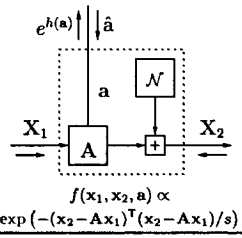
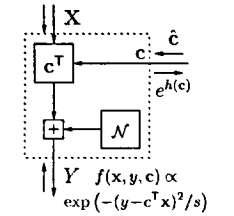
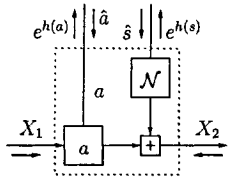
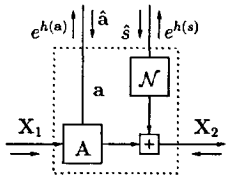
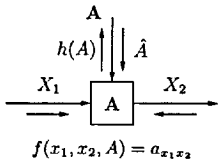
	Graph	Node	EM update rule	
6		Diagonal covariance matrix	$e^{h(s)} \propto \prod_{\ell=1}^{n} \mathrm{Ig}\left(s_\ell \,\middle|\, -\frac{1}{2}, \frac{1}{2}\mathsf{E}[(X_\ell - m_\ell)^2]\right)$	$\mathbf{X}, \mathbf{m} \in \mathbb{R}^n$ $\mathbf{V} = \mathrm{diag}\,(\mathbf{s})$ $\mathbf{s} \in \mathbb{R}^n$
7		Scalar multiplication	$e^{h(a)} \propto \mathcal{N}_W\left(a \,\middle|\, \frac{\mathsf{E}[X_1 X_2]}{\mathsf{E}[X_1^2]}, \frac{\mathsf{E}[X_1^2]}{s}\right)$	$X_2 \in \mathbb{R}$ $a \in \mathbb{R}$
8		Auto-regression	$e^{h(\mathbf{a})} \propto \mathcal{N}_W\left(\mathbf{a} \,\middle|\, \mathsf{E}[\mathbf{X}_1\mathbf{X}_1^\mathsf{T}]^{-1}\mathsf{E}[\mathbf{X}_1 X_2], \frac{\mathsf{E}[\mathbf{X}_1\mathbf{X}_1^\mathsf{T}]}{s}\right)$	$\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^n$ $\mathbf{a} \in \mathbb{R}^n$ $\mathbf{A} = [\mathbf{a}^\mathsf{T}; \mathbf{I}\,\mathbf{0}]$
9		Inner vector product	$e^{h(\mathbf{c})} \propto \mathcal{N}_W\left(\mathbf{c} \,\middle|\, \mathsf{E}[\mathbf{X}\mathbf{X}^\mathsf{T}]^{-1}\mathsf{E}[\mathbf{X}Y], \frac{\mathsf{E}[\mathbf{X}\mathbf{X}^\mathsf{T}]}{s}\right)$	$\mathbf{c} \in \mathbb{R}^n$ $Y \in \mathbb{R}$

Table 5.5: (continued)

	Graph	Node	EM update rule	
10		Joint coefficient/variance estimation scalar	$e^{h(a)} \propto \mathcal{N}_W\left(a \mid \frac{\mathsf{E}[X_1X_2]}{\mathsf{E}[X_1^2]}, \frac{\mathsf{E}[X_1^2]}{\hat{s}}\right)$ $e^{h(s)} \propto \mathrm{Ig}\left(s \mid -\frac{1}{2}, \frac{\mathsf{E}[(X_2-\hat{a}X_1)^2]}{2}\right)$	$X_2 \in \mathbb{R}$ $a \in \mathbb{R}$ $s \in \mathbb{R}_0^+$
11		Joint coefficient/variance estimation	$e^{h(\mathbf{a})} \propto \mathcal{N}_W\left(a \mid \mathsf{E}[\mathbf{X}_1\mathbf{X}_1^\mathsf{T}]^{-1}\mathsf{E}[\mathbf{X}_1X_2], \frac{\mathsf{E}[\mathbf{X}_1\mathbf{X}_1^\mathsf{T}]}{\hat{s}}\right)$ $e^{h(s)} \propto \mathrm{Ig}\left(s \mid -\frac{1}{2}, \frac{\mathsf{E}[(\mathbf{X}_2-\hat{\mathbf{A}}\mathbf{X}_1)^\mathsf{T}(\mathbf{X}_2-\hat{\mathbf{A}}\mathbf{X}_1)]}{2}\right)$	$\mathbf{X}_1, \mathbf{X}_2, \mathbf{a} \in \mathbb{R}^n$ $s \in \mathbb{R}_0^+$ $X_2 = \mathbf{X}_2^{[1]}$
12		Finite state machine	$h(\mathbf{A}) = \sum_{x_1,x_2} p(x_1,x_2) \log a_{x_1,x_2}$	$X_1, X_2 \in \{0,\ldots,n-1\}$ $a_{ij} \in [0,1]$

5.6.9 Local EM update rule

A mixture form of the sum-product rule (2.6) and the message passing EM update rule (5.63) or (5.73) can be devised by integrating out some of the hidden variables. Assume the set of hidden variables $\{X\}_N$ is partitioned into two sets $\{X\}_M$ and $\{Z\}_P$, $N = M + P$, where all variables in the set $\{Z\}_P$ are integrated out by the sum-product algorithm and the variables $\{X\}_M$ are "left over" to the message passing EM rule:

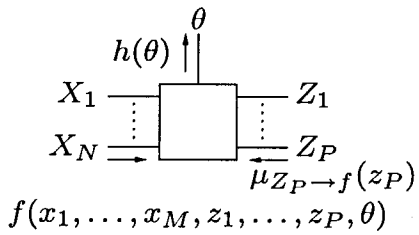


Figure 5.15: Factor graph node for the local EM update rule.

Local EM update rule:

$$h(\theta) = \mathsf{E}_p[\log g(X_1, \ldots, X_M, \theta)] \tag{5.87}$$

$$= \int_{\mathcal{D}_\mathcal{X}} \cdots \int p(x_1, \ldots, x_M | \hat{\theta}^{(k)}) \log g(x_1, \ldots, x_M, \theta) dx \tag{5.88}$$

where

$$g(x_1, \ldots, x_M, \theta) = \int_{\mathcal{D}_Z} \cdots \int f(x_1, \ldots, x_M, z_1, \ldots, z_P, \theta) \cdot \mu_{Z_1 \to f}(z_1) \cdots \mu_{Z_P \to f}(z_P) \cdot dz_1 \cdots dz_P \tag{5.89}$$

This alternative update rule is of special interest if the node is a deterministic relation, i.e. if the node function $f(.)$ is a Dirac delta function. In this case, $e^{h(\theta)}$ would also be a Dirac delta function, which obviously makes no sense. Instead, one can incorporate one message by computing (5.89) to eliminate the corresponding variable, such that $g(.)$ (and $h(\theta)$, consequently) is a continuous function.

This alternative update rule (5.88) could not (so far) be justified in the global view. Nevertheless, in the local view it is reasonable and

simulations show comparable performance to the message passing EM update rule. The following example show the application to the AR coefficient estimation problem.

Example 5.10. (Scalar multiplication node)

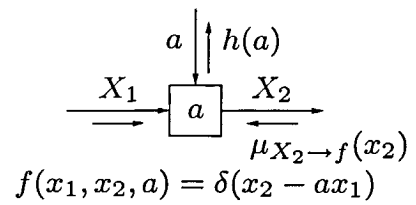


Figure 5.16: Scalar multiplication node to demonstrate the application of the local EM update rule.

The h-message after (5.63) for the node depicted in Fig. 5.16 is

$$h(a) = \int_{-\infty}^{\infty}\int p(x_1, x_2|\hat{a}^{(k)}) \log f(x_1, x_2, a)\, dx_1 dx_2. \tag{5.90}$$

Here we apply (5.88) and (5.89) and integrate out x_2. So with

$$\begin{aligned} g(x_1, a) &= \int_{-\infty}^{\infty} f(x_1, x_2, a)\mu_{X_2\to f}(x_2)\, dx_2 && (5.91)\\ &= \int_{-\infty}^{\infty} \delta(x_2 - ax_1)\mu_{X_2\to f}(x_2)\, dx_2 && (5.92)\\ &= \mu_{X_2\to f}(ax_1) && (5.93)\\ &= \mathcal{N}(ax_1 \mid m_2, v_2) && (5.94) \end{aligned}$$

the h-message becomes

$$\begin{aligned} h'(a) &= \int_{-\infty}^{\infty} p(x_1|\hat{a}^{(k)}) \log g(x_1, a)\, dx_1 && (5.95)\\ &= \int_{-\infty}^{\infty} p(x_1|\hat{a}^{(k)}) \log \mu_{X_2\to f}(ax_1)\, dx_1 && (5.96)\\ &= C - \frac{1}{2v_2}\left(a^2 \mathsf{E}[X_1^2|\hat{a}^{(k)}] - 2am_2\mathsf{E}[X_1|\hat{a}^{(k)}] + m_2^2\right). && (5.97) \end{aligned}$$

with

$$C = -\frac{1}{2}\log(2\pi v_2) \tag{5.98}$$

and in the exponential domain

$$e^{h'(a)} \propto \mathcal{N}_W\left(a \,\middle|\, \frac{m_2 \mathsf{E}[X_1]}{\mathsf{E}[X_1^2]}, \frac{\mathsf{E}[X_1^2]}{v_2}\right) \tag{5.99}$$

where the expectations are w.r.t. the distribution $p(x_1|\hat{a}^{(k)})$. Compare to (Table 5.5-7).

□

5.6.10 Message passing EM summary

The formulation of the EM algorithm as message passing on a factor graph has some interesting implications:

- A **local view** onto the EM algorithm has been given. It is not necessary to handle complicated equations for the whole model. The problem is divided into simple and small units. The global model is built by connecting these units into a graph. The EM algorithm is then computed by passing messages along the edges of this graph.

- A **generic update rule** for EM messages on a factor graph has been given (5.63). Given the node function and the incoming sum-product messages this rule leads to the message sent along the edges modelling the parameters.

- A **table of reusable building blocks** has been given. This Table 5.5 was derived by applying the update rule (5.63) for nodes often used in this thesis.

- The message passing EM **fits well into the factor graph framework** developed so far. Once a probabilistic problem is stated in the factor graph language, different algorithms can be derived by passing different messages along the edges of the graph. The EM messages are an alternative among others.

- There is much **flexibility in choosing a schedule** which opens up the opportunity to develop different forms of the EM algorithm, especially online estimation algorithms.

- It is also possible to **combine with other message types.** Either the E-step as well as the M-step can apply different techniques such as simulation based methods (particle filter, etc.) or gradient methods (steepest descent, Newton, etc.).

- **Non-trivial a priori models** for the parameters are possible. The M-step amounts to the max-product algorithm on the graph for the parameters, for Gaussian e^h-messages this equals the Kalman filter.

- A **local EM update rule** has been devised. Although not globally justified it leads to practical algorithms. It is also a possible solution to use deterministic nodes in the EM algorithm.

5.7 Summary

This section summarises the most important contributions made in Chapter 5. It proposes different techniques to move beyond Gaussian models. Every presented technique can be expressed as message passing on the corresponding factor graph. The proposed techniques are summarised in the following list:

- In **sum-product with tentative decision** one or more input messages are replaced by some hard estimates. This replacement is made only temporary for computing one outgoing sum-product message. The method of tentative decision is a generalisation of the method of iterative conditional modes (ICM).

- Sometimes non-Gaussian sum-product messages still belong to a family of functions which can be described by a bounded set of parameters, as is the case with **inverted gamma messages**, which appear in variance estimation. Unfortunately, the application of such messages is limited.

- When looking for a maximum a posteriori estimation, it may be possible to find sufficient statistics to described the **approximated mode** of the posterior density. We could show that such an approximation may also be computed by messages passing.

- The mode of a posterior density or of a likelihood function may also be found by **gradient methods**. We have shown that passing gradients as messages in the factor graph leads to gradient descend or hill climbing methods.

- Another way to approximate messages which cannot be expressed in closed form is to sample from it a couple of times. The resulting list of samples represents the message. Passing such messages in the factor graph leads to different forms of **particle methods**.

- A striking finding has been that the **expectation maximisation (EM) algorithm** can also be expressed as message passing on the factor graph. A messages passing EM update rule has been given which allows to derive the EM algorithm by local message updates. This new view onto EM has some interesting implications; in addition, many things about the message passing EM remain to be investigated.

All of the above mentioned methods lead to local message update rules which are tabulated in the corresponding sections. The algorithm designer can simply apply those building blocks to build complex stochastic models. The combination of different building blocks leads to completely new types of algorithms.

Seite Leer /
Blank leaf

Chapter 6

Complete Algorithms and Simulation Results

In this chapter complete algorithms are formulated in detail in the factor graph language. We treat four different examples of application: AR coefficient estimation in Section 6.1, variance estimation in Section 6.2 and the joint estimation of the coefficient and the variances of the autoregressive model in Section 6.3. We make use of all the techniques presented in Chapter 4 and Chapter 5.

6.1 Coefficient Estimation

In Section 4.3 we showed how message passing on the factor graph of a state-space model amounts to the classic Kalman filter recursions. In that case the AR coefficient vector $\mathbf{a}$, and therefore the state-transition matrix $\mathbf{A}$, was known. Here, the coefficients are unknown. The factor graph has to be extended to incorporate these (unknown) parameters by adding additional edges to the graph. In the following subsections we show different use cases of coefficient estimation which lead to different types of algorithms.

6.1.1 Coefficient estimation without measurement noise (LPC)

In this section the estimation of the coefficients of the autoregressive model without measurement noise (i.e. observation noise) is considered. For the unknown coefficients in the state transition matrix $\mathbf{A}$ additional edges are added to the graph (dashed lines in Fig. 6.1(a)). Because we assume the coefficients to be constant from one section to the next, these edges are connected via equality nodes across all sections. The messages along the edges for the state (① to ⑧) are computed according to the same scheme as in Section 4.3 but without the nodes representing the observation noise. Because the state is directly observable the message passing amounts to merely collecting the observations into the state vector. Consequently, no iterations are necessary—forward-only processing (cf. Section 3.2.1) suffices to obtain the exact posteriors.

① State estimate at time $n-1$:

$$\mu_1(\mathbf{x}_{n-1}) = \mathcal{N}(\mathbf{x}_{n-1} \mid \mathbf{y}_{n-1}, \mathbf{0})$$

where $\mathbf{y}_{n-1} = (y_{n-1}, \ldots, y_{n-M-1})^\mathsf{T}$ and M is the model order.

② State transition (Table 4.1-3):

$$\mu_2(.) = \mathcal{N}(\,.\mid \mathbf{A}_{n-1}\mathbf{y}_{n-1}, \mathbf{0})$$

Note that here the actual estimate $\hat{\mathbf{a}}_{n-1}$ of the coefficients are used to build the matrix $\mathbf{A}$. This is reflected by the time index $n-1$.

③ Control input and forward matrix multiplication (Table 4.1-3):

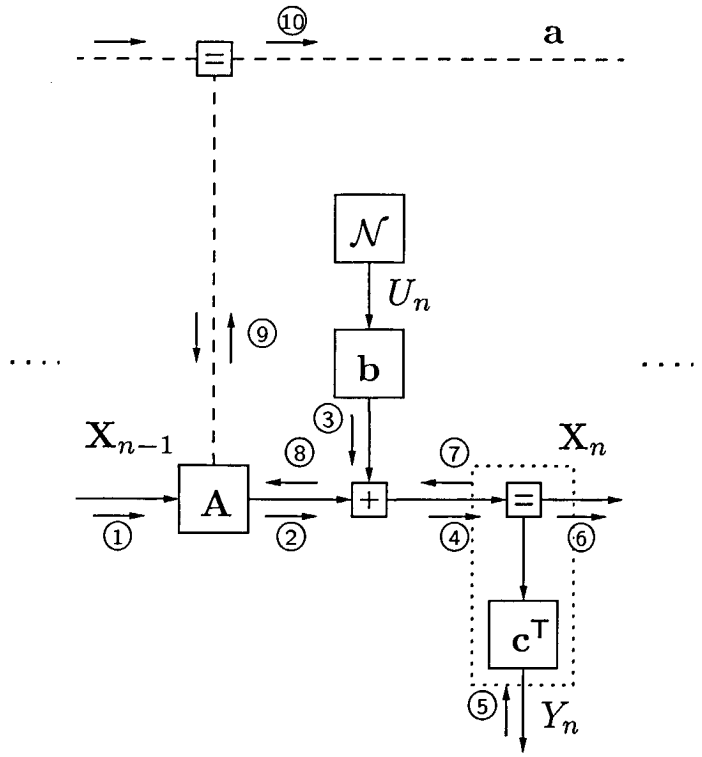
$$\mu_3(.) = \mathcal{N}(\,.\mid \mathbf{0}, \sigma_U^2\mathbf{b}\mathbf{b}^\mathsf{T})$$

④ Adding input and state (Table 4.1-2):

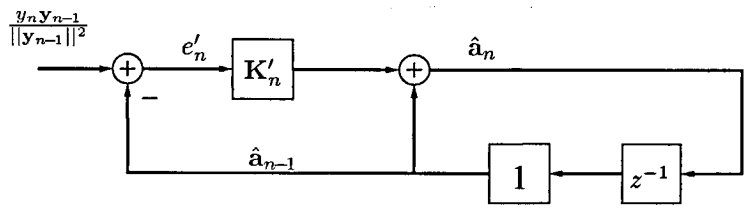
$$\mu_4(.) = \mathcal{N}(\,.\mid \mathbf{A}_{n-1}\mathbf{y}_{n-1}, \sigma_U^2\mathbf{b}\mathbf{b}^\mathsf{T})$$

⑤ Observation (Table 4.1-7):

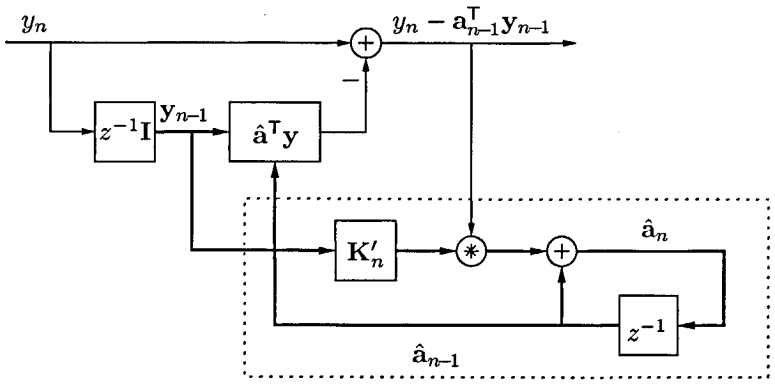
$$\mu_5(.) = \mathcal{N}(\,.\mid y_n, 0)$$



(a) Factor graph and messages.



(b) Recursive estimator according to (6.8).



(c) Recursive estimator according to (6.7).

Figure 6.1: Auto-regression (AR) coefficient estimation. The basis is the Kalman filter with additional edges for the unknown coefficients (a). Computing all messages in the forward direction leads to a recursion, which can be formulated in two different ways (b) and (c).

⑥ Update with information from new observation (Table 4.1-5):

$$\begin{aligned}\mu_6(.) &= \mathcal{N}(\mathbf{x}_n \mid \mathbf{y}_n, \mathbf{0}) \\ \mathbf{y}_n &= \mathbf{A}_{n-1}\mathbf{y}_{n-1} + \mathbf{c}\left(y_n - \mathbf{c}^{\mathsf{T}}\mathbf{A}_{n-1}\mathbf{y}_{n-1}\right) \\ &= \mathbf{S}\mathbf{y}_{n-1} + \mathbf{c}\, y_n \end{aligned} \tag{6.1}$$

The matrix $\mathbf{S}$ in (6.1) is the shift matrix, which has ones in its subdiagonal and is zero elsewhere. The operation performed by messages ① to ⑥ is solely collecting the observations into the state vector.

⑦

$$\mu_7(\mathbf{x}_n) = \mathcal{N}(\mathbf{x}_n \mid \mathbf{m}_7, \mathbf{V}_7)$$

$$\mathbf{m}_7 = \mathbf{c}y_n = \begin{pmatrix} y_n \\ \mathbf{0} \end{pmatrix} \quad \mathbf{V}_7 = \begin{bmatrix} 0 & \\ & \mathbf{I}_\infty \end{bmatrix}$$

There is no information about some entries in the state vector; this is reflected by the $\mathbf{I}_\infty$ in the covariance matrix, which has ∞'s in its diagonal and is zero elsewhere.

⑧ Adding control input (Table 4.1-2):

$$\mu_8(.) = \mathcal{N}(\,.\mid \mathbf{m}_8, \mathbf{V}_8)$$

$$\mathbf{m}_8 = \begin{pmatrix} y_n \\ \mathbf{0} \end{pmatrix} \quad \mathbf{V}_8 = \begin{bmatrix} \sigma_U^2 & \\ & \mathbf{I}_\infty \end{bmatrix}$$

⑨ Since the covariance matrix of message ① is zero, and only the first entry of message ⑧ carries information, the following sum-product update formula applies:

$$\begin{aligned}\mu_9(\mathbf{a}) = &\iint_{-\infty}^{\infty} \delta(\mathbf{c}^{\mathsf{T}}\mathbf{x}_n - \mathbf{a}^{\mathsf{T}}\mathbf{x}_{n-1}) \cdot \\ &\delta(\mathbf{x}_{n-1} - \mathbf{y}_{n-1})\, \mathcal{N}\left(\mathbf{c}^{\mathsf{T}}\mathbf{x}_n \mid y_n, \sigma_U^2\right)\, d\mathbf{x}_{n-1} d\mathbf{x}_n \\ &\propto \exp\left(-\frac{y_n - \mathbf{a}^{\mathsf{T}}\mathbf{y}_{n-1}}{2\sigma_U^2}\right) \\ &\propto \exp \|\mathbf{a} - \mathbf{m}_9\|^2_{\mathbf{W}_9}\end{aligned} \tag{6.2}$$

with mean and weight matrix

$$\mathbf{m}_9 = \frac{y_n}{\|\mathbf{y}_{n-1}\|}\mathbf{y}_{n-1} \tag{6.3}$$

$$\mathbf{W}_9 = \frac{\mathbf{y}_{n-1}\mathbf{y}_{n-1}^{\mathsf{T}}}{\sigma_U^2} \tag{6.4}$$

Hence, message ⑨ is a (multivariate) Gaussian:

$$\mu_9(\mathbf{a}) = \mathcal{N}_W(\mathbf{a} \mid \mathbf{m}_9, \mathbf{W}_9)$$

Note that the inverse of $\mathbf{W}_9$ does not exist because $\mathbf{W}_9$ has rank one only. The derivation of this update rule in general can be found in Appendix B.3.

⑩ The update at the equality node is (Table 4.1-1):

$$\mu_{10}(\mathbf{a}_n) = \mathcal{N}_W(\mathbf{a}_n \mid \hat{\mathbf{a}}_n, \mathbf{W}_{\mathbf{a}_n})$$

$$\mathbf{W}_{a_n} = \mathbf{W}_{a_{n-1}} + \frac{\mathbf{y}_{n-1}\mathbf{y}_{n-1}^{\mathsf{T}}}{\sigma_U^2} \tag{6.5}$$

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \frac{1}{\sigma_U^2}\mathbf{W}_{a_n}^{-1}\mathbf{y}_{n-1}\left(y_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}}\mathbf{y}_{n-1}\right) \tag{6.6}$$

This is generally implemented in the following form

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \mathbf{K}_n\mathbf{y}_{n-1}e_n \tag{6.7}$$

$$\mathbf{K}_n = \frac{\mathbf{W}_{a_n}^{-1}}{\sigma_U^2}$$

$$e_n = y_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}}\mathbf{y}_{n-1}$$

as depicted in Fig. 6.1(c).

Another form of this update looks like

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \mathbf{K}'_n e'_n \tag{6.8}$$

$$e'_n = \frac{y_n\mathbf{y}_{n-1}}{\|\mathbf{y}_{n-1}\|^2} - \hat{\mathbf{a}}_{n-1} \tag{6.9}$$

$$\mathbf{K}'_n = \mathbf{W}_{a_n}^{-1}\frac{\mathbf{y}_{n-1}\mathbf{y}_{n-1}^{\mathsf{T}}}{\sigma_U^2} \tag{6.10}$$

which reveals the similarity to the Kalman filter.

After the N-th section the estimate of the coefficient vector is:

$$\hat{\mathbf{a}}_N = \left(\sum_{n=1}^{N}\mathbf{W}_{\mathbf{a}_n}\right)^{-1}\left(\sum_{n=1}^{N}\mathbf{W}_{\mathbf{a}_n}\hat{\mathbf{a}}_n\right) \tag{6.11}$$

$$= \left(\sum_{n=1}^{N}\mathbf{y}_{n-1}\mathbf{y}_{n-1}^{\mathsf{T}}\right)^{-1}\left(\sum_{n=1}^{N}\mathbf{y}_{n-1}y_n\right) \tag{6.12}$$

$$= \mathbf{R}_N^{-1}\mathbf{r}_N \tag{6.13}$$

where $\mathbf{R}_N$ is the (deterministic) correlation matrix and $\mathbf{r}_N$ the (deterministic) correlation vector. This is the classic least-squares solution for the LPC (linear predictive coding) problem [54, 90].

The dashed edges for the coefficient estimation in Fig. 6.1(a) are in fact a degenerate version of the Kalman filter structure (Fig. 4.1(a)) without input and dynamics. Additionally, because the state covariance matrices are zero (messages ①, ②, ④ and ⑥) and the covariance matrices of messages ⑦ and ⑧ have a special structure, message passing simply amounts to collecting the observations into the state vector.

So the graph could be redrawn as in Fig. 6.2 (cf. Section 4.4). We learn two things from that representation. First, the graph has no loops, therefore no iterations are needed to compute every posterior density exactly; forward-only message passing suffices. Second, the LPC recursion is a degenerate form of Kalman filtering, as has been shown previously in [55, 81, 116].

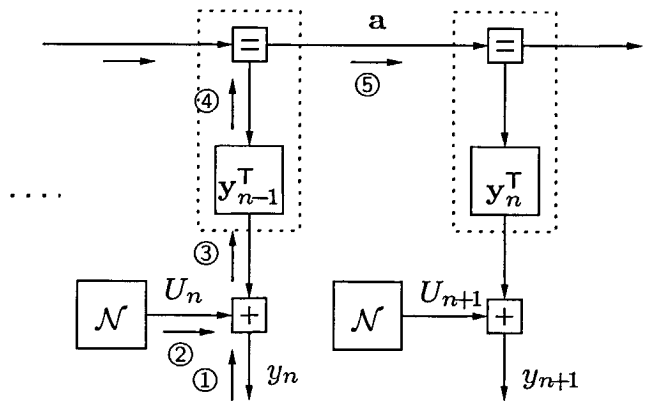


Figure 6.2: The coefficient estimation represented as Kalman filter. Past observations are collected into the vector $\mathbf{y}_{n-1}$. The complete derivation of this representation is given in Section 4.4.

Simulation results for the LPC algorithm are shown in Fig. 6.3. The plot shows the averaged mean squared estimation error versus the block length. As is well known, the error decreases exponentially with increasing block length.

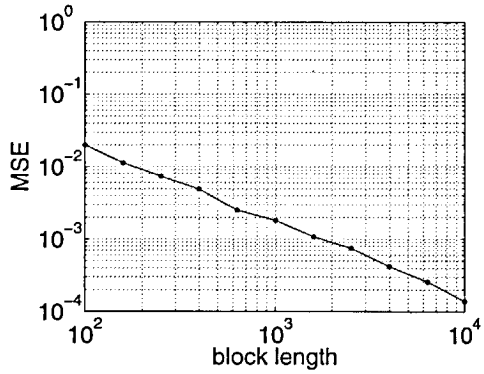


Figure 6.3: Estimation error vs. block length for the LPC algorithm.

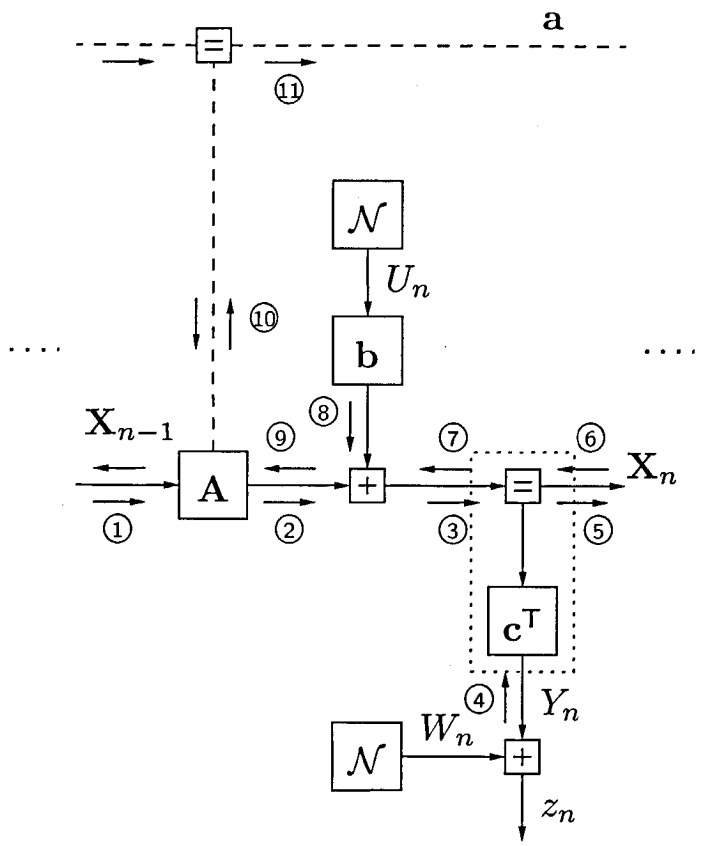
6.1.2 Coefficient estimation with measurement noise

In Section 6.1.1 an algorithm for coefficient estimation of clean autoregressive (AR) signals has been derived. In this section we assume that the AR signal is obscured by additive noise. Using the above mentioned coefficient estimation method when the observations are obscured by noise leads to wrong estimates of the AR coefficient vector. Classic approaches [14, 24, 66, 145] try to compensate the noise in different ways. Here we estimate the clean state from which the coefficient vector is derived together with the coefficients.

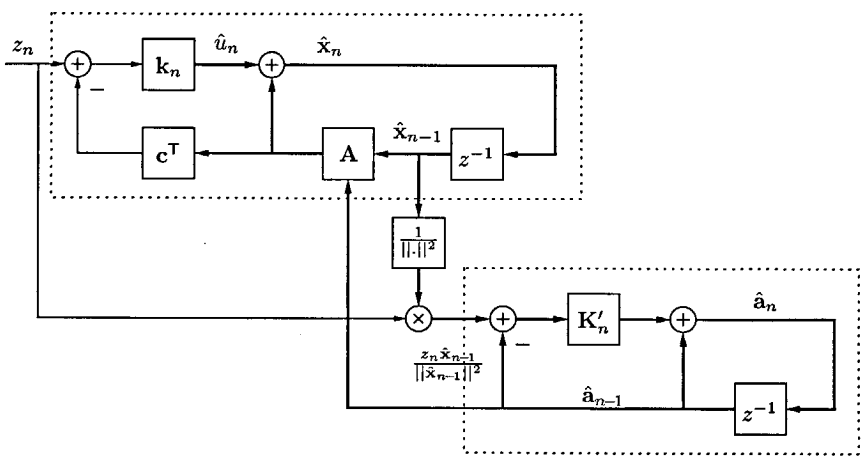
The factor graph for an autoregressive model with observation noise is depicted in Fig. 6.4(a). Because the state is not directly observable, it has to be estimated as well. This means, the graph does not degenerate (as in Fig. 6.2), hence the cycles do not collapse. Because the graph has cycles now, the right choice of the message update schedule comes into play. Sticking to the forward-only scheme gives suboptimal results. Through iterating the results are improved. The algorithm designer has the freedom to trade off accuracy versus computational complexity by choosing different message update schedules.

After the standard Kalman filter update (messages ① to ⑤ in Fig. 6.4(a)) messages ⑥ to ⑪ are computed as follows:

⑥ Backward message from the last iteration. In the first iteration it is a neutral Gaussian $\mu_6(.) = \mathcal{N}(. \mid 0, \mathbf{I}_\infty)$.



(a) Factor graph and messages.



(b) Recursive estimator.

Figure 6.4: Auto-regression (AR) coefficient estimation with noisy observations. This graph (a) has cycles, so finding the right schedule is not straight-forward. A forward-only scheme leads to an recursive estimator (b) comprising two coupled Kalman filters.

⑦

$$\mu_7(\mathbf{x}_n) = \mathcal{N}\left(\mathbf{x}_n \,\middle|\, \begin{pmatrix} z_n \\ \mathbf{0} \end{pmatrix}, \begin{bmatrix} \sigma_W^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_\infty \end{bmatrix}\right)$$

⑧ Input and forward matrix multiplication ((Table 4.1-7) and (Table 4.1-3)):

$$\mu_8(.) = \mathcal{N}\left(. \,\middle|\, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{bmatrix} \sigma_U^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\right)$$

⑨ Adding input (Table 4.1-2):

$$\mu_9(.) = \mathcal{N}\left(. \,\middle|\, \begin{pmatrix} z_n \\ 0 \end{pmatrix}, \begin{bmatrix} \sigma_W^2 + \sigma_U^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_\infty \end{bmatrix}\right)$$

⑩ Because of the special structure of the covariance matrix of μ_9 the node function can be set to $\delta(\mathbf{c}^\mathsf{T}\mathbf{x}_n - \mathbf{a}^\mathsf{T}\mathbf{x}_{n-1})$, i.e. the first element of $\mathbf{x}_n$ must equal the vector product $\mathbf{a}^\mathsf{T}\mathbf{x}_{n-1}$ for valid configurations[1]. The node update is therefore:

$$\begin{aligned}\mu_{10}(\mathbf{a}) = \iint_{-\infty}^{\infty} & \delta(\mathbf{c}^\mathsf{T}\mathbf{x}_n - \mathbf{a}^\mathsf{T}\mathbf{x}_{n-1})\delta(\mathbf{x}_{n-1} - \hat{\mathbf{x}}_{n-1}) \cdot \\ & \mathcal{N}\left(\mathbf{c}^\mathsf{T}\mathbf{x}_n \mid z_n, \sigma_W^2 + \sigma_U^2\right) \, d\mathbf{x}_{n-1} d\mathbf{x}_n \\ \propto & \exp\left(-\frac{z_n - \mathbf{a}^\mathsf{T}\hat{\mathbf{x}}_{n-1}}{2(\sigma_W^2 + \sigma_U^2)}\right) \qquad (6.14) \\ \propto & \exp \|\mathbf{a} - \mathbf{m}_{10}\|^2_{\mathbf{W}_{10}} \qquad (6.15)\end{aligned}$$

with mean and weight matrix

$$\mathbf{m}_{10} = \frac{z_n}{\|\hat{\mathbf{x}}_{n-1}\|}\hat{\mathbf{x}}_{n-1} \qquad (6.16)$$

$$\mathbf{W}_{10} = \frac{\hat{\mathbf{x}}_{n-1}\hat{\mathbf{x}}_{n-1}^\mathsf{T}}{\sigma_U^2 + \sigma_W^2}. \qquad (6.17)$$

The derivation of this update rule in general can be found in Appendix B.3.

[1]Note, that this is only valid for the first iteration. For every further iteration, this is an approximation, where the information from the rest of the elements of $\mathbf{x}_{n-1}$ is ignored

⑪ The update at the equality node is:

$$\mu_{11}(\mathbf{a}_n) = \mathcal{N}_W(\mathbf{a}_n \mid \hat{\mathbf{a}}_n, \mathbf{W}_{\mathbf{a}_n})$$

$$\mathbf{W}_{a_n} = \mathbf{W}_{a_{n-1}} + \frac{\hat{\mathbf{x}}_{n-1}\hat{\mathbf{x}}_{n-1}^{\mathsf{T}}}{\sigma_U^2 + \sigma_W^2} \tag{6.18}$$

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \frac{\mathbf{W}_{a_n}^{-1}\hat{\mathbf{x}}_{n-1}}{\sigma_U^2 + \sigma_W^2}\left(z_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}}\hat{\mathbf{x}}_{n-1}\right) \tag{6.19}$$

This is generally implemented in the following form:

$$\begin{aligned} \hat{\mathbf{a}}_n &= \hat{\mathbf{a}}_{n-1} + \mathbf{K}_n\hat{\mathbf{x}}_{n-1}e_n \\ \mathbf{K}_n &= \frac{\mathbf{W}_{a_n}^{-1}}{\sigma_U^2 + \sigma_W^2} \\ e_n &= z_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}}\hat{\mathbf{x}}_{n-1}. \end{aligned} \tag{6.20}$$

Another form of this update looks like

$$\begin{aligned} \hat{\mathbf{a}}_n &= \hat{\mathbf{a}}_{n-1} + \mathbf{W}_{a_n}^{-1}\frac{\hat{\mathbf{x}}_{n-1}\hat{\mathbf{x}}_{n-1}^{\mathsf{T}}}{\sigma_U^2 + \sigma_W^2}\left(\frac{z_n\hat{\mathbf{x}}_{n-1}}{\|\hat{\mathbf{x}}_{n-1}\|^2} - \hat{\mathbf{a}}_{n-1}\right) \\ &= \hat{\mathbf{a}}_{n-1} + \mathbf{K}'_n e'_n \end{aligned} \tag{6.21}$$

with

$$e'_n = \frac{z_n\hat{\mathbf{x}}_{n-1}}{\|\hat{\mathbf{x}}_{n-1}\|^2} - \hat{\mathbf{a}}_{n-1} \tag{6.22}$$

$$\mathbf{K}'_n = \mathbf{W}_{a_n}^{-1}\frac{\hat{\mathbf{x}}_{n-1}\hat{\mathbf{x}}_{n-1}^{\mathsf{T}}}{\sigma_U^2 + \sigma_W^2}. \tag{6.23}$$

The resulting recursive estimator is depicted in Fig. 6.4(b). It comprises two coupled Kalman filters, one for the state estimation and the other for the coefficient estimation. This is again only valid for a forward-only schedule.

When iterating, messages ⑥ to ⑨ contain information from future observations. Thus, their covariance matrix has no more such a special structure. Despite that, message ⑩ is computed the same way as before, which is now an approximation. This approximation is reasonable because in the steady state the covariances of the state are usually small (cf. Section 5.1).

Simulation results for the coefficient estimation algorithm are shown in Fig. 6.5. In the case of noisy observations the standard LPC algorithm

fails. The dashed lines in Fig. 6.5 in the plot to the left are the estimation errors for three different noise levels for the LPC algorithm. The solid lines represent the estimation errors of the proposed coefficient estimation algorithm.

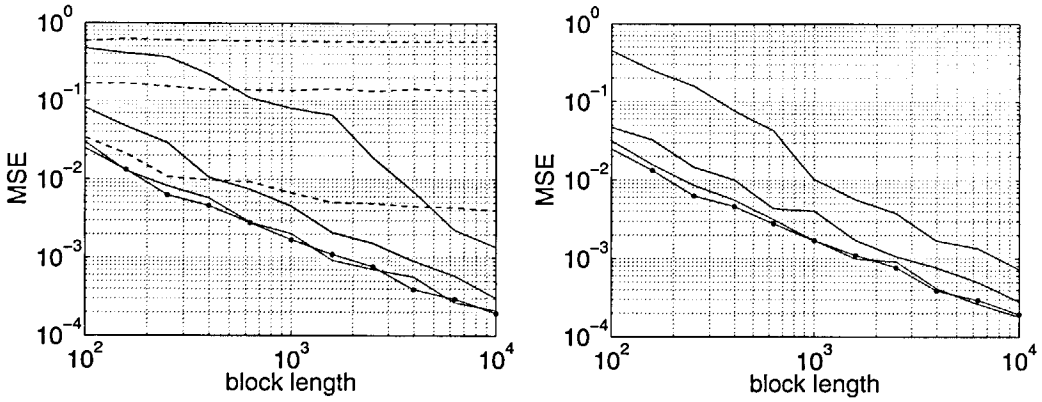


Figure 6.5: Coefficient estimation with noisy observations. Noise levels $\sigma_W^2 = 0.1/0.01/0.001$ and innovation variance $\sigma_U^2 = 0.1$. Left: Forward-only processing (dashed lines: standard LPC algorithm); Right: 3 iterations. The line with markers represents the noise-free case ($\sigma_W^2 = 0$).

For the left-hand plot only the forward messages (i.e. no iterations) were computed, whereas for the right-hand plot three iterations were computed. Further iterations yield only insignificant further improvements. The noise-free case acts as a benchmark and is shown in Fig. 6.5 as line with markers.

In conclusion, the new algorithm for coefficient estimation is able to compensate the noise, even in forward-only processing.

6.1.3 RLS adaptive filter

In this section the estimation of the coefficients of an FIR filter is considered. In Section 4.5 the RLS adaptive filter has been derived from scratch. Here, in contrast, we start from our state-space model view and add edges for the unknown coefficient vector to our graph. As will become obvious, this approach leads to the same result, which emphasises the versatility of the factor graph respresentation of the state-space model.

Adaptive filters usually are of the FIR-type (finite impulse response). The FIR filter corresponds to the moving-average process in statistics. In state-space form it is defined as [55, 81, 116]:

$$\mathbf{x}_n = \mathbf{S}\mathbf{x}_{n-1} + \mathbf{b}u_n \tag{6.24}$$

$$y_n = \mathbf{a}^{\mathsf{T}}\mathbf{x}_n \tag{6.25}$$

$$z_n = y_n + w_n \tag{6.26}$$

with

$$\mathbf{S} = \begin{bmatrix} \mathbf{0} & 0 \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \tag{6.27}$$

$$\mathbf{a} = [a_1, \ldots, a_M]^{\mathsf{T}} \tag{6.28}$$

$$\mathbf{b} = [1\ 0 \ldots 0]^{\mathsf{T}}. \tag{6.29}$$

Where $\mathbf{S}$ is just a shifting matrix and $\mathbf{a}$ are the unknown coefficients. The corresponding factor graph is depicted in Fig. 6.6.

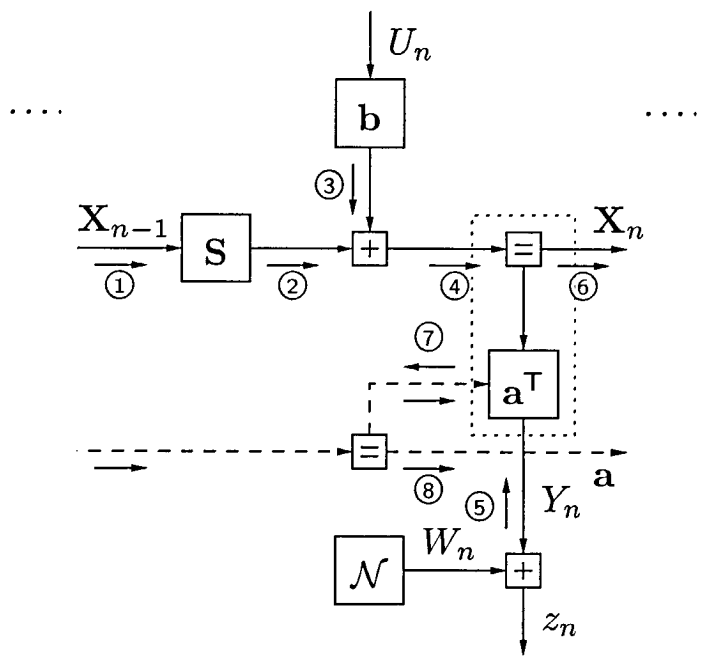


Figure 6.6: Factor graph for an adaptive FIR filter.

The messages in Fig. 6.6 are computed as follows:

① State at time $n-1$:

$$\mu_1(\mathbf{x}_{n-1}) = \mathcal{N}(\mathbf{x}_{n-1} \mid \mathbf{u}_{n-1}, \mathbf{0})$$

② State transition (Table 4.1-3):

$$\mu_2(.) = \mathcal{N}(\,.\mid \mathbf{S}\mathbf{u}_{n-1}, \mathbf{0})$$

③ (Table 4.1-3):

$$\mu_3(.) = \mathcal{N}(\,.\mid \mathbf{b}u_n, \mathbf{0})$$

④ Adding input and state (Table 4.1-2):

$$\mu_4(.) = \mathcal{N}(\,.\mid \mathbf{S}\mathbf{u}_{n-1} + \mathbf{b}u_n, \mathbf{0})$$

⑤ Observation with noise (Table 4.1-7):

$$\mu_5(.) = \mathcal{N}(\,.\mid z_n, \sigma_W^2)$$

⑥ The update with information from new observation (Table 4.1-5) is degenerate:

$$\begin{aligned} \mu_6(.) &= \mathcal{N}(\mathbf{x}_n \mid \mathbf{u}_n, \mathbf{0}) \\ \mathbf{u}_n &= \mathbf{S}\mathbf{u}_{n-1} + \mathbf{b}u_n \end{aligned} \tag{6.30}$$

This operation is just shifting the vector $\mathbf{u}_{n-1}$ one element down and setting u_n as its first element.

⑦ Here, the covariance matrix of message ④ is zero. That means, we are completely certain about $\mathbf{u}_n$ (the vector of input values). Consequently, the multiplication node reduces to the standard backward matrix multiplication (Table 4.1-4) where the value of the fixed matrix is $\mathbf{u}_n$.

$$\begin{aligned} \mu_7(\mathbf{a}) &= \mathcal{N}_W(\mathbf{a} \mid \mathbf{m}_7, \mathbf{W}_7) \\ \mathbf{m}_7 &= \frac{z_n}{\|\mathbf{u}_n\|}\mathbf{u}_n \qquad \mathbf{W}_7 = \frac{\mathbf{u}_n\mathbf{u}_n^\mathsf{T}}{\sigma_W^2} \end{aligned} \tag{6.31}$$

⑧ The update at the equality node is (Table 4.1-1):

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \frac{1}{\sigma_W^2}\mathbf{W}_{a_n}^{-1}\mathbf{u}_n\left(z_n - \hat{\mathbf{a}}_{n-1}^\mathsf{T}\mathbf{u}_n\right) \tag{6.32}$$

$$\mathbf{W}_{a_n} = \mathbf{W}_{a_{n-1}} + \frac{\mathbf{u}_n\mathbf{u}_n^\mathsf{T}}{\sigma_W^2} \tag{6.33}$$

Because of the equivalence of the coefficient estimation recursion and the Kalman filter (Section 4.3.1), the coefficient update formulas (6.32)-(6.33) can be represented in different ways, e.g. by using the covariance matrix or the weight matrix to represent Gaussian messages.

The coefficient update formulas using the covariance matrix $\mathbf{V}_{\mathbf{a}_n}$ instead of its inverse $\mathbf{W}_{\mathbf{a}_n}$ look like

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \mathbf{k}_n e_n \tag{6.34}$$

$$\mathbf{V}_{\mathbf{a}_n} = \left(\mathbf{I} - \mathbf{k}_n \mathbf{u}_n^{\mathsf{T}}\right) \mathbf{V}_{\mathbf{a}_{n-1}} \tag{6.35}$$

$$e_n = z_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}} \mathbf{u}_n \tag{6.36}$$

$$\mathbf{k}_n = \frac{\mathbf{V}_{\mathbf{a}_{n-1}} \mathbf{u}_n}{\sigma_W^2 + \mathbf{u}_n^{\mathsf{T}} \mathbf{V}_{\mathbf{a}_{n-1}} \mathbf{u}_n} \tag{6.37}$$

Multiplying the covariance matrix by a forgetting factor λ leads to the weighted RLS:

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \mathbf{k}_n e_n \tag{6.38}$$

$$\mathbf{V}_{\mathbf{a}_n} = \lambda^{-1} \left(\mathbf{I} - \mathbf{k}_n \mathbf{u}_n^{\mathsf{T}}\right) \mathbf{V}_{\mathbf{a}_{n-1}} \tag{6.39}$$

$$e_n = z_n - \hat{\mathbf{a}}_{n-1}^{\mathsf{T}} \mathbf{u}_n \tag{6.40}$$

$$\mathbf{k}_n = \frac{\lambda^{-1} \mathbf{V}_{\mathbf{a}_{n-1}} \mathbf{u}_n}{\sigma_W^2 + \lambda^{-1} \mathbf{u}_n^{\mathsf{T}} \mathbf{V}_{\mathbf{a}_{n-1}} \mathbf{u}_n} \tag{6.41}$$

This is the standard form of the RLS adaptive filter algorithm [54].

Similar to the alternative factor graph for LPC in Section 4.4 we can redraw the graph given in Fig. 6.6 as shown in Fig. 6.7 (cf. Section 4.5).

6.1.4 LMS adaptive filter

Updating the full covariance matrix can be costly in high dimensions, so one can apply gradient descent to find the (maximum a posteriori) estimate. Gradient methods fit naturally into the factor graph framework, because the log-derivative of a factored function equals the sum of the log-derivatives of the individual factors. These individual derivatives and sums can be computed by message passing along the corresponding edges in the graph. An introduction to this technique is given in Section 5.4, for further details refer to [23, 85].

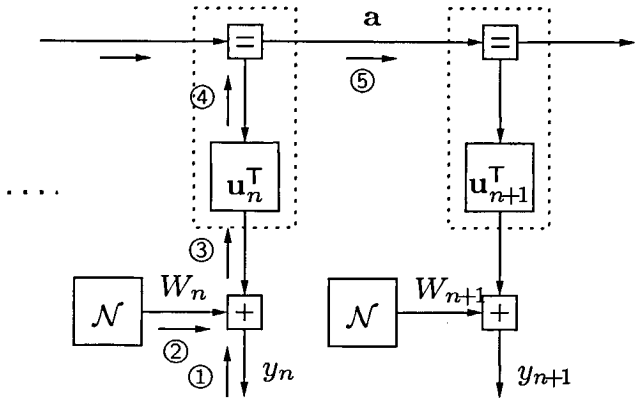


Figure 6.7: The RLS algorithm represented as Kalman filter. Past inputs are collected into the vector $\mathbf{u}_n$.

The factor graph (Fig. 6.6) and messages ① to ⑥ are the same as in Section 6.1.3.

⑦ Instead of computing the message in closed form, compute its gradient at the given estimate $\hat{\mathbf{a}}_{n-1}$:

$$\log \mu_7(\mathbf{a}) = -\frac{1}{2}(\mathbf{a}-\hat{\mathbf{a}})^\mathsf{T}\mathbf{W}_\mathbf{a}(\mathbf{a}-\hat{\mathbf{a}}) + \zeta$$

$$\nabla \log \mu_7(\mathbf{a}) = -\mathbf{W}_\mathbf{a}(\mathbf{a}-\hat{\mathbf{a}}) \tag{6.42}$$

$$= -\frac{\mathbf{u}_n\mathbf{u}_n^\mathsf{T}}{\sigma_W^2}\left(\mathbf{a} - \frac{y_n\mathbf{u}_n}{\|\mathbf{u}_n\|}\right) \tag{6.43}$$

$$\nabla \log \mu_7(\mathbf{a})\big|_{\mathbf{a}=\hat{\mathbf{a}}_{n-1}} = -\frac{\mathbf{u}_n}{\sigma_W^2}\left(y_n - \hat{\mathbf{a}}_{n-1}^\mathsf{T}\mathbf{u}_n\right) \tag{6.44}$$

where ζ is an appropriate scaling factor.

⑧ Update of the coefficients with step-size λ:

$$\hat{\mathbf{a}}_n = \hat{\mathbf{a}}_{n-1} + \lambda\, \nabla \log \mu_7(\mathbf{a})\big|_{\mathbf{a}=\hat{\mathbf{a}}_{n-1}} \tag{6.45}$$

$$= \hat{\mathbf{a}}_{n-1} + \lambda'\, \mathbf{u}_n e_n \tag{6.46}$$

$$e_n = y_n - \hat{\mathbf{a}}_{n-1}^\mathsf{T}\mathbf{u}_n \tag{6.47}$$

This is the standard form of the least mean squares (LMS) algorithm [54]. Note that the topology of the graphs for the RLS (Section 6.1.3) and the LMS adaptive filter algorithm is the same. They differ only in the way how the messages are computed. This unified view of different signal processing algorithms make factor graphs a very powerful tool.

6.2 Variance Estimation

So far, the variances of the input and observation noise of the AR model were assumed to be known. In this section some approaches to estimate the variance of the input noise, in case of known coefficients, are shown. In the scope of variance estimation we point out the problems arising when messages are non-Gaussian and use some of the remedies proposed in Chapter 5.

Classic variance estimation techniques for the Kalman filter can be found in [8, 97, 98].

6.2.1 Variance estimation without measurement noise

Here we want to estimate the variance σ_U^2 of the input noise sequence of the AR model when the variance of the observation noise is zero, $\sigma_W^2 = 0$. The transition matrix $\mathbf{A}$ is assumed to be fixed and known. A single section of the corresponding FFG is shown in Fig. 6.8(a). The $\mathcal{N}$-node with two edges represents a Gaussian density with unknown variance: the corresponding factor is $\mathcal{N}(u_n \mid 0, \sigma_U^2)$ where σ_U^2 is itself a random variable in that case. The messages along the dotted edges are non-Gaussian; the update rules for these messages are derived in Section 5.2.

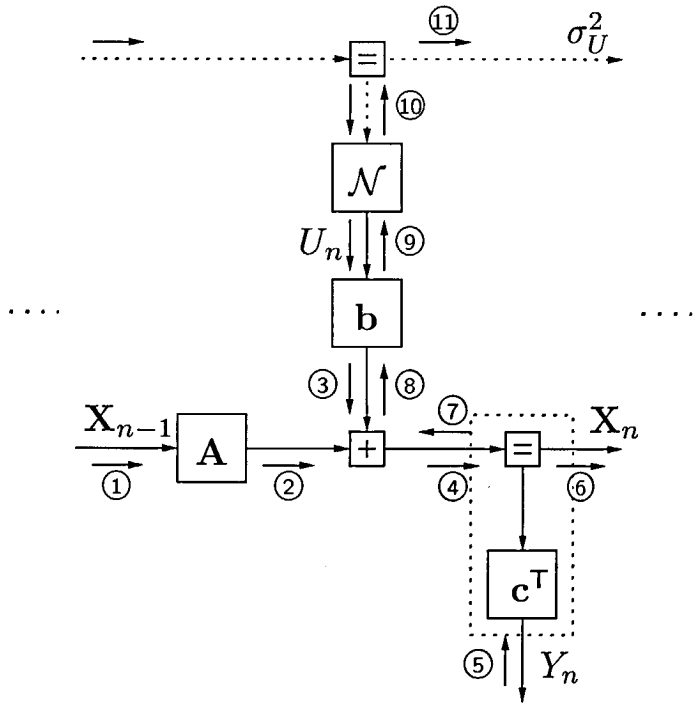
First, messages ① to ⑥ in Fig. 6.8(a) are computed as in Section 4.3 (Kalman filter). The innovation variance σ_U^2 is set to the latest estimate $\hat{\sigma}^2_{U_{n-1}}$. Second, given the observation y_n, the messages for the variance estimation are:

⑦ (Table 4.1-1)

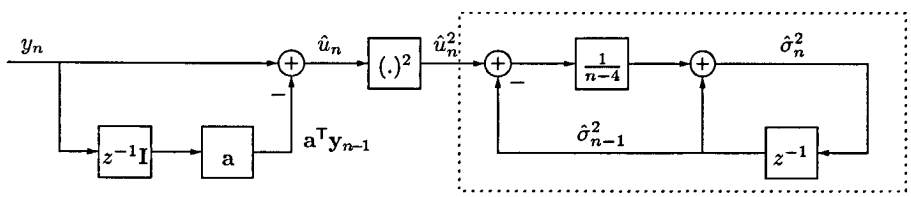
$$\mu_7(\mathbf{x}_n) = \mathcal{N}\left(\mathbf{x}_n \,\middle|\, \mathbf{c}y_n, \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_\infty \end{bmatrix}\right)$$

⑧ (Table 4.1-2)

$$\mu_8(.) = \mathcal{N}(\,.\mid \mathbf{c}y_n - \mathbf{A}\mathbf{y}_{n-1}, \mathbf{V}_8)$$



(a) Factor graph and messages.



(b) Recursive estimator.

Figure 6.8: Variance estimation of the input noise sequence of the AR model. Because the graph (a) is again degenerate a forward-only message update schedule suffices to gain a recursive estimator (b).

where $\mathbf{y}_{n-1} = (y_{n-1}, \ldots, y_{n-P})^\mathsf{T}$ and

$$\mathbf{V}_8 = \mathbf{V}_2 + \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_\infty \end{bmatrix}. \tag{6.48}$$

⑨ Estimation of the input u_n (Table 4.1-4):

$$\mu_9(u_n) = \mathcal{N}(u_n \mid \hat{u}_n, 0)$$

$$\hat{u}_n = \frac{\mathbf{b}^\mathsf{T}}{|\mathbf{b}|^2}(\mathbf{c}y_n - \mathbf{A}\mathbf{y}_{n-1}) \tag{6.49}$$

$$= y_n - \mathbf{a}^\mathsf{T}\mathbf{y}_{n-1} \tag{6.50}$$

⑩ Applying the node update equation (2.6):

$$\mu_{10}(\sigma^2) \propto \int_{-\infty}^{\infty} \mathcal{N}(u_n \mid 0, \sigma^2)\ \delta(u_n - \hat{u}_n) du_n$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\hat{u}_n^2}{2\sigma^2}\right) \tag{6.51}$$

$$\propto \mathrm{Ig}\left(\sigma^2 \,\middle|\, -\frac{1}{2}, \frac{\hat{u}_n^2}{2}\right) \tag{6.52}$$

where Ig stands for an inverted gamma distribution with parameters α and β (cf. Appendix A.1.2).

⑪ Because the inverted gamma distribution is closed under multiplication, computing this message is a mere update of the α's and β's (Table 5.2):

$$\mu_{11}(\sigma_n^2) \propto \mathrm{Ig}\left(\sigma_n^2 \,\middle|\, \alpha_{n-1} + \frac{1}{2}, \beta_{n-1} + \frac{\hat{u}_n^2}{2}\right) \tag{6.53}$$

Repeated application of (6.53) gives after N sections:

$$\mu(\sigma_N^2) \propto \frac{1}{(2\pi\sigma_N^2)^{N/2}} \exp\left(-\frac{\sum_{n=1}^N \hat{u}_n^2}{2\sigma_N^2}\right)$$

$$\propto \mathrm{Ig}\left(\sigma_N^2 \,\middle|\, \frac{N}{2} - 1, \frac{1}{2}\sum_{n=1}^N \hat{u}_n^2\right) \tag{6.54}$$

The mean of the message for the variance, which corresponds to the MMSE estimate, at time N is

$$\hat{\sigma}_N^2 = E[\sigma_N^2] = \frac{1}{N-4}\sum_{n=1}^{N}\hat{u}_n^2 = \frac{1}{N-4}\sum_{n=1}^{N}\left(y_n - \mathbf{a}^{\mathsf{T}}\mathbf{y}_{n-1}\right)^2 \tag{6.55}$$

The updated estimate (6.55) can be computed iteratively:

$$\begin{aligned}\hat{\sigma}_n^2 &= \frac{1}{n-4}\left(\hat{u}_n^2 + \sum_{i=1}^{n-1}\hat{u}_i^2\right)\\ &= \hat{\sigma}_{n-1}^2 + \frac{1}{n-4}\left(\hat{u}_n^2 - \hat{\sigma}_{n-1}^2\right)\end{aligned} \tag{6.56}$$

which is a scalar Kalman filter for the variance having $\hat{u}_n^2$ as input and whose block diagram is given in Fig. 6.8(b). The factor $1/(n-4)$ results from the (implicitly) used prior $\mathrm{Ig}\left(\sigma^2 \mid -1, 0\right)$, which is a non-informative prior over 0^+ to ∞.

Instead of taking the mean one could compute the mode of message ⑪ which corresponds to the maximum a posteriori estimate.

$$\hat{\sigma}_N^2 = \underset{\sigma_N^2}{\operatorname{argmax}}\,\mu(\sigma_N^2) = \frac{1}{N}\sum_{n=1}^{N}\hat{u}_n^2 = \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \mathbf{a}^{\mathsf{T}}\mathbf{y}_{n-1}\right)^2 \tag{6.57}$$

which differs from the MMSE estimate only in the factor $1/N$ (cf. Appendix D).

The algorithm of Fig. 6.8(a) was used as a benchmark for variance estimation in general. Since the graph has no cycles and every message is exact, the results are exact minimum mean squared error (MMSE) estimates. Fig. 6.9 shows the averaged mean squared estimation errors as a function of the block length.

These estimation errors provide a lower bound on the estimation errors for the joint coefficient/variance estimation cases.

6.2.2 Variance estimation with measurement noise

Here we consider the case where we want to estimate the variance of the input noise sequence of the AR model when observation noise is present.

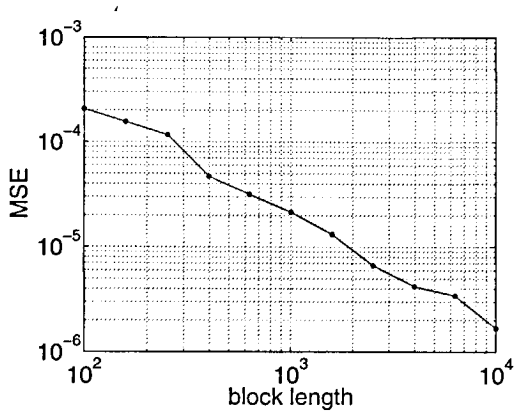


Figure 6.9: Estimation error vs. block length for the variance estimation algorithm for a variance of $\sigma_U^2 = 0.1$.

In Section 6.2.1 the state was directly observable. Here, the observations are obscured by additive noise such that the state has to be estimated as well. The corresponding factor graph is depicted in Fig. 6.10.

Messages ① to ⑨ are the same as in Section 4.3 (Kalman filter) except that for the innovation variance σ_U^2 the actual estimate $\hat{\sigma}_{U_{n-1}}^2$ is used. The messages for the variance re-estimation are:

⑩ (Table 4.1-1)

$$\mu_{10}(.) = \mathcal{N}(\,.\mid \mathbf{c}y_n, \mathbf{V}_{10})$$
$$\mathbf{V}_{10} = \begin{bmatrix} \sigma_W^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_\infty \end{bmatrix}$$

⑪ (Table 4.1-2)

$$\mu_{11}(.) = \mathcal{N}(\,.\mid \mathbf{c}y_n - \mathbf{A}\mathbf{y}_{n-1}, \mathbf{V}_{11})$$
$$\mathbf{V}_{11} = \mathbf{V}_{10} + \mathbf{A}\mathbf{V}_{n-1|\overrightarrow{n-1}}\mathbf{A}^{\mathsf{T}}$$

⑫ Estimation of the input u_n:

$$\mu_{12}(u_n) = \mathcal{N}(u_n \mid \hat{u}_n, V_{u_n})$$
$$\hat{u}_n = y_n - \mathbf{a}^{\mathsf{T}}\mathbf{y}_{n-1}$$
$$V_{u_n} = \left(\mathbf{b}^{\mathsf{T}}\mathbf{V}_{11}^{\#}\mathbf{b}\right)^{-1}$$

where $\mathbf{y}_{n-1} = (y_{n-1}, \ldots, y_{n-P})^{\mathsf{T}}$.

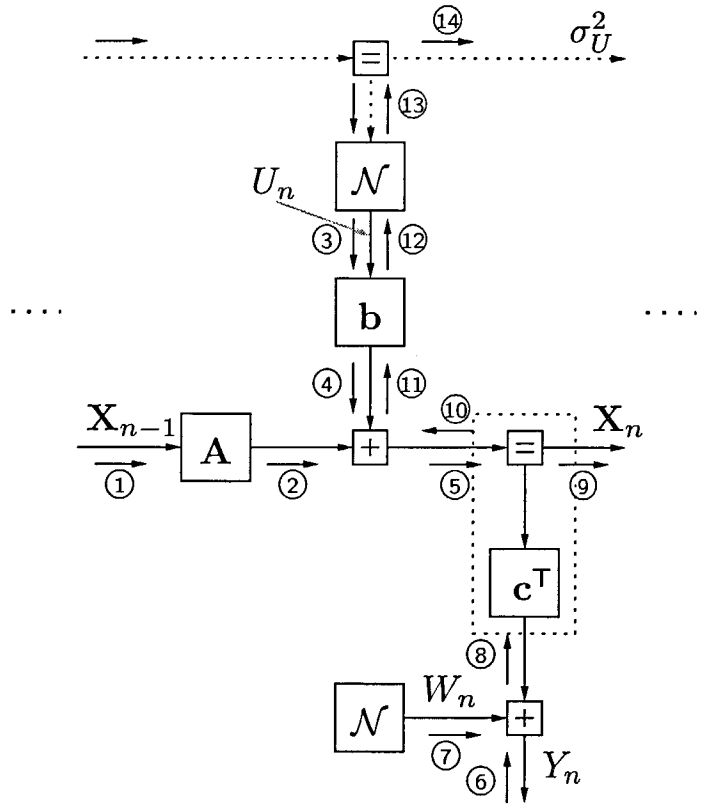


Figure 6.10: Factor graph for variance estimation of the input noise sequence of the AR model when observation noise is present.

⑬ Applying the node update equation (2.6):

$$\begin{aligned}\mu_{13}(\sigma^2) &\propto \int_{-\infty}^{\infty} \mathcal{N}\big(u_n \mid 0, \sigma^2\big)\, \mathcal{N}(u_n \mid \hat{u}_n, V_{u_n})\, du_n \\ &= \frac{1}{\sqrt{2\pi(\sigma^2 + V_{u_n})}} \exp\left(-\frac{\hat{u}_n^2}{2(\sigma^2 + V_{u_n})}\right) \\ &\propto \mathrm{Ig}\left(\sigma^2 + V_{u_n} \;\middle|\; -\frac{1}{2}, \frac{\hat{u}_n^2}{2}\right) \qquad (6.58)\end{aligned}$$

where Ig denotes an inverted gamma distribution; it is shifted by the amount of V_{u_n} (confer Appendix A.1.2).

⑭ Here we run into troubles, because the shifted inverted gamma distribution is not closed under multiplication anymore. Combining shifted versions of the inverted gamma distribution at the equality-nodes leads to a combinatorial explosion. After N slices the message would look like

$$\mu(\sigma_N^2) = \frac{1}{(2\pi)^{N/2}} \prod_{n=1}^{N} \frac{1}{(\sigma_N^2 + V_{u_n})^{N/2}} \exp\left(-\frac{1}{2}\sum_{n=1}^{N} \frac{\hat{u}_n^2}{\sigma_N^2 + V_{u_n}}\right) \quad (6.59)$$

In [124] such densities are called non-reproducing. Possible solutions are to find an approximation to (6.59) which is closed under multiplication (Section 6.3.2) or apply stochastic simulation methods, such as particle techniques (Section 6.3.1).

6.3 Joint Coefficient/Variance Estimation

In the foregoing sections algorithms have been derived for state estimation Section 4.3, coefficient estimation Section 6.1 and variance estimation Section 6.2. In this section we solve the joint estimation problem of the AR model specified in Section 3.2 by combining techniques applied in the former sections. This is simply done by constructing the corresponding graph and applying the messages already derived.

6.3.1 Particle filter

Fig. 6.11 shows the factor graph together with the messages of the forward pass of the update schedule used, Fig. 6.12 the backward sweep, which is only used for the iterative algorithm.

First, the state estimate is updated (messages ①-⑨), then the estimates of the parameters (messages ⑩-⑬ for the coefficients $\mathbf{a}$, messages ⑭-⑯ for the input noise variance σ_U^2 and messages ⑰-⑱ for the observation noise variance σ_W^2). After computing those messages for section n the algorithm proceeds with section $n+1$ until the end of the block is reached.

Then the same is carried out backwards. Messages ⑲-㉖ for the state estimate, messages ㉗-㉘ for the coefficient estimate, messages ㉙-㉛ for the input noise variance and messages ㉜-㉞ for the observation noise variance.

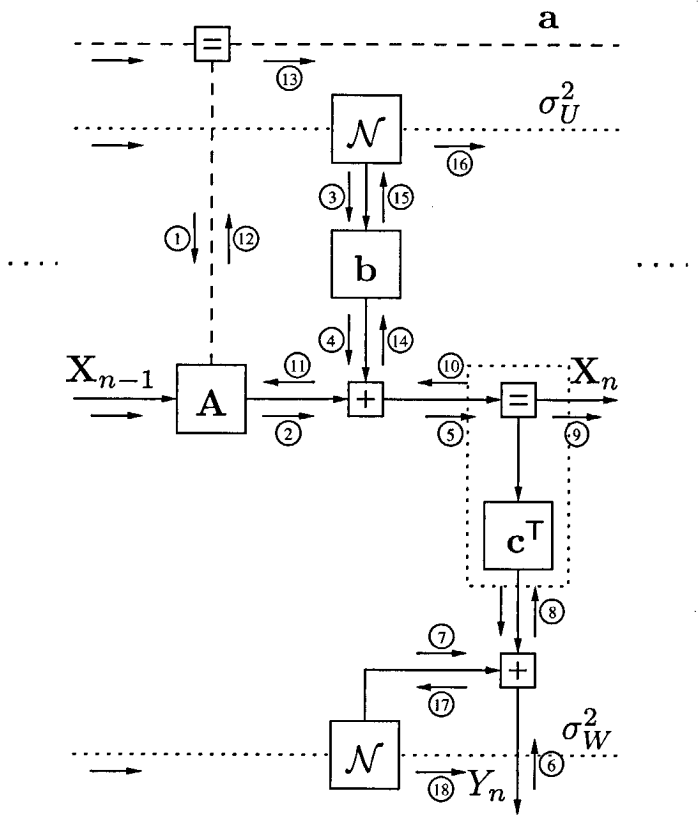


Figure 6.11: Factor graph for joint state/parameter estimation with messages for the forward pass.

Note that the forward messages already exist, so the estimation of the parameters during the backward pass is already based on smoothed estimates of the state and vice versa.

One strength of the factor graph approach is that different techniques can be used as building blocks to construct more sophisticated estimation algorithms. In our example, we may combine any of the mentioned algorithms for coefficient and variance estimation (e.g. Gradient techniques, approximated mode, particle filter).

Using a forward-only schedule clearly leads to suboptimal estimates. But as shown in the following such a schedule is often good enough and further iterations yield only insignificant further improvements.

Estimation errors for the coefficients are shown in Fig. 6.13(a), for the input noise variance in Fig. 6.13(b) and for the observation noise variance in Fig. 6.13(c). Again, the lines with markers in Fig. 6.13(a) and Fig. 6.13(b) indicate the estimation errors for the noise-free case.

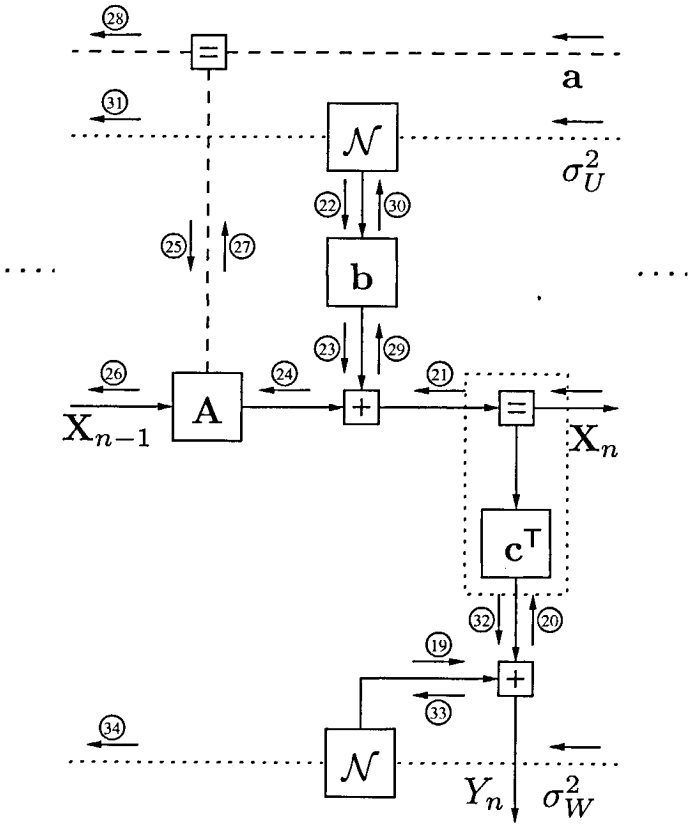
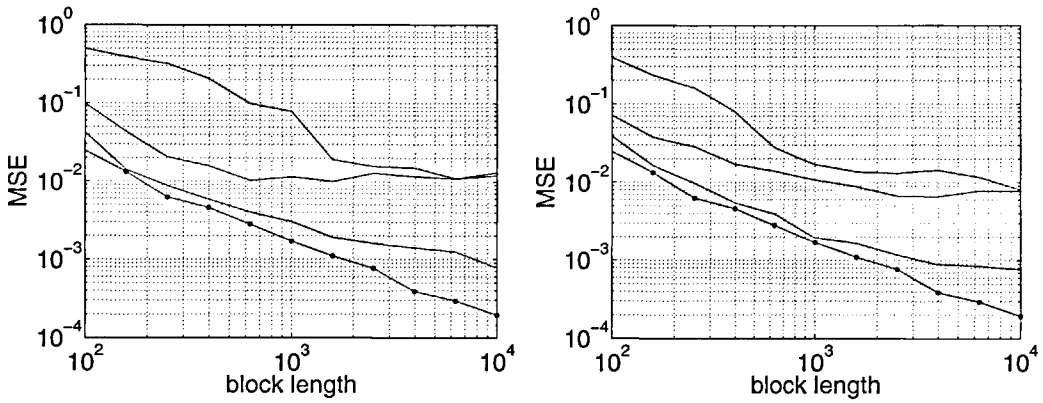


Figure 6.12: Factor graph for joint state/parameter estimation of the AR model with messages for the backward pass.

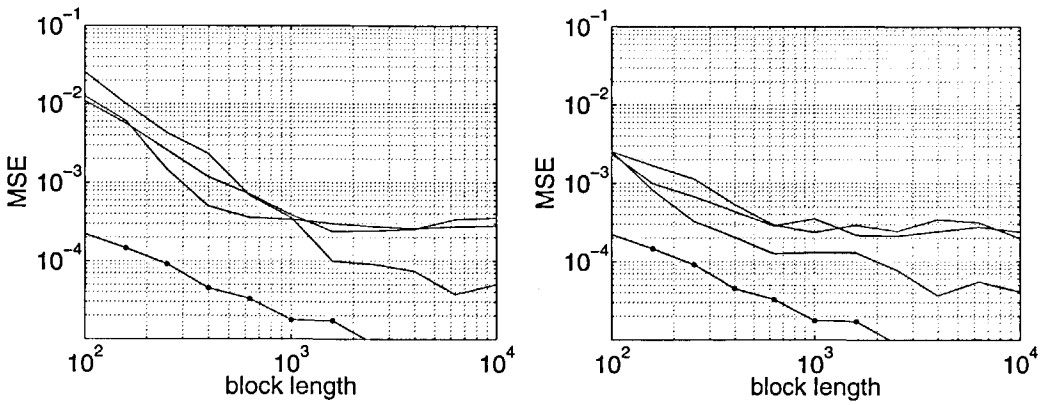
The results for forward-only processing are shown on the left-hand side, and after 3 iterations on the right-hand side. A number of 50 particles and shrinking was used to alleviate the effects for constant parameter estimation mentioned in Section 5.5.

As is obvious from Fig. 6.13, the performance of forward-only message passing is nearly as good as the performance of the iterative algorithm. This is surprising because messages ㉚ and ㉝ flowing into the Gaussian noise nodes are exactly the same if both the messages for σ_U^2 and σ_W^2 are initialised the same way. The symmetry breaks only when at least one full iteration is computed or both variances are initialised differently. The problem with the symmetry in joint estimation of σ_U^2 and σ_W^2 with forward-only messages passing is detailed in Appendix C. For the particle filter algorithms we used different initial particle lists for σ_U^2 and σ_W^2.

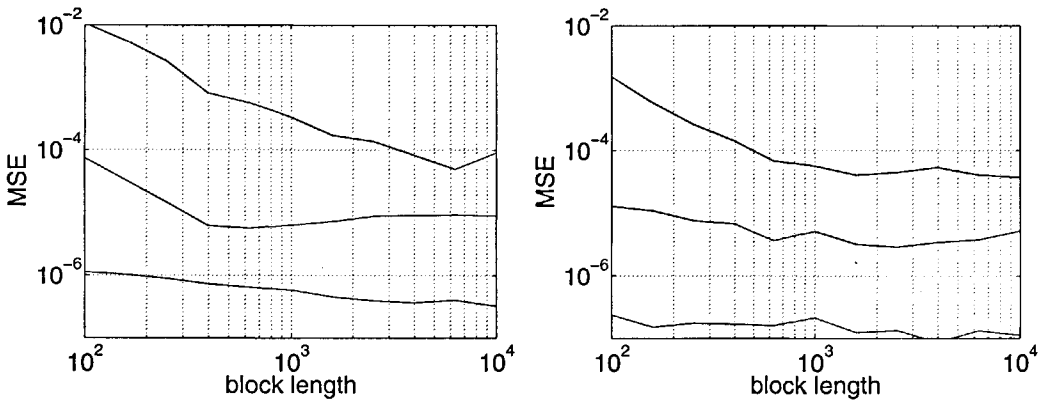
If the ranges of the (constant) parameters are known in advance, it is worth to look at the grid based algorithm. Here, the particles are fixed at their original positions. The results for this approach are shown in



(a) Coefficient estimation errors.



(b) Innovation variance estimation errors.



(c) Noise variance estimation errors.

Figure 6.13: Joint coefficient/noise estimation with **particle based** algorithm. Noise levels $\sigma_W^2 = 0.1/0.01/0.001$ and innovation variance $\sigma_U^2 = 0.1$. Left: Forward-only processing; Right: 3 iterations. The line with markers represents the noise-free case ($\sigma_W^2 = 0$).

Fig. 6.14. A number of 50 particles were used in equidistant positions from zero to four times the true parameter value.

6.3.2 Approximated mode

The factor graph for this problem is given in Fig. 3.1. Estimation errors for the coefficients are shown in Fig. 6.15(a), for the input noise variance in Fig. 6.15(b) and for the observation noise variance in Fig. 6.15(c). In the left-hand column are the results for the forward-only processing and in the right-hand column the results are shown after five iterations.

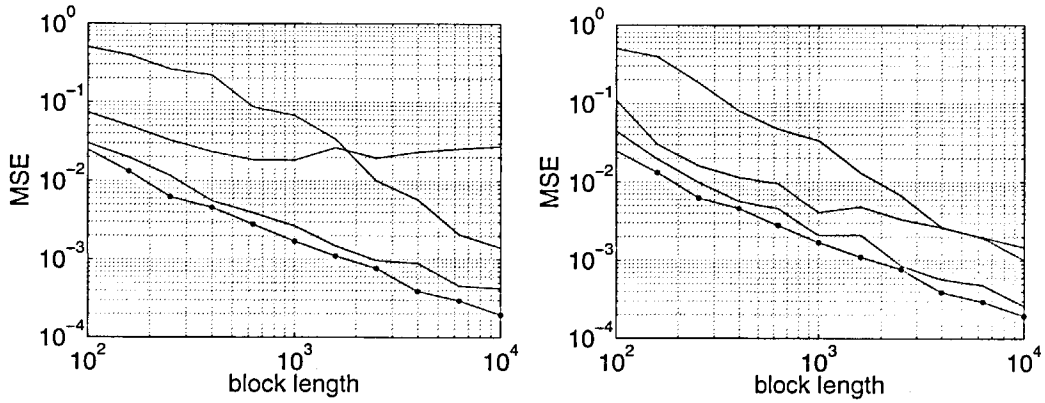
The estimation errors in the case of forward-only processing are very bad. This indicates that the estimation of the variances (which are based on the approximated mode) is sensitive to errors in the state estimate. Consequently, when allowing the algorithm to reestimate the parameters based on new estimates of the states, the results are much better. The right-hand column of Fig. 6.15 shows the estimation errors after five iterations.

A drawback of this approach to variance estimation is the numerical instability. Sometimes (in one out of approx. hundred cases), the actual estimate of the variance becomes negative. In this case, it is set to a small positive constant, which clearly spoils the whole estimation result. A possible solution would be to compute one iteration more or stop one iteration earlier. For the plots in Fig. 6.15 these cases were sorted out by a simple threshold detection algorithm.

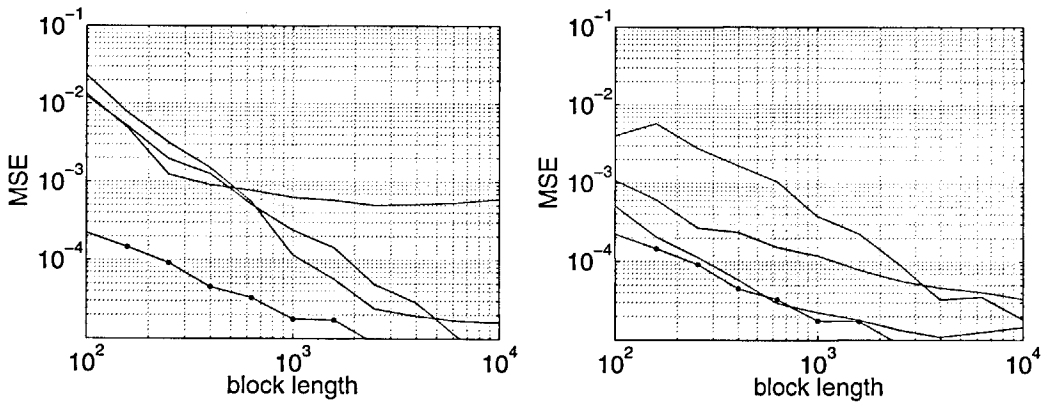
In addition, the bumpy curves in Fig. 6.15 indicate, that the estimation errors are highly scattered. In some simulations very good results are obtained, whereas in other simulations higher estimation errors may result.

6.3.3 Message passing EM

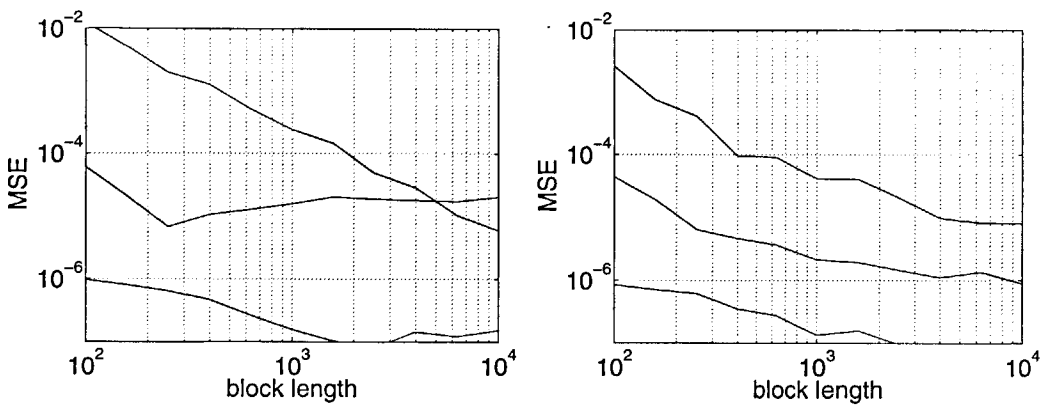
In this section a completely worked out example of using the message passing EM algorithm is presented. We consider the autoregressive model with unknown coefficients and unknown variances. One section of the factor graph of this model is depicted in Fig. 6.16.



(a) Coefficient estimation errors.

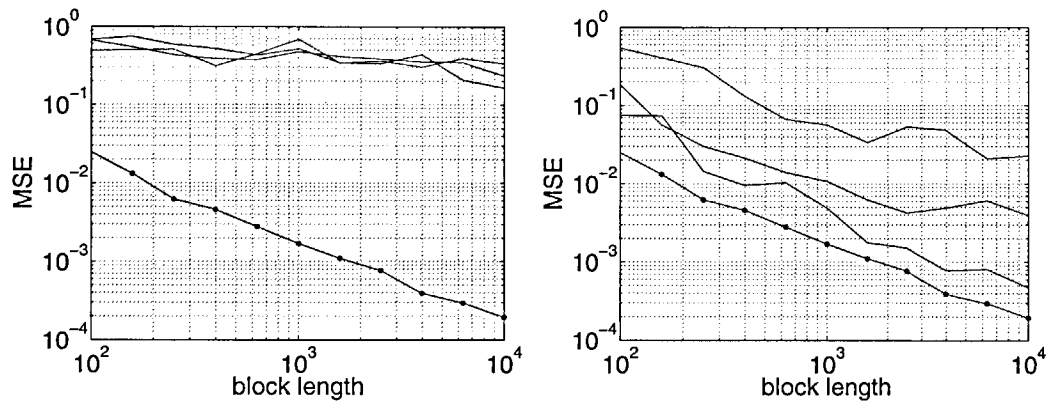


(b) Innovation variance estimation errors.

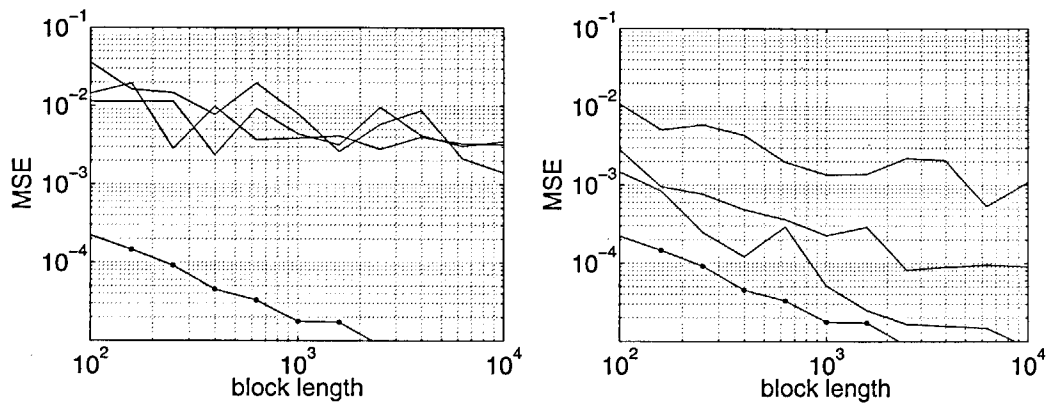


(c) Noise variance estimation errors.

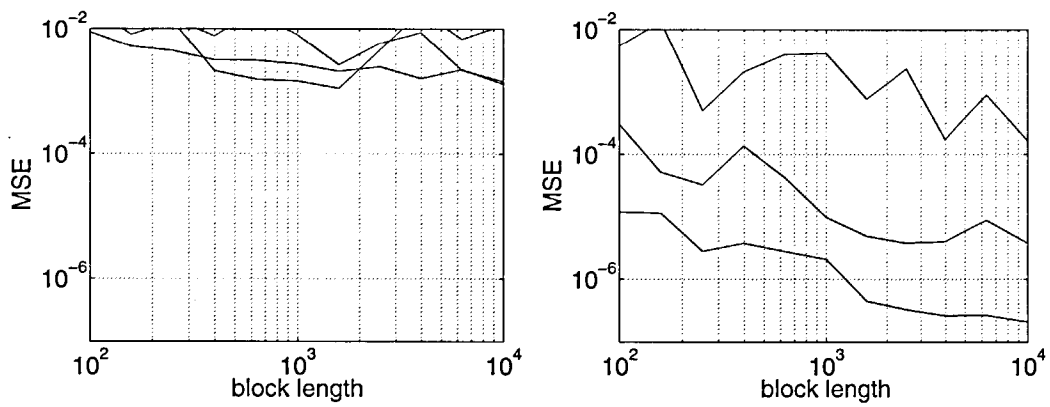
Figure 6.14: Joint coefficient/noise estimation with **grid based** algorithm. Noise levels $\sigma_W^2 = 0.1/0.01/0.001$ and innovation variance $\sigma_U^2 = 0.1$. Left: Forward-only processing; Right: 3 iterations. The line with markers represents the noise-free case ($\sigma_W^2 = 0$).



(a) Coefficient estimation errors.



(b) Innovation variance estimation errors.



(c) Noise variance estimation errors.

Figure 6.15: Joint coefficient/noise estimation with the technique of **approximated mode**. Noise levels $\sigma_W^2 = 0.1/0.01/0.001$ and innovation variance $\sigma_U^2 = 0.1$. Left-hand: Forward-only processing; Right-hand: 5 iterations. The line with markers represents the noise-free case ($\sigma_W^2 = 0$).

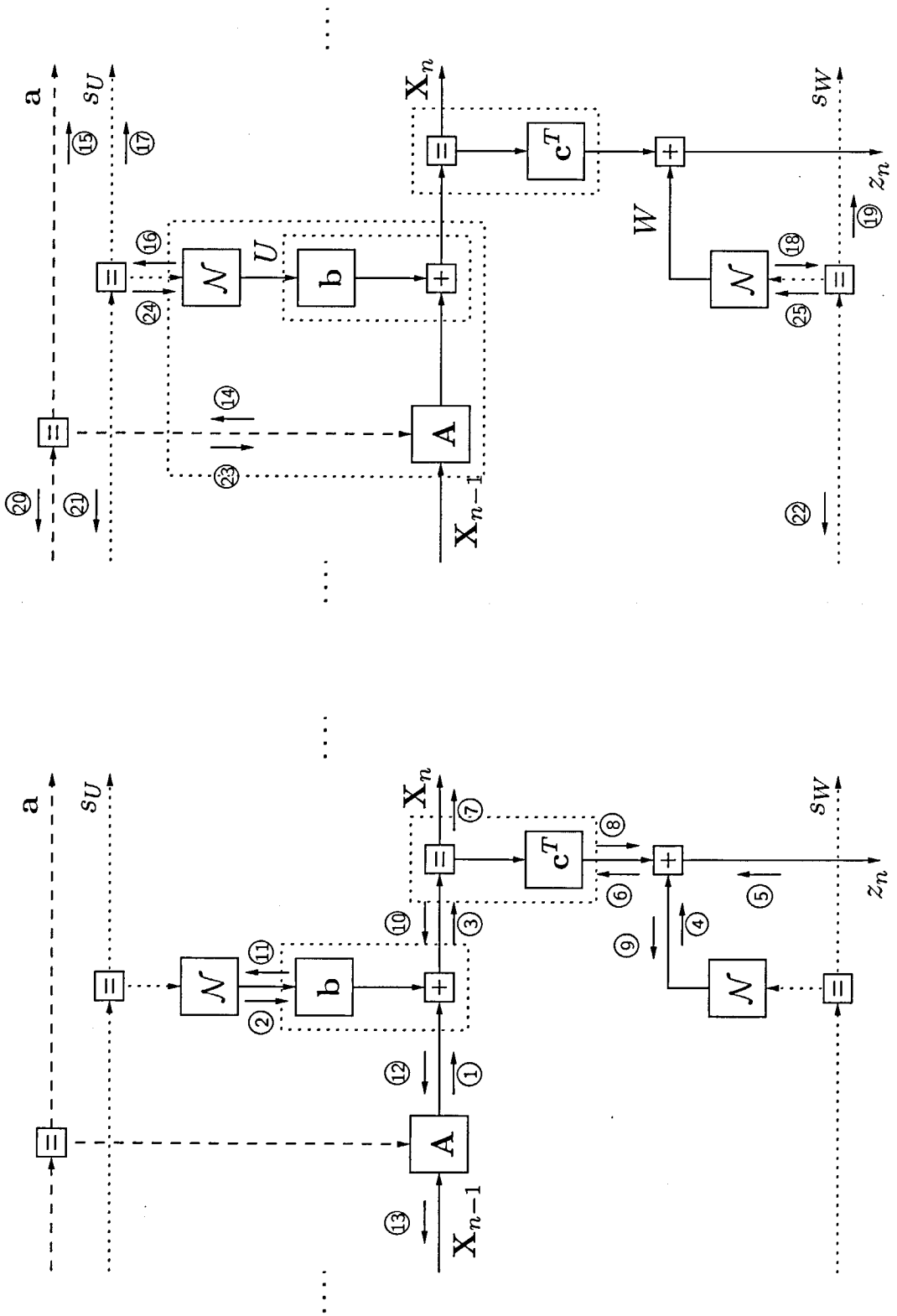


Figure 6.16: Factor graph for joint state/parameter estimation of the AR model with EM messages. Left: E-step, Right: M-step.

For the E-step (Fig. 6.16 left), messages ① to ⑦ comprise the forward sweep and messages ⑧ to ⑬ the backward sweep. Messages ① to ⑦ are computed for one section followed by the section to the right and so on. This leads to the standard Kalman filter formulas given the actual parameter estimates $\mathbf{a}^{(k)}$ for the coefficients and $(\sigma_U^2)^{(k)}$ and $(\sigma_W^2)^{(k)}$ for the input and noise variance, respectively (cf. Section 4.3.1). In the backward sweep, messages ⑧ to ⑬ are computed in the same manner, but from right to left. The forward and the backward sweep constitute the Kalman smoother.

The M-step re-estimates the parameters $\mathbf{a}$, σ_U^2 and σ_W^2 and proceeds as follows. It is again separated into a forward and backward sweep. The forward sweep consists of messages ⑭ to ⑲ (Fig. 6.16 right):

⑭ e^h-message (Table 5.5-11):

$$e^{h(\mathbf{a})} \propto \mathcal{N}_W(\mathbf{a} \mid \mathbf{m}_{14}, \mathbf{W}_{14}) \tag{6.60}$$

with

$$\mathbf{m}_{14} = \mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathsf{T}}]^{-1}\mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_n^{[1]}] \qquad \mathbf{W}_{14} = \frac{\mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathsf{T}}]}{\sigma_U^2} \tag{6.61}$$

where the subscript 14 denotes the message number.

⑮ Gaussian equality node (Table 4.1-1):

$$e^{h(\mathbf{a}_n)} \propto \mathcal{N}_W(\mathbf{a}_n \mid \hat{\mathbf{a}}_n, \mathbf{W}_{\mathbf{a}_n}) \tag{6.62}$$

with

$$\hat{\mathbf{a}}_n = (\mathbf{W}_{\mathbf{a}_{n-1}} + \mathbf{W}_{14})^{-1}(\mathbf{W}_{\mathbf{a}_{n-1}}\hat{\mathbf{a}}_{n-1} + \mathbf{W}_{14}\mathbf{m}_{14}) \tag{6.63}$$

$$\mathbf{W}_{\mathbf{a}_n} = \mathbf{W}_{\mathbf{a}_{n-1}} + \mathbf{W}_{14} \tag{6.64}$$

⑯ (Table 5.5-11):

$$e^{h(\sigma_U^2)} \propto \mathrm{Ig}\left(\sigma_U^2 \;\middle|\; -\frac{1}{2}, \frac{\mathsf{E}[(\mathbf{X}_n - \hat{\mathbf{A}}\mathbf{X}_{n-1})^{\mathsf{T}}(\mathbf{X}_n - \hat{\mathbf{A}}\mathbf{X}_{n-1})]}{2}\right) \tag{6.65}$$

With the special structure of $\mathbf{A}$ this can be simplified to:

$$e^{h(\sigma_U^2)} \propto \mathrm{Ig}\left(\sigma_U^2 \mid \alpha_{16}, \beta_{16}\right) \tag{6.66}$$

with

$$\alpha_{16} = -1/2 \tag{6.67}$$

$$\beta_{16} = \frac{\mathsf{E}[X_n^2] - 2\hat{\mathbf{a}}^{\mathsf{T}}\mathsf{E}[\mathbf{X}_{n-1}X_n] + \hat{\mathbf{a}}^{\mathsf{T}}\mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathsf{T}}]\hat{\mathbf{a}}}{2} \tag{6.68}$$

where X_n is the first element of the vector $\mathbf{X}_n$.

⑰ equ-node for inverted gamma messages (Table 5.2):

$$e^{h(\sigma_{U_n}^2)} \propto \mathrm{Ig}\left(\sigma_{U_n}^2 \mid \alpha_n, \beta_n\right) \tag{6.69}$$

with α_n and β_n computed according to Table 5.2:

$$\alpha_n = \alpha_{n-1} + 1/2 \tag{6.70}$$

$$\beta_n = \beta_{n-1} + \beta_{16} \tag{6.71}$$

⑱ (Table 5.5-3): Like message ⑯, but for σ_W^2.

⑲ equ-node for Ig: Like message ⑰, but for σ_W^2.

And the backward sweep consists of messages ⑳ to ㉕:

⑳ (Table 4.1-1)

㉑ Ig: Like message ⑰.

㉒ Ig: Like message ⑰.

㉓ This message is the combination of the forward and backward messages (Table 4.1-1). The new estimate $\hat{\mathbf{a}}$ is the mode of the total message at this edge.

$$\hat{\mathbf{a}} = \left(\sum_{n=1}^{N} \mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathsf{T}}]\right)^{-1} \left(\sum_{n=1}^{N} \mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_n^{[1]}]\right) \tag{6.72}$$

where

$$\mathsf{E}[\mathbf{X}_0\mathbf{X}_0^{\mathsf{T}}] = \mathbf{0} \qquad \mathsf{E}[\mathbf{X}_0\mathbf{X}_1^{[1]}] = \mathbf{0}. \tag{6.73}$$

㉔ Combination of forward and backward messages (Table 5.2). The new estimate $\hat{\sigma}_U^2$ is the mode of the total message at this edge.

$$e^{h_{\text{tot}}(\sigma_U^2)} = \text{Ig}\left(\sigma_U^2 \mid \alpha_{\text{tot}}, \beta_{\text{tot}}\right) \tag{6.74}$$

$$\alpha_{\text{tot}} = N - 1 + \sum_{n=1}^{N} \alpha_n = \frac{N}{2} - 1 \tag{6.75}$$

$$\beta_{\text{tot}} = \sum_{n=1}^{N} \beta_n \tag{6.76}$$

$$= \frac{1}{2} \sum_{n=1}^{N} \left(\mathsf{E}[X_n^2] - 2\hat{\mathbf{a}}^{\mathsf{T}}\mathsf{E}[\mathbf{X}_{n-1}X_n] + \hat{\mathbf{a}}^{\mathsf{T}}\mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathsf{T}}]\hat{\mathbf{a}}\right) \tag{6.77}$$

$$\hat{\sigma}_U^2 = \frac{\beta_{\text{tot}}}{\alpha_{\text{tot}} + 1} \tag{6.78}$$

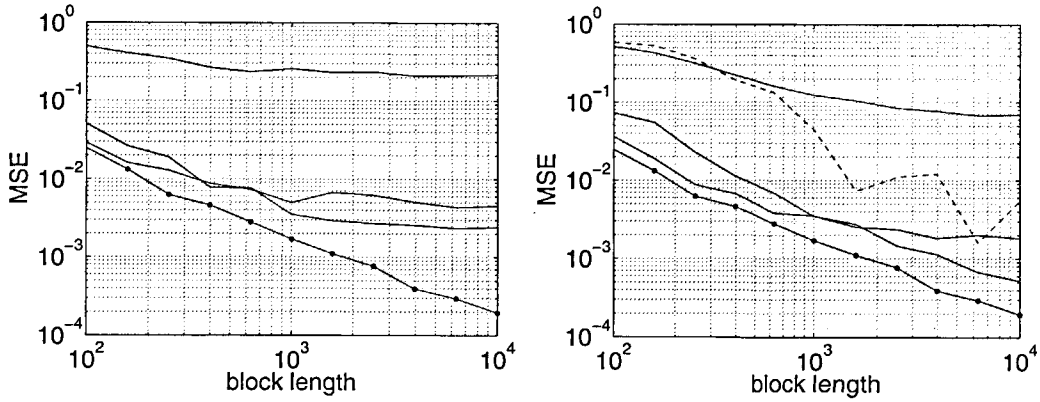
$$= \frac{1}{N} \sum_{n=1}^{N} \left(\mathsf{E}[X_n^2] - 2\hat{\mathbf{a}}^{\mathsf{T}}\mathsf{E}[\mathbf{X}_{n-1}X_n] + \hat{\mathbf{a}}^{\mathsf{T}}\mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathsf{T}}]\hat{\mathbf{a}}\right) \tag{6.79}$$

㉕ Computed like message ㉔ but for σ_W^2.

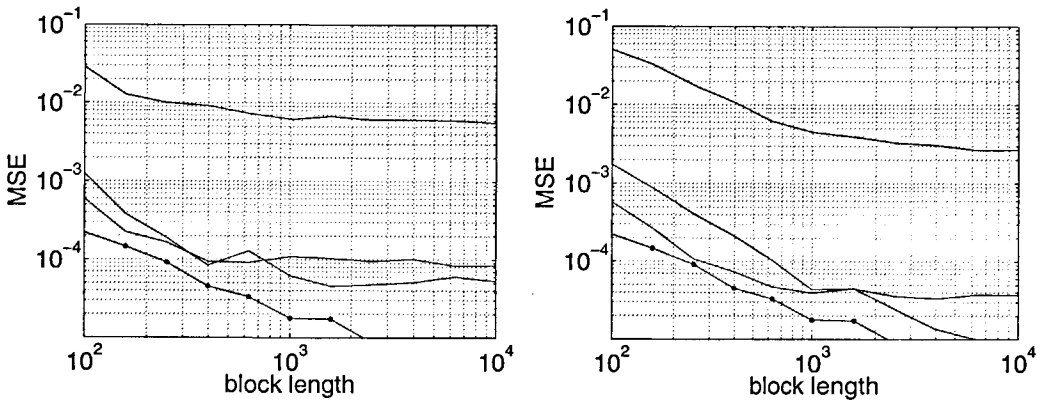
For the EM messages ⑭, ⑯ and ⑱ the expectations $\mathsf{E}[X_n^2]$, $\mathsf{E}[\mathbf{X}_{n-1}X_n]$ and $\mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathsf{T}}]$ are needed. The correlations $\mathsf{E}[X_n^2]$ and $\mathsf{E}[\mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathsf{T}}]$ can simply be read off the (total) sum-product messages of the E-step. The cross-correlation $\mathsf{E}[\mathbf{X}_{n-1}X_n]$ is somewhat trickier to compute, since it involves variables on two different edges. The simplest way to compute this cross-correlation is to augment the state-space by one dimension, so that X_n and $\mathbf{X}_{n-1}$ appear on the same edge. A different alternative is to explicitely compute those expectations in the node, which is shown in detail in Appendix E.6.

The resulting estimation errors are shown in Fig. 6.17(a) for the coefficients, in Fig. 6.17(b) for the input noise variance and in Fig. 6.17(c) for the observation noise variance. The plots in the left-hand column show the results after 20 iterations, whereas the right-hand column shows the results after 40 iterations.

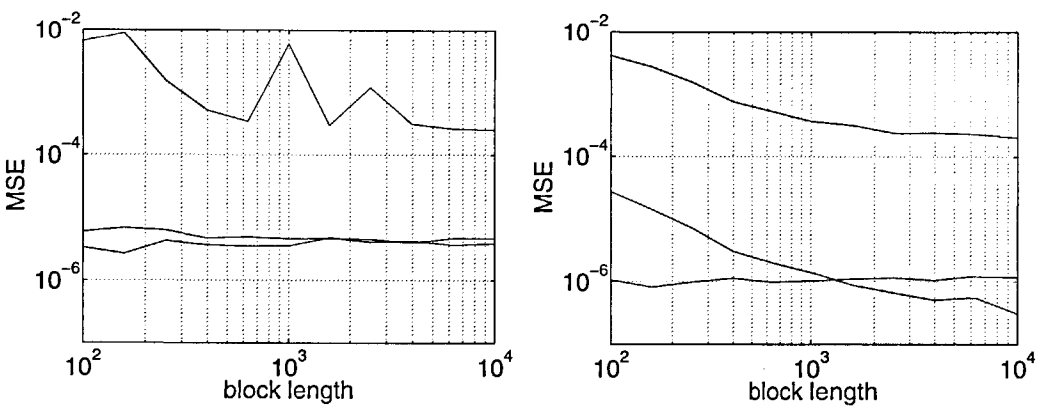
The dashed line in the top right plot represents the estimation errors of the coefficients for the high noise situation ($\sigma_W^2 = 0.1$) after 100 iterations. Since this number of iterations is prohibitively large, only this one case has been computed.



(a) Coefficient estimation errors.



(b) Innovation variance estimation errors.



(c) Noise variance estimation errors.

Figure 6.17: Joint coefficient/noise estimation with **EM**. Noise levels $\sigma_W^2 = 0.1/0.01/0.001$ and innovation variance $\sigma_U^2 = 0.1$. Left: 20 iterations, Right: 40 iterations (dashed line 100 iterations). The line with markers represents the noise-free case ($\sigma_W^2 = 0$).

An obvious disadvantage of the EM algorithm is that many iterations are needed to get satisfactory results. This is especially true if the likelihood surface is flat near the optimum, i.e. in high noise situations. Another point is that the EM algorithm converges to the next local optimum, so it can get trapped in a bad solution, globally seen. Consequently, a good choice of initial values is vital.

6.4 Summary

- Using the factor graph language developed in the foregoing chapters a couple of estimation algorithms have been developed. The autoregressive model has been used as a common basis.
- A first assumption has been that the input noise variance and the observation noise variance are known and the coefficients of the AR model have to be estimated. When there is no observation noise present at all, the resulting algorithm resembles the classic linear predictive coding (LPC) algorithm. The LPC algorithm fails in the noisy case. The iterative algorithm we proposed compensates the spoiling effect of the observation noise.
- The recursive least squares (RLS) adaptive filter and the least mean squares (LMS) adaptive filter have been treated. They may also be derived from the factor graph representation of the state-space model.
- A special difficulty in variance estimation is the appearance of non-Gaussian messages. With no observation noise it is possible to compute the messages, which are inverted gamma functions, exactly. If there is observation noise present in the AR model, we have to resort to approximative techniques.
- The joint estimation problem, where the coefficients as well as the variances of the AR model have to be estimated, has been approached by the particle method and the message passing EM method.
- In conclusion, in this chapter we demonstrated the application of the factor graph approach to signal modelling and estimation by means of different estimation tasks in the autoregressive model.

Chapter 7

Overall Conclusions and Outlook

7.1 Summary

This thesis is about a graphical approach to model-based signal processing. In model-based signal processing signals and systems are defined by stochastic equations. Given such a description of the system/signal at hand and some observations, the aim of system identification and filtering is to determine the 'best' (e.g. most probable) system/signal out of the class of systems/signals modelled.

With our approach it is possible to use more complex models than has been possible so far. For existing models only specialised estimation algorithms have been available. Hidden Markov models (finite state-space) are used with the Baum-Welch algorithm or the Viterbi algorithm, continuous state-space models with Kalman filter type algorithms, for instance. We adapt and extend the framework of factor graphs to be able to model more complex models and to derive estimation algorithms (filtering, denoising, parameter estimation, etc.) for such models.

A factor graph consists of nodes and edges and represents a factorisation of a function of many variables, where edges correspond to variables

and nodes correspond to relations between variables (the factors of the function represented by the graph). Inference about the value of an unknown variable is carried out by passing messages along the edges of the graph.

The aim of this thesis is threefold. First, we show that it is possible to systematically derive classic model-based signal processing algorithms within the framework of factor graphs. Second, once classic algorithms are stated in the factor graph language, they can easily be combined for models with many unknown parameters; and third, we show that factor graphs provide a framework for the systematic development of completely new algorithms.

All developments in this thesis are demonstrated by means of the autoregressive (AR) model. The parameters of the AR model, the coefficient vector, the input noise variance and the observation noise variance, are either known or unknown. The factor graph of the AR model is detailed in Chapter 3.

Kalman filtering algorithms, the linear predictive coding algorithm and the recursive least squares adaptive filter algorithm can be derived from factor graph models where every variable is Gaussian. In that case update rules for the messages sent along the edges of the graph take on a simple form. In addition, as has been shown in [83], reusable linear building blocks are defined (Chapter 4).

We propose several techniques to handle models where messages cannot be represented by Gaussian functions. Sometimes making a tentative decision about the value of one variable renders the relevant message Gaussian (Section 5.1); in other cases it may be possible to represent the non-Gaussian message by a finite set of parameters (Section 5.2), etc. Gradient methods as well as particle methods can also be incorporated into the factor graph framework. The messages comprise therefore gradients or lists of samples of the exact message function.

The expectation maximisation (EM) algorithm has been the main competitor to message passing algorithms for parameter estimation. In this thesis, we show that the EM algorithm can itself be viewed as message passing on factor graphs (Section 5.6). We devise a message passing EM update rule, similar to the sum-product update rule. So it is easy to derive an EM based parameter estimation algorithm once the model

is stated in the factor graph notation.

Variants of the classic expectation maximisation algorithm can be created by altering the message update schedule, which may result in faster convergence. Additionally, a local EM update rule is devised. This local update rule implements a mixture form of the sum-product and message passing EM update rule, with the advantage that it neglects less information than the classic EM update rule.

In many simulations it is shown that the algorithms based on message passing on the factor graph work well. Different techniques have been compared to demonstrate the advantages and disadvantages of every approach.

7.2 Future work

In this last section some thoughts are sketched about how the work presented in this thesis could be continued. They can roughly be classified into three lines.

Factor graph tools and theory:

- *Different representations for Gaussian messages.* In Section 4.1 it has been mentioned that Gaussian messages can be represented in many different ways. The effects of different representations on the performance or complexity of the resulting algorithm should be further analysed. Also the technique of structured messages [21] can be applied in the context of Gaussian models.
- *Particle messages in the context of loopy graphs.* Particle filters are well studied in the context of online estimation (i.e. forward-only processing in our terminology). A completely new situation arises when the graph has cycles and particle messages are recomputed. New ways of dealing with this situation have to be developed.
- *Local EM update rule.* A new kind of EM update rule has been proposed based on a mixture of ideas from sum-product and EM. Its operability has been shown in a few simulations only. Further analysis has to be carried out to show possible advantages of this new scheme.

Modelling issues:

- *Real-time implementation.* Some of the presented algorithms in this thesis are suited for real-time operation. This should consequently be utilised in a real-time version.

- *Sophisticated speech models.* There are many extensions to the basic AR model around, especially in the speech coding and speech recognition literature. For example, typical AR coefficient vectors of speech signals are not equally distributed but can be clustered into different regions. This can be utilised by vector-quantisation of the AR coefficient vector.
  Voiced/unvoiced segments of speech could be distinguished and treated differently. For voiced segments a harmonic excitation may be applied, etc.

- *Noise models.* So far, noise was treated only as 'everything that is not speech'. If the character of the perturbing noise is known, better noise models may be incorporated.

New applications:

- *Classification.* The factor graph can in principle be used to perform classification. Different ways are conceivable. For example, the factor graph can represent the likelihood function used by the classifier. A problem arises in loopy graphs, because the likelihood is scaled by an unknown factor. The scaling factor may be computed (at least approximately) by free-energy based approaches [89, 143, 144].

- *Machine Learning.* The connections between parameter estimation and machine learning are manifold. Many machine learning algorithms can be represented as message passing on a factor graph. On the other hand, learning methods could be applied to learn an unknown node function.

Appendix A

Mathematical Background Material

A.1 Some Distributions

A.1.1 The Gauss distribution

A continuous random quantity X has a Gauss distribution (or normal distribution) with parameters m and v ($m, v \in \mathbb{R}, v > 0$) if its density function $\mathcal{N}(x \mid m, v)$ is

$$\mathcal{N}(x \mid m, v) = \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{(x-m)^2}{2v}\right) \qquad x \in \mathbb{R}. \tag{A.1}$$

The mean and mode are $\mathsf{E}[X] = \mathsf{M}[X] = m$ and the variance is $\mathsf{V}[X] = v$. Alternatively, $\mathcal{N}(x \mid m, v)$ is denoted by $\mathcal{N}_W(x \mid m, w)$, where $w = v^{-1}$ is called the precision.

In the vector case the normal distribution is defined as follows:

$$\mathcal{N}_W(\mathbf{x} \mid \mathbf{m}, \mathbf{W}) = \sqrt{\frac{|\mathbf{W}|}{(2\pi)^n}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathsf{T}}\mathbf{W}(\mathbf{x}-\mathbf{m})\right) \qquad \mathbf{x} \in \mathbb{R}^n \tag{A.2}$$

and if $\mathbf{V} = \mathbf{W}^{-1}$ exists as

$$\mathcal{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{V}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^\mathsf{T}\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m})\right) \qquad \mathbf{x} \in \mathbb{R}^n \tag{A.3}$$

The matrix $\mathbf{W}$ is also the negative Hessian matrix (i.e. the second derivative, or curvature) of $\log \mathcal{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{W})$ at the mean $\mathbf{m}$:

$$W_{ij} = -\frac{d^2}{dx_i dx_j} \log \mathcal{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{W}) \Big|_{\mathbf{x}=\mathbf{m}} \tag{A.4}$$

The higher the certainty about the random variable $\mathbf{x}$, the peakier the distribution, the higher the curvature, the larger the entries in $\mathbf{W}$. Therefore, the matrix $\mathbf{W}$ is also called weight matrix or information matrix.

A.1.2 The inverted gamma distribution

A continuous random quantity X has an inverted-gamma distribution with parameters α and β ($\alpha, \beta \in \mathbb{R}$) if its density function $\mathrm{Ig}(x \mid \alpha, \beta)$ is

$$\mathrm{Ig}(x \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{-(\alpha+1)} e^{-\frac{\beta}{x}} \qquad x > 0 \tag{A.5}$$

The mean, variance and mode of the distribution are

$$\mathsf{E}[X] = \frac{\beta}{\alpha - 1} \qquad \alpha > 1 \tag{A.6}$$

$$\mathsf{V}[X] = \frac{\beta^2}{(\alpha-1)^2(\alpha-2)} \qquad \alpha > 2 \tag{A.7}$$

$$\mathsf{M}[X] = \frac{\beta}{\alpha + 1} \tag{A.8}$$

A.2 The Matrix Inversion Lemma

This identity is useful because it shows how a matrix changes if something is added to its inverse. It is variously called the Matrix Inversion Lemma, Sherman-Morrison formula or Sherman-Morrison-Woodbury formula [13].

$$\left(A^{-1} + VC^{-1}V^{\mathsf{T}}\right)^{-1} = A - AV(C + V^{\mathsf{T}}AV)^{-1}V^{\mathsf{T}}A \tag{A.9}$$

This formula is especially important when v is a vector and c a scalar. In that case (A.9) becomes

$$\left(A^{-1} + \frac{vv^{\mathsf{T}}}{c}\right)^{-1} = A - \frac{Avv^{\mathsf{T}}A}{c + v^{\mathsf{T}}Av}. \tag{A.10}$$

A.3 Integrating Gaussian Densities

Let $q(x, y)$ be a quadratic form with the following partitioning

$$q(x,y) = \left((x - m_X)^{\mathsf{T}}, (y - m_Y)^{\mathsf{T}}\right) \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} \begin{pmatrix} x - m_X \\ y - m_Y \end{pmatrix} \tag{A.11}$$

with W_{11} positive definite and symmetric and $W_{12} = W_{21}^{\mathsf{T}}$. Then

$$\int_{-\infty}^{\infty} e^{-q(x,y)} dx \propto e^{-\min_x q(x,y)} \tag{A.12}$$

$$= \max_x e^{-q(x,y)} \tag{A.13}$$

Proof:
Reordering the terms of the quadratic form $q(x, y)$ and expanding it with $-\min_x q(x, y)$:

$$q(x,y) = x^{\mathsf{T}}W_{11}x - 2x^{\mathsf{T}} \underbrace{(W_{11}m_X - W_{12}\,y + W_{12}m_Y)}_{W_{11}\,m} + k(y) \tag{A.14}$$

$$= x^{\mathsf{T}}W_{11}x - 2x^{\mathsf{T}}W_{11}m + k(y) - \min_x q(x,y) + \min_x q(x,y) \tag{A.15}$$

where $k(y)$ covers every term of $q(x, y)$ that does not depend on x. With

$$\min_x q(x,y) = k(y) - m^{\mathsf{T}}W_{11}m \tag{A.16}$$

the quadratic form becomes

$$q(x,y) = x^{\mathsf{T}}W_{11}x - 2x^{\mathsf{T}}W_{11}m + m^{\mathsf{T}}W_{11}m + \min_x q(x,y) \tag{A.17}$$

$$= (x - m)^{\mathsf{T}}W_{11}(x - m) + \min_x q(x,y) \tag{A.18}$$

Hence, the integral becomes

$$\int_{-\infty}^{\infty} e^{-q(x,y)}dx = e^{-\min_x q(x,y)} \cdot \int_{-\infty}^{\infty} e^{-(x-m)^T W_{11}(x-m)}dx \quad \text{(A.19)}$$

$$\propto e^{-\min_x q(x,y)} \quad \text{(A.20)}$$

$$= \max_x e^{-q(x,y)} \quad \text{(A.21)}$$

Appendix B

Derivation of some Message Update Rules

B.1 Gradient Message Update Rules

In this section the proof of the gradient node update rule of Section 5.4 is given. Suppose we have the trivial factorisation

$$f(\theta) = f_A(\theta) \cdot f_B(\theta) \tag{B.1}$$

and want to find

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} f(\theta) \tag{B.2}$$

by solving

$$\nabla \log f(\theta) \stackrel{!}{=} 0. \tag{B.3}$$

(B.3) by the product rule splits into two separate terms

$$\nabla \log(f_A(\theta) f_B(\theta)) = \nabla \log f_A(\theta) + \nabla \log f_B(\theta). \tag{B.4}$$

This proofs the equality node update rule in Table 5.3. The gradient can then be used to find the general estimate $\hat{\theta}$ by iteration

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda \nabla \log f(\theta)\big|_{\theta=\hat{\theta}^{(k)}} \tag{B.5}$$

where λ is a step-size parameter.

This procedure can be performed by message passing on the factor graph. The corresponding graph is depicted in Fig. B.1. The nodes f_A

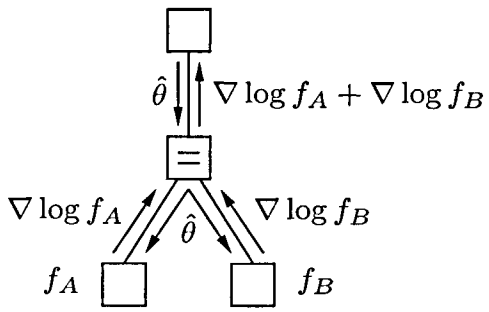


Figure B.1: Factor graph node for the generic gradient descend.

and f_B may replace subgraphs attached at the corresponding position.

First, an initial estimate $\hat{\theta}$ is distributed to every node depending on that parameter. Every node responds by sending the message $\nabla \log f(\theta)$ at the current estimate $\hat{\theta}$:

$$\nabla \log f(\hat{\theta}) = \nabla_\theta \log f(\theta)\big|_{\theta=\hat{\theta}} \tag{B.6}$$

e.g. the node f_A replies with $\nabla \log f_A(\hat{\theta})$ and the node f_B replies with $\nabla \log f_B(\hat{\theta})$. The nodes f_A and f_B may also depend on some other variables X. Those variables X are integrated out by the sum-product rule (2.6). The gradient messages becomes therefore

$$\nabla \log f(\hat{\theta}) = \nabla_\theta \log \int_{\mathcal{D}_x} f(x,\theta)\mu_{X\to f}(x)\,dx\Big|_{\theta=\hat{\theta}} \tag{B.7}$$

The integral on the right hand side is just the standard sum-product message leaving the node along edge θ. So (B.7) can be written in terms of this outgoing message:

$$\nabla \log f(\hat{\theta}) = \nabla_\theta \log \mu_{f\to\theta}(\theta)\big|_{\theta=\hat{\theta}} \tag{B.8}$$

This proofs (5.31) and (5.32).

B.2 Particle Message Update Rules

In this section, the particle message update rules of Table 5.4 shall be proven. We refer to the situation depicted in Fig. B.2. Message μ_1 is

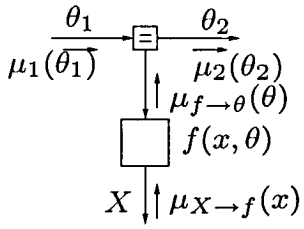


Figure B.2: Factor graph for the proof of the particle message update.

defined as follows:

$$\mu_1(\theta_1) = \sum_{i=1}^{M} \rho_1^{(i)} \delta(\theta_1 - \theta_1^{(i)}). \tag{B.9}$$

The message out of node f is:

$$\mu_{f\to\theta}(\theta) = \int_{\mathcal{D}_x} f(x,\theta)\mu_{X\to f}(x)dx \tag{B.10}$$

Applying the sum-product update rule (2.6) for μ_2:

$$\mu_2(\theta_2) = \int_{\mathcal{D}_\theta}\int_{\mathcal{D}_{\theta_1}} \delta(\theta_1-\theta_2)\delta(\theta-\theta_2)\mu_{f\to\theta}(\theta)\mu_1(\theta_1)d\theta d\theta_1 \tag{B.11}$$

$$= \mu_{f\to\theta}(\theta_2)\cdot\mu_1(\theta_2) \tag{B.12}$$

$$= \int_{\mathcal{D}_x} f(x,\theta_2)\mu_{X\to f}(x)dx \cdot \sum_{i=1}^{M} \rho_1^{(i)}\delta(\theta_2-\theta_1^{(i)}) \tag{B.13}$$

$$= \sum_{i=1}^{M} \underbrace{\rho_1^{(i)} \int_{\mathcal{D}_x} f(x,\theta_1^{(i)})\mu_{X\to f}(x)dx}_{\rho_2^{(i)}} \delta(\theta_2 - \underbrace{\theta_1^{(i)}}_{\theta_2^{(i)}}) \tag{B.14}$$

$$= \sum_{i=1}^{M} \rho_2^{(i)}\delta(\theta_2-\theta_2^{(i)}) \tag{B.15}$$

This proofs the equations in the first line of Table 5.4.

B.3 Coefficient Estimation, Sum-Product

In the following, the update formulas for the state transition node in state-space model, where the (linear) transition coefficients are unknown, is derived. Fig. B.3(a) shows the node in the general situation, where no assumptions are made on the state variables before ($\mathbf{x}_f$) and after ($\mathbf{x}_b$) the transition.

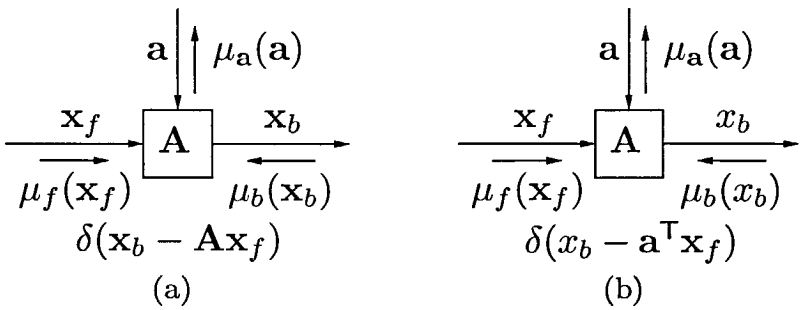


Figure B.3: Node A for coefficient estimation in higher-order auto-regression.

The node function is $f(\mathbf{x}_f, \mathbf{x}_b, \mathbf{a}) = \delta(\mathbf{x}_b - \mathbf{A}\mathbf{x}_f)$ where

$$\mathbf{A} \triangleq \begin{bmatrix} \mathbf{a}^{\mathsf{T}} \\ \mathbf{I} \quad \mathbf{0} \end{bmatrix} \tag{B.16}$$

and the incoming message $\mu_f(\mathbf{x}_f)$ is

$$\mu_f(\mathbf{x}_f) = \mathcal{N}(\mathbf{x}_f \mid \hat{\mathbf{x}}_f, \mathbf{V}_f) \propto \exp(-q_f(\mathbf{x}_f)) \tag{B.17}$$

with

$$q_f(\mathbf{x}_f) = (\mathbf{x}_f - \hat{\mathbf{x}}_f)^{\mathsf{T}}\mathbf{V}_f^{-1}(\mathbf{x}_f - \hat{\mathbf{x}}_f) = \|\mathbf{x}_f - \hat{\mathbf{x}}_f\|^2_{\mathbf{V}_f^{-1}} \tag{B.18}$$

and equivalently for $\mu_f(\mathbf{x}_b)$ and $q(\mathbf{x}_b)$. To find the message $\mu_{\mathbf{a}}(\mathbf{a})$ the sum-product update rule (2.6) is applied:

$$\begin{aligned}
\mu_{\mathbf{a}}(\mathbf{a}) &= \iint f(\mathbf{x}_f, \mathbf{x}_b, \mathbf{a})\mu_f(\mathbf{x}_f)\mu_b(\mathbf{x}_b)d\mathbf{x}_f d\mathbf{x}_b \\
&\propto \iint \delta(\mathbf{x}_b - \mathbf{A}\mathbf{x}_f)\exp(-q_f(\mathbf{x}_f) - q_b(\mathbf{x}_b))d\mathbf{x}_f d\mathbf{x}_b \\
&= \int \exp(-q_f(\mathbf{x}_f) - q_b(\mathbf{A}\mathbf{x}_f))d\mathbf{x}_f \\
&= \int \exp(-q(\mathbf{x}_f, \mathbf{a}))d\mathbf{x}_f
\end{aligned} \tag{B.19}$$

with

$$q(\mathbf{x}_f, \mathbf{a}) = \|\mathbf{x}_f - \hat{\mathbf{x}}_f\|^2_{\mathbf{V}_f^{-1}} + \|\mathbf{A}\mathbf{x}_f - \hat{\mathbf{x}}_b\|^2_{\mathbf{V}_b^{-1}} \tag{B.20}$$

To integrate (B.19) we apply the trick to replace the integral by maximisation (Appendix A.3):

$$\begin{aligned}\mu_{\mathbf{a}}(\mathbf{a}) &= \int \exp(-q(\mathbf{x}_f, \mathbf{a}))d\mathbf{x}_f \\ &= \exp(-\min_{\mathbf{x}_f} q(\mathbf{x}_f, \mathbf{a})) \\ &= \exp(-q(\mathbf{a}))\end{aligned} \tag{B.21}$$

Find the minimum of $q(\mathbf{x}_f, \mathbf{a})$ by setting the derivative to zero:

$$\nabla_{\mathbf{x}_f} q(\mathbf{x}_f, \mathbf{a}) = \mathbf{V}_f^{-1}(\mathbf{x}_f - \hat{\mathbf{x}}_f) + \mathbf{A}^T\mathbf{V}_b^{-1}(\mathbf{A}\mathbf{x}_f - \hat{\mathbf{x}}_b) \stackrel{!}{=} 0 \tag{B.22}$$

$$(\mathbf{V}_f^{-1} + \mathbf{A}^T\mathbf{V}_b^{-1}\mathbf{A})\mathbf{x}_f = \mathbf{V}_f^{-1}\hat{\mathbf{x}}_f + \mathbf{A}^T\mathbf{V}_b^{-1}\hat{\mathbf{x}}_b \tag{B.23}$$

Inverting the term in parenthesis using the matrix inversion lemma (Appendix A.2) leads to:

$$(\mathbf{V}_f^{-1} + \mathbf{A}^T\mathbf{V}_b^{-1}\mathbf{A})^{-1} = \mathbf{V}_f - \mathbf{V}_f\mathbf{A}^T(\mathbf{V}_b + \mathbf{A}\mathbf{V}_f\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{V}_f \tag{B.24}$$

Finally, the minimum $\mathbf{x}_f^* = \text{argmin}_{\mathbf{x}_f}\, q(\mathbf{x}_f, \mathbf{a})$ is:

$$\mathbf{x}_f^* = \hat{\mathbf{x}}_f + \mathbf{V}_f\mathbf{A}^T(\mathbf{V}_b + \mathbf{A}\mathbf{V}_f\mathbf{A}^T)^{-1}(\hat{\mathbf{x}}_b - \mathbf{A}\hat{\mathbf{x}}_f) \tag{B.25}$$

Inserted back into the quadratic forms of (B.20):

$$\|\mathbf{x}_f^* - \hat{\mathbf{x}}_f\|^2_{\mathbf{V}_f^{-1}} = \|\hat{\mathbf{x}}_b - \mathbf{A}\hat{\mathbf{x}}_f\|^2_{\mathbf{U}^T\mathbf{A}\mathbf{V}_f\mathbf{A}^T\mathbf{U}} \tag{B.26}$$

$$\|\hat{\mathbf{x}}_b - \mathbf{A}\mathbf{x}_f^*\|^2_{\mathbf{V}_b^{-1}} = \|\hat{\mathbf{x}}_b - \mathbf{A}\hat{\mathbf{x}}_f\|^2_{\mathbf{U}^T\mathbf{V}_b\mathbf{U}} \tag{B.27}$$

with

$$\mathbf{U} = (\mathbf{V}_b + \mathbf{A}\mathbf{V}_f\mathbf{A}^T)^{-1} \tag{B.28}$$

Adding (B.26) and (B.27) leads to

$$q(\mathbf{a}) = \|\hat{\mathbf{x}}_b - \mathbf{A}\hat{\mathbf{x}}_f\|^2_{(\mathbf{V}_b + \mathbf{A}\mathbf{V}_f\mathbf{A}^T)^{-1}} \tag{B.29}$$

Unfortunately, (B.29) cannot be expressed in the form $\|\mathbf{a} - \hat{\mathbf{a}}\|^2_{\mathbf{W}_a}$, that means $\mu_{\mathbf{a}}(\mathbf{a})$ is not Gaussian in general.

To derive $\mu_{\mathbf{a}}(\mathbf{a})$ in a Gaussian form, we make the following assumptions: First, the variance $\mathbf{V}_f$ of the message $\mu_f(\mathbf{x}_f)$ is set to zero, that

means, the uncertainty in the forward estimate is ignored. Second, because the vector $\mathbf{x}_b$ is a shifted version of $\mathbf{x}_f$ only the first element of $\mathbf{x}_b$ is still an independent variable.

It is noteworthy, that for some cases (e.g. Section 6.1.1) this assumption is consistent with the real model. In most other cases, this assumption leads to an approximation to the true sum-product message, but is still reasonable, because the variance of the state estimate tends to small values.

Ignoring the variance of an incoming message and setting the variable to its current estimate is exactly what is done for every variable in the iterative conditional modes (ICM) algorithm [43]. Here it is only selectively applied where the original sum-product rule would yield unmanageable messages.

The node function for the node in Fig. B.3(b) is therefore

$$f(\mathbf{x}_f, x_b, \mathbf{a}) = \delta(x_b - \mathbf{a}^\mathsf{T}\mathbf{x}_f) \tag{B.30}$$

and the backward message is

$$\mu_b(x_b) = \mathcal{N}\big(x_b \mid \hat{x}_b, \sigma_b^2\big) \propto \exp(-q_b(x_b)) \tag{B.31}$$

Applying the node update equation (2.6) for this situation:

$$\begin{aligned}\mu(\mathbf{a}) &\propto \iint \delta(x_b - \mathbf{a}^\mathsf{T}\mathbf{x}_f) \exp(-q_f(\mathbf{x}_f) - q_b(x_b)) d\mathbf{x}_f dx_b \\ &= \int \exp\left(-q_f(\mathbf{x}_f) - q_b(\mathbf{a}^T\mathbf{x}_f)\right) d\mathbf{x}_f \end{aligned} \tag{B.32}$$

Again, replace the integral in (B.32) by a minimisation operation:

$$\mu(\mathbf{a}) = \exp\left(-\min_{\mathbf{x}_f}\left(q_f(\mathbf{x}_f) + q_b(\mathbf{a}^\mathsf{T}\mathbf{x}_f)\right)\right) \tag{B.33}$$

with

$$q_f(\mathbf{x}_f) = \|\hat{\mathbf{x}}_f - \mathbf{x}_f\|^2_{\mathbf{V}^{-1}} \tag{B.34}$$

$$q_b(\mathbf{a}^T\mathbf{x}_f) = \frac{(\hat{x}_b - \mathbf{a}^T\mathbf{x}_f)^2}{\sigma_b^2} \tag{B.35}$$

In order to find the minimum of this quadratic form, we set the gradient to zero:

$$\frac{1}{2}\nabla_{\mathbf{x}_f} q(\mathbf{x}_f, \mathbf{a}) = (\hat{x}_b - \mathbf{a}^T\mathbf{x}_f)\frac{\mathbf{a}}{\sigma_b^2} + \mathbf{V}^{-1}(\hat{\mathbf{x}}_f - \mathbf{x}_f) \stackrel{!}{=} 0 \tag{B.36}$$

$$\left(\mathbf{V}^{-1} + \frac{\mathbf{a}\mathbf{a}^T}{\sigma_b^2}\right)\mathbf{x}_f^* = \frac{\hat{x}_b}{\sigma_b^2}\mathbf{a} + \mathbf{V}^{-1}\hat{\mathbf{x}}_f \tag{B.37}$$

The expression in brackets is inverted using the matrix inversion lemma:

$$\left(\mathbf{V}^{-1} + \frac{\mathbf{a}\mathbf{a}^T}{\sigma_b^2}\right)^{-1} = \mathbf{V} - \frac{\mathbf{V}\mathbf{a}\mathbf{a}^T\mathbf{V}}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}} \tag{B.38}$$

Thus,

$$\mathbf{x}_f^* = \left(\mathbf{V} - \frac{\mathbf{V}\mathbf{a}\mathbf{a}^T\mathbf{V}}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}}\right)\frac{\hat{x}_b}{\sigma_b^2}\mathbf{a} + \left(\mathbf{V} - \frac{\mathbf{V}\mathbf{a}\mathbf{a}^T}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}}\right)\hat{\mathbf{x}}_f \tag{B.39}$$

$$= \frac{\mathbf{V}\mathbf{a}}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}}(\hat{x}_b - \mathbf{a}^T\hat{\mathbf{x}}_f) + \hat{\mathbf{x}}_f \tag{B.40}$$

which minimises (B.33). Put back into (B.34) and (B.35):

$$\hat{x}_b - \mathbf{a}^T\mathbf{x}_f^* = \left(1 - \frac{\mathbf{a}^T\mathbf{V}\mathbf{a}}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}}\right)\hat{x}_b \tag{B.41}$$

$$+ \frac{\mathbf{a}^T\mathbf{V}\mathbf{a}\mathbf{a}^T}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}}\hat{\mathbf{x}}_f - \mathbf{a}^T\hat{\mathbf{x}}_f \tag{B.42}$$

$$= \frac{\sigma_b^2}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}}\left(\hat{x}_b - \mathbf{a}^T\hat{\mathbf{x}}_f\right) \tag{B.43}$$

$$\hat{\mathbf{x}}_f - \mathbf{x}_f^* = -\frac{\mathbf{V}\mathbf{a}}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}}\left(\hat{x}_b - \mathbf{a}^T\hat{\mathbf{x}}_f\right) \tag{B.44}$$

$$\|\hat{\mathbf{x}}_f - \mathbf{x}_f^*\|^2_{\mathbf{V}_f^{-1}} = \left(\hat{x}_b - \mathbf{a}^T\hat{\mathbf{x}}_f\right)^2 \frac{\mathbf{a}^T\mathbf{V}\mathbf{a}}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}} \tag{B.45}$$

And collecting everything we obtain:

$$q(\mathbf{a}) = \frac{\left(\hat{x}_b - \mathbf{a}^T\hat{\mathbf{x}}_f\right)^2}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}} \tag{B.46}$$

$$= \left(\mathbf{a} - \frac{\hat{x}_b}{\|\hat{\mathbf{x}}_f\|^2}\hat{\mathbf{x}}_f\right)^T \frac{\hat{\mathbf{x}}_f\hat{\mathbf{x}}_f^T}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}}\left(\mathbf{a} - \frac{\hat{x}_b}{\|\hat{\mathbf{x}}_f\|^2}\hat{\mathbf{x}}_f\right) \tag{B.47}$$

$$= \|\mathbf{a} - \hat{\mathbf{a}}\|^2_{\mathbf{W}_a} \tag{B.48}$$

with

$$\hat{\mathbf{a}} = \frac{\hat{x}_b}{\|\hat{\mathbf{x}}_f\|^2}\hat{\mathbf{x}}_f \qquad \mathbf{W}_a = \frac{\hat{\mathbf{x}}_f\hat{\mathbf{x}}_f^T}{\sigma_b^2 + \mathbf{a}^T\mathbf{V}\mathbf{a}} = \frac{\hat{\mathbf{x}}_f\hat{\mathbf{x}}_f^T}{\sigma_b^2} \tag{B.49}$$

The last equality holds because of our original assumption $\mathbf{V} = \mathbf{0}$. This proofs the equations in the first row of Table 5.1.

B.4 Coefficient Estimation, Gradient

In this section, the gradient based update formulas for the state transition node in the state-space model (the last line in Table 5.3) are derived. Fig. B.4 shows the factor graph node.

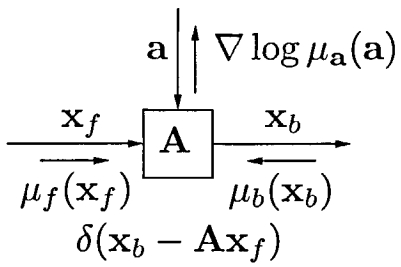


Figure B.4: Node A for coefficient estimation in higher-order auto-regression.

The node function is $f(\mathbf{x}_f, \mathbf{x}_b, \mathbf{a}) = \delta(\mathbf{x}_b - \mathbf{A}\mathbf{x}_f)$ where

$$\mathbf{A} \triangleq \begin{bmatrix} \mathbf{a}^{\mathsf{T}} \\ \mathbf{I} \quad \mathbf{0} \end{bmatrix} \tag{B.50}$$

and the incoming message $\mu_f(\mathbf{x}_f)$ is

$$\mu_f(\mathbf{x}_f) = \mathcal{N}(\mathbf{x}_f \mid \hat{\mathbf{x}}_f, \mathbf{V}_f) \propto \exp(-q_f(\mathbf{x}_f)) \tag{B.51}$$

with

$$q_f(\mathbf{x}_f) = (\mathbf{x}_f - \hat{\mathbf{x}}_f)^{\mathsf{T}} \mathbf{V}_f^{-1} (\mathbf{x}_f - \hat{\mathbf{x}}_f) = \|\mathbf{x}_f - \hat{\mathbf{x}}_f\|^2_{\mathbf{V}_f^{-1}} \tag{B.52}$$

and equivalently for $\mu_f(\mathbf{x}_b)$ and $q(\mathbf{x}_b)$. To find the message $\nabla \log \mu_{\mathbf{a}}(\mathbf{a})$ the gradient note update rule (5.32) is applied (cf. (B.19)):

$$\begin{aligned} \mu_{\mathbf{a}}(\mathbf{a}) &= \iint f(\mathbf{x}_f, \mathbf{x}_b, \mathbf{a}) \mu_f(\mathbf{x}_f) \mu_b(\mathbf{x}_b) d\mathbf{x}_f d\mathbf{x}_b \\ &\propto \exp(-\min_{\mathbf{x}_f} q(\mathbf{x}_f, \mathbf{a})) \\ &= \exp(-q(\mathbf{x}_f^*, \mathbf{a})) \end{aligned} \tag{B.53}$$

with

$$q(\mathbf{x}_f, \mathbf{a}) = \|\mathbf{x}_f - \hat{\mathbf{x}}_f\|^2_{\mathbf{V}_f^{-1}} + \|\mathbf{A}\mathbf{x}_f - \hat{\mathbf{x}}_b\|^2_{\mathbf{V}_b^{-1}} \tag{B.54}$$

and $\mathbf{x}_f^* = \text{argmin}_{\mathbf{x}_f} q(\mathbf{x}_f, \mathbf{a})$. The log-derivative is then

$$\begin{aligned}\nabla_{\mathbf{a}} \log \mu_{\mathbf{a}}(\mathbf{a}) &\propto -\nabla_{\mathbf{a}} q(\mathbf{x}_f^*, \mathbf{a}) \\ &= -\nabla_{\mathbf{a}} \left[\|\mathbf{x}_f^* - \hat{\mathbf{x}}_f\|^2_{\mathbf{V}_f^{-1}} + \|\mathbf{A}\mathbf{x}_f^* - \hat{\mathbf{x}}_b\|^2_{\mathbf{V}_b^{-1}} \right] \\ &= J_{\mathbf{a}}^{\mathsf{T}}(\mathbf{x}_f^*)\mathbf{V}_f^{-1}(\hat{\mathbf{x}}_f - \mathbf{x}_f^*) + \\ &\quad J_{\mathbf{a}}^{\mathsf{T}}(\hat{\mathbf{x}}_b - \mathbf{A}\mathbf{x}_f^*)\mathbf{V}_b^{-1}(\hat{\mathbf{x}}_b - \mathbf{A}\mathbf{x}_f^*) \end{aligned} \tag{B.55}$$

where $J_{\mathbf{a}}(.)$ denotes the Jacobian w.r.t. $\mathbf{a}$. Obviously, the gradient takes a complicated form. The special structure of $\mathbf{A}$ allows some simplifications, nonetheless simpler forms are appreciated. A first approximation is based on the representation (cf. (B.29))

$$q(\mathbf{a}) = \|\hat{\mathbf{x}}_b - \mathbf{A}\hat{\mathbf{x}}_f\|^2_{(\mathbf{V}_b + \mathbf{A}\mathbf{V}_f\mathbf{A}^{\mathsf{T}})^{-1}} \tag{B.56}$$

where for computing the gradient the dependency of $(\mathbf{V}_b + \mathbf{A}\mathbf{V}_f\mathbf{A}^{\mathsf{T}})^{-1}$ on $\mathbf{a}$ is ignored:

$$\nabla_{\mathbf{a}} q(\mathbf{a}) \approx -2\hat{\mathbf{x}}_f \mathbf{c}^{\mathsf{T}}(\mathbf{V}_b + \mathbf{A}\mathbf{V}_f\mathbf{A}^{\mathsf{T}})^{-1}(\hat{\mathbf{x}}_b - \mathbf{A}\hat{\mathbf{x}}_f) \tag{B.57}$$

where $\mathbf{c}^{\mathsf{T}} \triangleq (1, 0, \ldots, 0)^{\mathsf{T}}$. A second alternative is to ignore the variance completely (i.e. $q(\mathbf{a}) \approx \|\hat{\mathbf{x}}_b - \mathbf{A}\hat{\mathbf{x}}_f\|^2$) which leads to

$$\nabla_{\mathbf{a}} q(\mathbf{a}) \approx -2\hat{\mathbf{x}}_f(\hat{\mathbf{x}}_b^{[1]} - \mathbf{a}^{\mathsf{T}}\hat{\mathbf{x}}_f) \tag{B.58}$$

This last form is very simple and simulations have shown, that all three gradient approximations (B.55), (B.57) and (B.58) lead to algorithms with equal performance.

Seite Leer /
Blank leaf

Appendix C

About the Symmetry Problem in Forward-only Joint Variance Estimation

This section shows in detail that the message for estimating the input noise and the message for estimating the observation noise are algebraically the same in forward-only message passing. For both cases all messages are explicitely written out. The message numbers correspond to the numbers in Fig. C.1.

Observation noise variance:

① $\mu_1(\mathbf{x}_{n-1}) = \mathcal{N}\Big(\mathbf{x}_{n-1} \;\Big|\; \hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}, \mathbf{V}_{n-1|\overrightarrow{n-1}}\Big)$

② (Table 4.1-3)

$$\mu_2(\mathbf{x}_n^\circ) = \mathcal{N}\Big(\mathbf{x}_n^\circ \mid \hat{\mathbf{x}}_{n|\overrightarrow{n-1}}, \mathbf{V}_{n|\overrightarrow{n-1}}\Big)$$
$$\hat{\mathbf{x}}_{n|\overrightarrow{n-1}} = \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}$$
$$\mathbf{V}_{n|\overrightarrow{n-1}} = \mathbf{A}\mathbf{V}_{n-1|\overrightarrow{n-1}}\mathbf{A}^\mathsf{T}$$

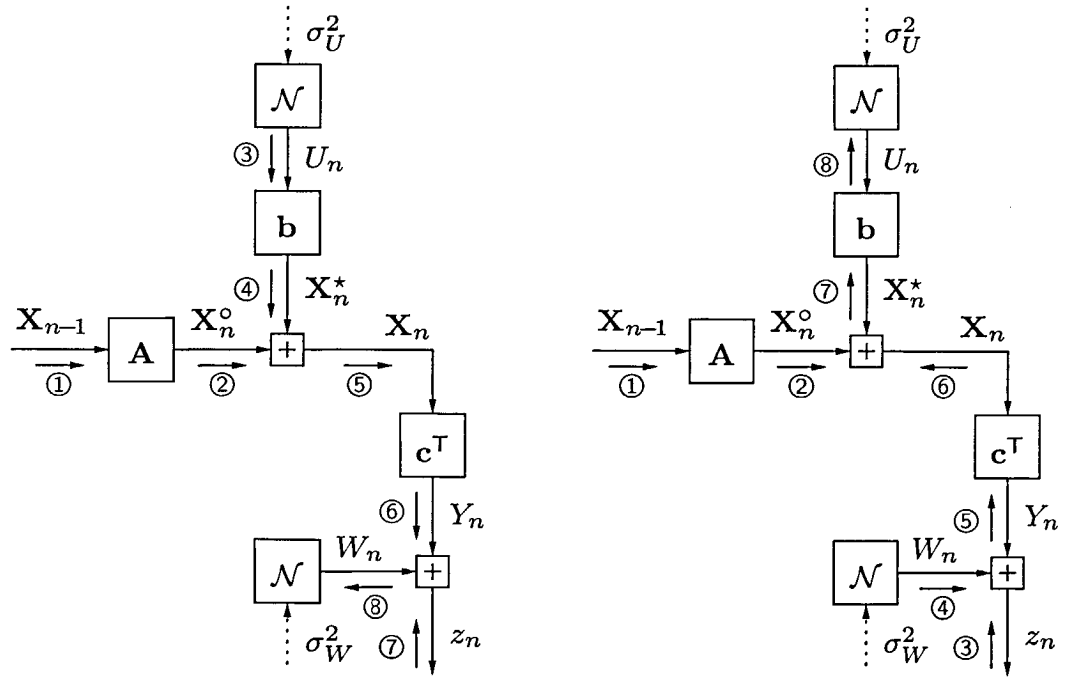


Figure C.1: One section of the factor graph for joint variance estimation. The messages for the estimation of the observation noise variance are shown in the left-hand graph, whereas the messages for the estimation of the input noise variance are shown in the right-hand graph.

③ (Table 4.1-7) $\mu_3(u_n) = \mathcal{N}\big(u_n \mid 0, \sigma_U^2\big)$

④ (Table 4.1-3) $\mu_4(\mathbf{x}_n^\star) = \mathcal{N}\big(\mathbf{x}_n^\star \mid \mathbf{0}, \mathbf{b}\mathbf{b}^\mathsf{T}\sigma_U^2\big)$

⑤ (Table 4.1-2) $\mu_5(\mathbf{x}_n) = \mathcal{N}\Big(\mathbf{x}_n \mid \hat{\mathbf{x}}_{n|\overrightarrow{n-1}}, \mathbf{V}_{n|\overrightarrow{n-1}} + \mathbf{b}\mathbf{b}^\mathsf{T}\sigma_U^2\Big)$

⑥ (Table 4.1-3)

$$\begin{aligned}\mu_6(y_n) &= \mathcal{N}(y_n \mid \hat{y}_n, v_6)\\ \hat{y}_n &= \mathbf{c}^\mathsf{T}\hat{\mathbf{x}}_{n|\overrightarrow{n-1}} = \mathbf{a}^\mathsf{T}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}\\ v_6 &= \sigma_U^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T}\end{aligned}$$

⑦ $\mu_7(z) = \delta(z - z_n)$

⑧ (Table 4.1-2)

$$\mu_8(w_n) = \mathcal{N}\Big(w_n \;\Big|\; z_n - \mathbf{a}^\mathsf{T}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}, \sigma_U^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T}\Big)$$

Input noise variance:

① $\mu_1(\mathbf{x}_{n-1}) = \mathcal{N}\left(\mathbf{x}_{n-1} \,\middle|\, \hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}, \mathbf{V}_{n-1|\overrightarrow{n-1}}\right)$

② (Table 4.1-3)

$$\begin{aligned}\mu_2(\mathbf{x}_n^\circ) &= \mathcal{N}\left(\mathbf{x}_n^\circ \mid \hat{\mathbf{x}}_{n|\overrightarrow{n-1}}, \mathbf{V}_{n|\overrightarrow{n-1}}\right)\\ \hat{\mathbf{x}}_{n|\overrightarrow{n-1}} &= \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}\\ \mathbf{V}_{n|\overrightarrow{n-1}} &= \mathbf{A}\mathbf{V}_{n-1|\overrightarrow{n-1}}\mathbf{A}^\mathsf{T}\end{aligned}$$

③ $\mu_3(z) = \delta(z - z_n)$

④ (Table 4.1-7) $\mu_4(w_n) = \mathcal{N}\left(w_n \mid 0, \sigma_W^2\right)$

⑤ (Table 4.1-2) $\mu_5(y_n) = \mathcal{N}\left(y_n \mid z_n, \sigma_W^2\right)$

⑥ (Table 4.1-4)

$$\begin{aligned}\mu_6(\mathbf{x}_n) &= \mathcal{N}_W(\mathbf{x}_n \mid \mathbf{m}_6, \mathbf{W}_6)\\ \mathbf{m}_6 &= \left(\frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2}\right)^{\#} \frac{\mathbf{c}}{\sigma_W^2} z_n = (\mathbf{c}\mathbf{c}^\mathsf{T})^{\#}\mathbf{c}z_n = \mathbf{c}z_n\\ \mathbf{W}_6 &= \frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2}\end{aligned}$$

⑦ (Table 4.1-2)

$$\begin{aligned}\mu_7(\mathbf{x}_n^\star) &= \mathcal{N}_W(\mathbf{x}_n^\star \mid \mathbf{m}_7, \mathbf{W}_7)\\ \mathbf{m}_7 &= \mathbf{c}z_n - \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}\\ \mathbf{W}_7 &= \mathbf{W}_{n|\overrightarrow{n-1}}(\mathbf{W}_{n|\overrightarrow{n-1}} + \mathbf{W}_6)^{\#}\mathbf{W}_6\\ &= \mathbf{V}_{n|\overrightarrow{n-1}}^{-1}\left(\mathbf{V}_{n|\overrightarrow{n-1}}^{-1} + \frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2}\right)^{\#}\frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2}\\ &= \mathbf{V}_{n|\overrightarrow{n-1}}^{-1}\left(\mathbf{V}_{n|\overrightarrow{n-1}} - \frac{\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}\mathbf{c}^\mathsf{T}\mathbf{V}_{n|\overrightarrow{n-1}}}{\sigma_W^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T}}\right)\frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2}\\ &= \frac{\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T}}\end{aligned}$$

⑧ (Table 4.1-4)

$$\begin{aligned}
\mu_8(u_n) &= \mathcal{N}(u_n \mid \hat{u}_n, v_8) \\
\hat{u}_n &= (\mathbf{b}^\mathsf{T}\mathbf{W}_7\mathbf{b})^{\#}\mathbf{b}^\mathsf{T}\mathbf{W}_7\mathbf{m}_7 \\
&= (\sigma_W^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T})\frac{\mathbf{b}^\mathsf{T}\mathbf{c}\mathbf{c}^\mathsf{T}}{\sigma_W^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T}}(\mathbf{c}z_n - \mathbf{A}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}) \\
&= z_n - \mathbf{a}^\mathsf{T}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}} \\
v_8 &= (\mathbf{b}^\mathsf{T}\mathbf{W}_7\mathbf{b})^{-1} \\
&= \sigma_W^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T}
\end{aligned}$$

The messages arriving at the node for the input noise and the node for the observation noise are almost the same:

$$\begin{aligned}
&\mathcal{N}\left(u_n \;\middle|\; z_n - \mathbf{a}^\mathsf{T}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}, \sigma_W^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T}\right) \\
&\mathcal{N}\left(u_n \;\middle|\; z_n - \mathbf{a}^\mathsf{T}\hat{\mathbf{x}}_{n-1|\overrightarrow{n-1}}, \sigma_U^2 + \mathbf{c}\mathbf{V}_{n|\overrightarrow{n-1}}\mathbf{c}^\mathsf{T}\right).
\end{aligned}$$

They differ only in the variance. When starting with the same initial value for σ_U^2 and σ_W^2, both messages are the same. Therefore, the update for the new estimate of σ_U^2 and σ_W^2 will be the same. This symmetry can be broken when using different initial messages for the variances.

Appendix D

Classic Estimators for Variance Estimation

In this section properties of different estimators for variance estimation are compared. Consider the following estimation problem: Let $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_n)^\mathsf{T}$ be white Gaussian noise with probability density function

$$p(\mathbf{y}|\sigma^2) = \prod_{\ell=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{y_\ell^2}{2\sigma^2}\right) \tag{D.1}$$

where σ^2 is the unknown variance of the Gaussian noise. Based on the observations $(Y_1, \ldots, Y_n) = (y_1, \ldots, y_n)$ we wish to estimate σ^2.

A Note about the Mean Squared Error
MSE in the classic sense is the average over different observations for a fixed parameter value:

$$\mathrm{MSE} = \mathsf{E}_{p(\mathbf{y}|\theta)}\left[(\theta - \hat{\theta}(\mathbf{Y}))^2\right]. \tag{D.2}$$

On the contrary, the quadratic cost function for the Bayesian MMSE estimator is an average over different parameters for fixed observations:

$$\mathrm{BMSE} = \mathsf{E}_{p(\theta|\mathbf{y})}\left[(\Theta - \hat{\theta}(\mathbf{y}))^2\right] \tag{D.3}$$

Maximum Likelihood (ML) Estimator

The maximum likelihood estimator solves the following maximisation problem:

$$\hat{\sigma}^2_{\mathrm{ML}} = \underset{\sigma^2}{\operatorname{argmax}}\, p(\mathbf{y}|\sigma^2) \tag{D.4}$$

Solving

$$\frac{\partial}{\partial \sigma^2} \prod_{\ell=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{y_\ell^2}{2\sigma^2}\right) \stackrel{!}{=} 0$$

$$\frac{1}{(2\pi)^{\frac{n}{2}}} \frac{\partial}{\partial \sigma^2} (\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{\sum y_\ell^2}{2\sigma^2}\right) \stackrel{!}{=} 0$$

$$\frac{1}{(2\pi)^{\frac{n}{2}}} \left[-\frac{n}{2}(\sigma^2)^{-\frac{n}{2}-1} \exp\left(-\frac{\sum y_\ell^2}{2\sigma^2}\right) + (\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{\sum y_\ell^2}{2\sigma^2}\right) \frac{\sum y_\ell^2}{2\sigma^2} (\sigma^2)^{-1} \right] \stackrel{!}{=} 0$$

$$-\frac{n}{2} + \frac{\sum y_\ell^2}{2} \sigma^{-2} \stackrel{!}{=} 0$$

leads to

ML estimator:

$$\hat{\sigma}^2_{\mathrm{ML}} = \frac{1}{n} \sum_{\ell=1}^{n} y_\ell^2 \tag{D.5}$$

The bias of this estimator is

$$\begin{aligned} b(\hat{\sigma}^2_{\mathrm{ML}}) &= \mathsf{E}[\hat{\sigma}^2_{\mathrm{ML}}] - \sigma^2 \\ &= \frac{1}{n} \sum_{\ell=1}^{n} \mathsf{E}[y_\ell^2] - \sigma^2 \\ &= \frac{n}{n}\sigma^2 - \sigma^2 = 0. \end{aligned}$$

Therefore the ML estimator is unbiased. The mean squared error (MSE)

is

$$\begin{aligned}\mathrm{MSE}(\hat{\sigma}^2_{\mathrm{ML}}) &= \mathsf{E}\left[\left(\hat{\sigma}^2_{\mathrm{ML}} - \sigma^2\right)^2\right] \\ &= \mathsf{E}\left[(\hat{\sigma}^2_{\mathrm{ML}})^2\right] - 2\sigma^2\mathsf{E}[\hat{\sigma}^2_{\mathrm{ML}}] + \sigma^4 \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathsf{E}[y_i^2 y_j^2] - \frac{2\sigma^2}{n}\sum_{\ell=1}^{n}\mathsf{E}[y_\ell^2] + \sigma^4 \\ &= \frac{n^2+2n}{n^2}\sigma^4 - 2\sigma^4 + \sigma^4 = \frac{2}{n}\sigma^4\end{aligned}$$

Minimum MSE biased estimator

In [125] it is shown that a biased estimator can have smaller MSE. The resulting estimator is

Minimum MSE biased estimator:

$$\hat{\sigma}^2_{\mathrm{b}} = \frac{1}{n+2}\sum_{\ell=1}^{n} y_\ell^2 \tag{D.6}$$

with mean squared error

$$\mathrm{MSE}(\hat{\sigma}^2_{\mathrm{b}}) = \frac{2\sigma^4}{n+2}$$

Maximum A Posteriori (MAP) Estimator

Here we assume the unknown variance σ^2 is a random variable whose particular realisation we have to estimate. The maximum a posteriori estimator is defined as follows:

$$\hat{\sigma}^2_{\mathrm{MAP}} = \underset{\sigma^2}{\operatorname{argmax}}\, p(\sigma^2|\mathbf{y}) \tag{D.7}$$

To derive an estimator based on (D.7) we are required to assume a prior probability density function $p(\sigma^2)$ for σ^2. Hence we can write (D.7) as

$$\begin{aligned}\hat{\sigma}^2_{\mathrm{MAP}} &= \underset{\sigma^2}{\operatorname{argmax}}\, p(\mathbf{y}|\sigma^2)p(\sigma^2) \\ &= \underset{\sigma^2}{\operatorname{argmax}}\left(\log p(\mathbf{y}|\sigma^2) + \log p(\sigma^2)\right)\end{aligned}$$

Choosing a prior $p(\sigma^2)$ which is the conjugate prior of $p(\mathbf{y}|\sigma^2)$ leads to an inverted gamma density (cf. Appendix A.1.2):

$$p(\sigma^2) = \mathrm{Ig}\left(\sigma^2 \mid \alpha, \beta\right)$$

The MAP estimator for the variance estimation problem is therefore

$$\begin{aligned}\hat{\sigma}^2_{\mathrm{MAP}} &= \operatorname*{argmax}_{\sigma^2}\left[-\frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum y_\ell^2 + (-\alpha-1)\log\sigma^2 - \frac{\beta}{\sigma^2}\right] \\ &= \operatorname*{argmax}_{\sigma^2}\left[-\left(\frac{n}{2}+\alpha+1\right)\log\sigma^2 - \frac{1}{\sigma^2}\left(\frac{1}{2}\sum y_\ell^2 + \beta\right)\right]\end{aligned}$$

MAP estimator:

$$\hat{\sigma}^2_{\mathrm{MAP}} = \frac{\frac{1}{2}\sum_{\ell=1}^{n} y_\ell^2 + \beta}{\frac{n}{2}+\alpha+1} \tag{D.8}$$

The bias of the MAP estimator is

$$\begin{aligned}b(\hat{\sigma}^2_{\mathrm{MAP}}) &= \mathsf{E}[\hat{\sigma}^2_{\mathrm{MAP}}] - \sigma^2 \\ &= \frac{\frac{n}{2}\sigma^2 + \beta}{\frac{n}{2}+\alpha+1} - \sigma^2\end{aligned}$$

and its mean squared error

$$\begin{aligned}\mathrm{MSE}(\hat{\sigma}^2_{\mathrm{MAP}}) &= \mathsf{E}\left[(\hat{\sigma}^2_{\mathrm{MAP}} - \sigma^2)^2\right] \\ &= \frac{n^2+2n}{n^2}\sigma^4 - \frac{n\sigma^4 + 2\sigma^2\beta}{\frac{n}{2}+\alpha+1} + \sigma^4 \\ &= \left(\frac{n^2+2n}{n^2} - \frac{n}{\frac{n}{2}+\alpha+1} + 1\right)\sigma^4 - \frac{2\beta}{\frac{n}{2}+\alpha+1}\sigma^2\end{aligned}$$

Bayes Estimator with Quadratic Cost Function

The Bayes estimator with quadratic cost function (also called minimum mean squared error estimator (MMSE); but care must be taken what kind of MSE is meant; see introduction to this section) is the conditional mean of the posterior distribution.

$$\hat{\sigma}^2_{\mathrm{MMSE}} = \mathsf{E}[\sigma^2|\mathbf{Y}=\mathbf{y}] \tag{D.9}$$

For the variance estimation problem (D.9) becomes

$$\hat{\sigma}^2_{\mathrm{MMSE}} = \int_0^\infty \sigma^2 p(\sigma^2|\mathbf{y})\,d\sigma^2 = \frac{1}{K}\int_0^\infty \sigma^2 p(\mathbf{y}|\sigma^2)p(\sigma^2)\,d\sigma^2$$

where

$$K = \int_0^\infty p(\mathbf{y}|\sigma^2)p(\sigma^2)d\sigma^2$$

Choosing the conjugate prior for σ^2

$$p(\sigma^2) = \mathrm{Ig}\left(\sigma^2 \mid \alpha, \beta\right)$$

the MMSE estimator becomes

MMSE estimator:

$$\hat{\sigma}^2_{\mathrm{MMSE}} = \frac{\frac{1}{2}\sum_{\ell=1}^{n} y_\ell^2 + \beta}{\frac{n}{2} + \alpha - 1} \tag{D.10}$$

The bias of the MMSE estimator is

$$\begin{aligned} b(\hat{\sigma}^2_{\mathrm{MMSE}}) &= \mathsf{E}[\hat{\sigma}^2_{\mathrm{MMSE}}] - \sigma^2 \\ &= \frac{\frac{n}{2}\sigma^2 + \beta}{\frac{n}{2} + \alpha - 1} - \sigma^2 \end{aligned}$$

and its mean squared error

$$\begin{aligned} \mathrm{MSE}(\hat{\sigma}^2_{\mathrm{MMSE}}) &= \mathsf{E}\left[(\hat{\sigma}^2_{\mathrm{MMSE}} - \sigma^2)^2\right] \\ &= \frac{n^2 + 2n}{n^2}\sigma^4 - \frac{n\sigma^4 + 2\sigma^2\beta}{\frac{n}{2} + \alpha + 1} + \sigma^4 \\ &= \left(\frac{n^2 + 2n}{(n-4)^2} - \frac{n}{\frac{n}{2} + \alpha - 1} + 1\right)\sigma^4 - \frac{2\beta}{\frac{n}{2} + \alpha - 1}\sigma^2 \end{aligned}$$

Bayes Estimator without Prior

We can use a Bayes estimator without a prior if we make the following definition:

$$p(\mathbf{y}, \theta) \overset{\Delta}{\propto} p(\mathbf{y}|\theta) \tag{D.11}$$

for fixed $\mathbf{y}$. For $p(\mathbf{y}|\theta)$ to be a valid distribution in θ it has to be integrable. Definition (D.11) also implies

$$p(\theta|\mathbf{y}) \propto p(\mathbf{y}|\theta) \tag{D.12}$$

Therefore, $p(\mathbf{y}|\theta)$ can be used to derive a Bayes estimator as long as the proportionality constant can be reconstructed, i.e. $p(\mathbf{y}|\theta)$ is integrable w.r.t. θ.

For the variance estimation problem the Bayes MMSE estimator without prior is

$$\hat{\sigma}^2_{\mathrm{QB}} = \frac{1}{K} \int_0^\infty \sigma^2 p(\mathbf{y}|\sigma^2)\, d\sigma^2 \tag{D.13}$$

with

$$\begin{aligned} K &= \int_0^\infty p(\mathbf{y}|\sigma^2)\, d\sigma^2 \\ &= \frac{1}{(2\pi)^{n/2}} \int_0^\infty (\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{\sum y_\ell}{2\sigma^2}\right) d\sigma^2 \qquad n > 2 \end{aligned} \tag{D.14}$$

The estimator is therefore

$$\hat{\sigma}^2_{\mathrm{QB}} = \frac{1}{n-4} \sum_{\ell=1}^{n} y_\ell^2 \qquad n > 4 \tag{D.15}$$

For less than five observations the estimator (D.15) does not exist. In addition, the posterior density (D.14) is not integrable for less than three observations, which indicates that any estimator with fewer observations is meaningless since it is impossible to determine the mass of the posterior density.

Appendix E

Derivation of the EM Update Rules

E.1 Mean Estimation

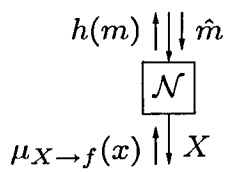


Figure E.1: Factor graph node for a Gaussian distribution with unknown mean.

We consider the situation depicted in Fig. E.1. The function of the node is

$$f(x, m) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x-m)^2}{2s}\right). \tag{E.1}$$

where $m \in \mathbb{R}$ is the unknown mean and $s > 0$ the known variance of the Gaussian distribution. The h-message is computed as follows

$$h(m) = \int_{\mathcal{D}_x} p(x|\hat{m}^{(k)}) \log f(x, m) dx \tag{E.2}$$

$$= \int_{\mathcal{D}_x} p(x|\hat{m}^{(k)}) \left(-\frac{1}{2}\log(2\pi s) - \frac{(x-m)^2}{2s} \right) dx \tag{E.3}$$

$$= C - \frac{1}{2s}\left(m^2 - 2m\mathsf{E}_p[X|\hat{m}^{(k)}] \right) \tag{E.4}$$

where C is a proper scaling constant. Looking at $e^{h(m)}$:

$$\boxed{e^{h(m)} \propto \mathcal{N}\Big(m \;\Big|\; \mathsf{E}_p[X|\hat{m}^{(k)}], s\Big)} \tag{E.5}$$

If $\mu_{X\to f}(x) \propto \mathcal{N}(x \mid m_X, s_X)$ is a Gaussian message, (E.5) becomes

$$e^{h(m)} \propto \mathcal{N}\left(m \;\middle|\; \frac{sm_X + s_X\hat{m}^{(k)}}{s+s_X}, s\right) \tag{E.6}$$

E.1.1 The vector case

The node function in the vector case is

$$f(\mathbf{x},\mathbf{m}) = \frac{1}{\sqrt{(2\pi)^n|\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathsf{T}}\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m}) \right). \tag{E.7}$$

The h-message is therefore

$$h(\mathbf{m}) = \int_{\mathcal{D}_\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{m}}^{(k)}) \log f(\mathbf{x},\mathbf{m}) d\mathbf{x} \tag{E.8}$$

$$= C - \frac{1}{2}\left(\mathbf{m}^{\mathsf{T}}\mathbf{V}^{-1}\mathbf{m} - 2\mathbf{m}^{\mathsf{T}}\mathbf{V}^{-1}\mathsf{E}_p[\mathbf{X}|\hat{\mathbf{m}}^{(k)}] \right) \tag{E.9}$$

Looking at $e^{h(\mathbf{m})}$:

$$\boxed{e^{h(\mathbf{m})} \propto \mathcal{N}\Big(\mathbf{m} \;\Big|\; \mathsf{E}_p[\mathbf{X}|\hat{\mathbf{m}}^{(k)}], \mathbf{V}\Big)} \tag{E.10}$$

E.2 Variance Estimation

We consider the situation as depicted in Fig. E.2. The function of the node is

$$f(x,s) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x-m)^2}{2s} \right). \tag{E.11}$$

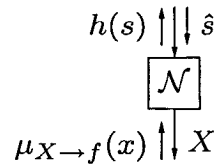


Figure E.2: Factor graph node for a Gaussian distribution with unknown variance.

where $s \in \mathbb{R}^+$ is the unknown variance and $m \in \mathbb{R}$ the known mean of the Gaussian distribution. The h-message is computed as follows

$$h(s) = \int_{\mathcal{D}_x} p(x|\hat{s}^{(k)}) \log f(x,s) dx \tag{E.12}$$

$$= \int_{\mathcal{D}_x} p(x|\hat{s}^{(k)}) \left(-\frac{1}{2}\log(2\pi s) - \frac{(x-m)^2}{2s} \right) dx \tag{E.13}$$

$$= C - \frac{1}{2}\log s - \frac{\mathsf{E}_p\left[(X-m)^2 \mid \hat{s}^{(k)}\right]}{2s} \tag{E.14}$$

Looking at $e^{h(s)}$:

$$\boxed{e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{1}{2}\mathsf{E}_p\left[(X-m)^2 \mid \hat{s}^{(k)}\right]\right)} \tag{E.15}$$

where Ig denotes an inverted gamma distribution and is closed under multiplication.

If $\mu_{X\to f}(x) \propto \mathcal{N}(x \mid m_X, s_X)$ is a Gaussian message, (E.15) becomes

$$e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{1}{2}\left(\frac{\hat{s}^{(k)}(s_X - 2mm_X + m^2) - m^2 s_X}{\hat{s}^{(k)} + s_X}\right)\right) \tag{E.16}$$

E.2.1 The vector case

The node function for the vector case is

$$f(\mathbf{x}, \mathbf{V}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathsf{T}}\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m})\right). \tag{E.17}$$

The h-message is therefore

$$h(\mathbf{V}) = \int_{\mathcal{D}_\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{V}}^{(k)}) \log f(\mathbf{x}, \mathbf{V}) d\mathbf{x} \tag{E.18}$$

$$= C - \frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}\mathsf{E}_p\left[(\mathbf{X}-\mathbf{m})^\mathsf{T}\mathbf{V}^{-1}(\mathbf{X}-\mathbf{m}) \,\middle|\, \hat{\mathbf{V}}^{(k)}\right] \tag{E.19}$$

E.2.2 Special forms of V

$\mathbf{V} = \mathbf{I}s, s \in \mathbb{R}^+$:

$$h(s) = C - \frac{n}{2}\log s - \frac{1}{2s}\mathsf{E}_p\left[(\mathbf{X}-\mathbf{m})^\mathsf{T}(\mathbf{X}-\mathbf{m})\right] \tag{E.20}$$

$$e^{h(s)} = \mathrm{Ig}\left(s \,\middle|\, \frac{n-2}{2}, \frac{1}{2}\mathsf{E}_p\left[(\mathbf{X}-\mathbf{m})^\mathsf{T}(\mathbf{X}-\mathbf{m})\right]\right) \tag{E.21}$$

$\mathbf{V} = \mathrm{diag}(\mathbf{s}), \mathbf{s} \in \mathbb{R}^{+n}$:

$$h(\mathbf{s}) = C - \frac{1}{2}\sum_{\ell=1}^{n}\log s_\ell - \sum_{\ell=1}^{n}\frac{1}{2s_\ell}\mathsf{E}_p[(X_\ell - m_\ell)^2] \tag{E.22}$$

$$\propto \prod_{\ell=1}^{n}\mathrm{Ig}\left(s_\ell \,\middle|\, -\frac{1}{2}, \frac{1}{2}\mathsf{E}_p[(X_\ell - m_\ell)^2]\right) \tag{E.23}$$

E.3 Coefficient Estimation

Here, the problem of estimating the coefficients of an autoregressive (AR) system is considered. The function $f(x_1, x_2, a)$ is defined as

$$f(x_1, x_2, a) = \delta(x_2 - a x_1) \tag{E.24}$$

with $a \in \mathbb{R}$ is the scalar AR coefficient. Computing the message $h(a)$ for this node leads to a singularity problem because of the $\log\delta(.)$. We, therefore, have to take neighbouring nodes into account to get rid of the $\delta(.)$. In the AR model there is also a driving input (or control input) as depicted in Fig. E.3. The dotted box in the graph to the left is represented by the node to the right.

The function represented by the graph in Fig. E.3 is

$$f(x_1, x_2, a) = \frac{1}{\sqrt{2\pi s}}\exp\left(-\frac{(x_2 - a x_1)^2}{2s}\right). \tag{E.25}$$

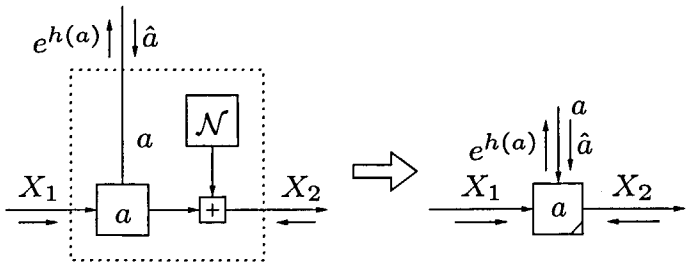


Figure E.3: Factor graph of the state transition node for a linear state-space model.

The message $h(a)$ becomes

$$h(a) = \iint_{\mathcal{D}_{x_1,x_2}} p(x_1, x_2|a^{(k)}) \log f(x_1, x_2, a) dx_1 dx_2 \tag{E.26}$$

$$= \iint_{\mathcal{D}_{x_1,x_2}} p(x_1, x_2|a^{(k)}) \left(-\frac{1}{2}\log(2\pi s) - \frac{(x_2 - a x_1)^2}{2s} \right) dx_1 dx_2 \tag{E.27}$$

$$= C - \frac{1}{2s}\left(a^2 \mathsf{E}[X_1^2|\hat{a}^{(k)}] - a2\mathsf{E}[X_1 X_2|\hat{a}^{(k)}] + \mathsf{E}[X_2^2|\hat{a}^{(k)}] \right) \tag{E.28}$$

with

$$C = -\frac{1}{2}\log(2\pi s). \tag{E.29}$$

Because (E.28) is a quadratic form, it is convenient to send the message $e^{h(a)}$ instead of $h(a)$:

$$\boxed{e^{h(a)} \propto \mathcal{N}_W\left(a \,\middle|\, \frac{\mathsf{E}[X_1 X_2|\hat{a}^{(k)}]}{\mathsf{E}[X_1^2|\hat{a}^{(k)}]}, \mathsf{E}[X_1^2|\hat{a}^{(k)}] s^{-1} \right)} \tag{E.30}$$

E.3.1 The vector case

The function $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{A})$ is defined as

$$f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{A}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)^{\mathsf{T}} \mathbf{V}^{-1} (\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1) \right) \tag{E.31}$$

with $\mathbf{A} \in \mathbb{R}^{n \times n}$. The message $h(\mathbf{A})$ becomes

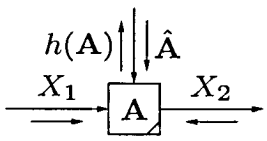


Figure E.4: Factor graph of the state transition node for the vector case.

$$h(\mathbf{A}) = \iint\limits_{\mathcal{D}_{x_1,x_2}} p(\mathbf{x}_1,\mathbf{x}_2|\hat{\mathbf{A}}^{(k)}) \log f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{A}) d\mathbf{x}_1 d\mathbf{x}_2 \tag{E.32}$$

$$= C - \frac{1}{2} \iint\limits_{\mathcal{D}_{\mathbf{x}_1,\mathbf{x}_2}} p(\mathbf{x}_1,\mathbf{x}_2|\hat{\mathbf{A}}^{(k)}) \cdot (\mathbf{x}_2-\mathbf{A}\mathbf{x}_1)^{\mathsf{T}}\mathbf{V}^{-1}(\mathbf{x}_2-\mathbf{A}\mathbf{x}_1) d\mathbf{x}_1 d\mathbf{x}_2 \tag{E.33}$$

$$= C - \frac{1}{2}\Big(\mathsf{E}[\mathbf{X}_2^{\mathsf{T}}\mathbf{V}^{-1}\mathbf{X}_2|\hat{\mathbf{A}}^{(k)}] - 2\mathsf{E}[\mathbf{X}_2^{\mathsf{T}}\mathbf{V}^{-1}\mathbf{A}\mathbf{X}_1|\hat{\mathbf{A}}^{(k)}] + \mathsf{E}[\mathbf{X}_1^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\mathbf{V}^{-1}\mathbf{A}\mathbf{X}_1|\hat{\mathbf{A}}^{(k)}]\Big) \tag{E.34}$$

$$= C - \frac{1}{2}\mathsf{E}\left[\|\mathbf{X}_2 - \mathbf{A}\mathbf{X}_1\|^2_{\mathbf{V}^{-1}} \Big| \hat{\mathbf{A}}^{(k)}\right] \tag{E.35}$$

with

$$C = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{V}|. \tag{E.36}$$

Unfortunately, (E.35) does not have a nice form in $\mathbf{A}$. Unless a special structure is imposed onto $\mathbf{A}$, it is impossible to parameterise (E.35).

E.3.2 The AR case

Here, a special case of section Section E.3.1 is treated. The function $f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{a})$ is defined as

$$f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{a}) = \frac{1}{\sqrt{(2\pi)^n|\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)^{\mathsf{T}}\mathbf{V}^{-1}(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)\right) \tag{E.37}$$

with

$$\mathbf{A} \triangleq \begin{bmatrix} \mathbf{a}^{\mathsf{T}} \\ \mathbf{I} \quad \mathbf{0} \end{bmatrix}. \tag{E.38}$$

The message $h(\mathbf{a})$ becomes

$$h(\mathbf{a}) = \iint_{\mathcal{D}_{x_1,x_2}} p(\mathbf{x}_1, \mathbf{x}_2|\mathbf{a}^{(k)}) \log f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{a}) d\mathbf{x}_1 d\mathbf{x}_2 \tag{E.39}$$

$$= C - \frac{1}{2} \iint_{\mathcal{D}_{\mathbf{x}_1,\mathbf{x}_2}} p(\mathbf{x}_1, \mathbf{x}_2|\mathbf{a}^{(k)}) (\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^{\mathsf{T}}\mathbf{a})^{\mathsf{T}} \mathbf{V}^{-1} (\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^{\mathsf{T}}\mathbf{a}) d\mathbf{x}_1 d\mathbf{x}_2 \tag{E.40}$$

with

$$C = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{V}| \tag{E.41}$$

and where

$$\mathbf{S} \triangleq \begin{bmatrix} \mathbf{0}^{\mathsf{T}} & \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \qquad \mathbf{c} \triangleq (1, 0, \ldots, 0)^{\mathsf{T}} \tag{E.42}$$

The product $\mathbf{A}\mathbf{x}_1$ is separated in the shifting operation $\mathbf{S}\mathbf{x}_1$, which shifts every element in the vector $\mathbf{x}_1$ one position down, and the inner vector product $\mathbf{c}\mathbf{x}_1^{\mathsf{T}}\mathbf{a}$.

Again, the integral in (E.40) is the expectation of a quadratic form, which, in this case has the parameterisation

$$e^{h(\mathbf{a})} \propto \mathcal{N}_W\Big(\mathbf{a} \,\Big|\, \mathsf{E}[\mathbf{X}_1\mathbf{X}_1^{\mathsf{T}}|\hat{\mathbf{a}}^{(k)}]^{-1}\mathsf{E}[\mathbf{X}_1 X_2|\hat{\mathbf{a}}^{(k)}], \mathsf{E}[\mathbf{X}_1\mathbf{X}_1^{\mathsf{T}}|\hat{\mathbf{a}}^{(k)}]\mathbf{c}^{\mathsf{T}}\mathbf{V}^{-1}\mathbf{c}\Big) \tag{E.43}$$

The matrix $\mathbf{V}$ does not necessarily have to be a full covariance matrix. It suffices to set it to $\mathbf{V} = \mathbf{I}s, s \in \mathbb{R}^+$, where $\mathbf{I}$ denotes the identity matrix. Then the message becomes

$$\boxed{e^{h(\mathbf{a})} \propto \mathcal{N}_W\left(\mathbf{a} \,\middle|\, \mathsf{E}[\mathbf{X}_1\mathbf{X}_1^{\mathsf{T}}|\hat{\mathbf{a}}^{(k)}]^{-1}\mathsf{E}[\mathbf{X}_1 X_2|\hat{\mathbf{a}}^{(k)}], \frac{\mathsf{E}[\mathbf{X}_1\mathbf{X}_1^{\mathsf{T}}|\hat{\mathbf{a}}^{(k)}]}{s}\right)} \tag{E.44}$$

E.4 Joint Coefficient and Variance Estimation

Here, the problem of jointly estimating the coefficients and the variance of the driving noise of an autoregressive (AR) system is considered. The

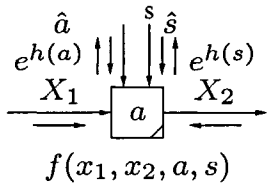


Figure E.5: Factor graph of the state transition node for joint coefficient/variance estimation.

node function $f(x_1, x_2, a, s)$ is defined as

$$f(x_1, x_2, a, s) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x_2 - ax_1)^2}{2s}\right). \tag{E.45}$$

with $a \in \mathbb{R}$ the scalar AR coefficient.

The message $h(a, s)$ becomes

$$h(a,s) = \iint\limits_{\mathcal{D}_{x_1,x_2}} p(x_1, x_2|\hat{a}^{(k)}, \hat{s}^{(k)}) \log f(x_1, x_2, a, s) dx_1 dx_2 \tag{E.46}$$

$$= \iint\limits_{\mathcal{D}_{x_1,x_2}} p(x_1, x_2|\hat{a}^{(k)}, \hat{s}^{(k)}) \left(-\frac{1}{2}\log(2\pi s) - \frac{(x_2 - ax_1)^2}{2s}\right) dx_1 dx_2 \tag{E.47}$$

$$= C - \frac{1}{2}\log s - \frac{1}{2s}\left(a^2 \mathsf{E}\left[X_1^2|\hat{a}^{(k)}, \hat{s}^{(k)}\right] - a2\mathsf{E}\left[X_1 X_2|\hat{a}^{(k)}, \hat{s}^{(k)}\right] + \mathsf{E}\left[X_2^2|\hat{a}^{(k)}, \hat{s}^{(k)}\right]\right) \tag{E.48}$$

with

$$C = -\frac{1}{2}\log(2\pi). \tag{E.49}$$

The message (E.48) is both a function of a and s. To find the maximum, all partial derivations of $h(a, s)$ are set to zero. From

$$\frac{\partial h(a,s)}{\partial a} = \iint\limits_{\mathcal{D}_{x_1,x_2}} p(x_1, x_2|\hat{a}^{(k)}, \hat{s}^{(k)}) \left(\frac{x_1(x_2 - ax_1)}{s}\right) dx_1 dx_2 \tag{E.50}$$

$$\stackrel{!}{=} 0 \tag{E.51}$$

$$\frac{\partial h(a,s)}{\partial s} = \iint\limits_{\mathcal{D}_{x_1,x_2}} p(x_1,x_2|\hat{a}^{(k)},\hat{s}^{(k)}) \left(-\frac{1}{2s} + \frac{(x_2-ax_1)^2}{2s^2}\right) dx_1 dx_2 \quad \text{(E.52)}$$

$$\stackrel{!}{=} 0 \quad \text{(E.53)}$$

it follows that

$$\hat{a}^{(k+1)} = \frac{\mathsf{E}\left[X_1X_2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{\mathsf{E}\left[X_1^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]} \quad \text{(E.54)}$$

$$\hat{s}^{(k+1)} = \mathsf{E}\left[(X_2-\hat{a}^{(k+1)}X_1)^2|\hat{a}^{(k)},\hat{s}^{(k)}\right] \quad \text{(E.55)}$$

Thus, the estimation of a and s is decoupled. We therefore can send the following messages separately:

$$\boxed{\begin{aligned} e^{h(a)} &\propto \mathcal{N}_W\left(a \,\middle|\, \frac{\mathsf{E}\left[X_1X_2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{\mathsf{E}\left[X_1^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}, \frac{\mathsf{E}\left[X_1^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{\hat{s}^{(k)}}\right) \\ e^{h(s)} &\propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{\mathsf{E}\left[(X_2-\hat{a}^{(k+1)}X_1)^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{2}\right) \end{aligned}}$$

E.4.1 The vector case

The node function $f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{a},s)$ is defined as

$$f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{a},s) = \frac{1}{\sqrt{(2\pi)^2|\mathbf{V}|}} \exp\left(-\frac{(\mathbf{x}_2-\mathbf{A}\mathbf{x}_1)^\mathsf{T}(\mathbf{x}_2-\mathbf{A}\mathbf{x}_1)}{2s}\right) \quad \text{(E.56)}$$

with

$$\mathbf{A} \triangleq \begin{bmatrix} \mathbf{a}^\mathsf{T} \\ \mathbf{I} \quad \mathbf{0} \end{bmatrix} \quad \text{(E.57)}$$

and $\mathbf{x}_1,\mathbf{x}_2 \in \mathbb{R}^n$, $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{V} = \mathbf{I}s$.

The message $h(\mathbf{a}, s)$ becomes

$$h(\mathbf{a}, s) = \iint_{\mathcal{D}_{\mathbf{x}_1,\mathbf{x}_2}} p(\mathbf{x}_1, \mathbf{x}_2 | \hat{\mathbf{a}}^{(k)} \hat{s}^{(k)}) \log f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{a}, s) d\mathbf{x}_1 d\mathbf{x}_2 \quad \text{(E.58)}$$

$$= C - \frac{1}{2s} \iint_{\mathcal{D}_{\mathbf{x}_1,\mathbf{x}_2}} p(\mathbf{x}_1, \mathbf{x}_2 | \hat{\mathbf{a}}^{(k)} \hat{s}^{(k)}) (\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^{\mathsf{T}}\mathbf{a})^{\mathsf{T}} (\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^{\mathsf{T}}\mathbf{a}) d\mathbf{x}_1 d\mathbf{x}_2 \quad \text{(E.59)}$$

with

$$C = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{V}| \quad \text{(E.60)}$$

and where

$$\mathbf{S} \triangleq \begin{bmatrix} \mathbf{0}^{\mathsf{T}} & \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \qquad \mathbf{c} \triangleq (1, 0, \ldots, 0)^{\mathsf{T}} \quad \text{(E.61)}$$

Again, the estimation of $\mathbf{a}$ and s is decoupled. Thus, we can write $h(\mathbf{a})$ and $h(s)$ separately:

$$\boxed{\begin{aligned} e^{h(\mathbf{a})} &\propto \mathcal{N}_W\left(\mathbf{a} \,\middle|\, \mathsf{E}[\mathbf{X}_1\mathbf{X}_1^{\mathsf{T}}]^{-1}\mathsf{E}[\mathbf{X}_1 X_2], \frac{\mathsf{E}[\mathbf{X}_1\mathbf{X}_1^{\mathsf{T}}]}{\hat{s}^{(k)}}\right) \\ e^{h(s)} &\propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{\mathsf{E}[(\mathbf{X}_2 - \hat{\mathbf{A}}^{(k+1)}\mathbf{X}_1)^{\mathsf{T}}(\mathbf{X}_2 - \hat{\mathbf{A}}^{(k+1)}\mathbf{X}_1)]}{2}\right) \end{aligned}}$$

where the expectations are w.r.t. the distribution $p(\mathbf{x}_1, \mathbf{x}_2 \mid \hat{\mathbf{a}}^{(k)}, \hat{s}^{(k)})$.

E.5 Finite state machine

This node is not used in this thesis. Nonetheless, its derivation is shown here, because it appears in one of the prime applications of the EM algorithm. The function of a node implementing the state transition of a finite state machine is (cf. Fig. E.6)

$$f(x_1, x_2, A) = a_{x_1 x_2} = \sum_{i,j} a_{ij} \delta[x_1 - i] \delta[x_2 - j]. \quad \text{(E.62)}$$

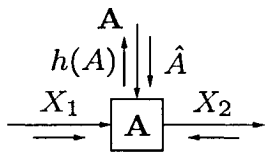


Figure E.6: Factor graph node of the state transition node for the finite state machine.

where $\delta[.]$ denotes the Kronecker delta. The h-message is

$$h(\mathbf{A}) = \sum_{x_1,x_2} p(x_1, x_2 \mid \hat{\mathbf{A}}^{(k)}) \log f(x_1, x_2, \mathbf{A}) \tag{E.63}$$

$$= \sum_{x_1,x_2} p(x_1, x_2 \mid \hat{\mathbf{A}}^{(k)}) \log a_{x_1,x_2} \tag{E.64}$$

This message can be represented as matrix with individual elements $p(x_1, x_2 \mid \hat{\mathbf{A}}^{(k)}) \log a_{x_1,x_2}$.

To find the estimate $\hat{\mathbf{A}}$, one has to compute the derivations

$$\frac{\partial h(\mathbf{A})}{\partial a_{ij}} + \lambda \frac{\partial g(\mathbf{A})}{\partial a_{ij}} = \frac{p(i, j \mid \hat{\mathbf{A}}^{(k)})}{a_{ij}} - \lambda \stackrel{!}{=} 0 \tag{E.65}$$

with the constraint

$$g(\mathbf{A}) = 1 - \sum_{\ell} a_{i\ell} = 0. \tag{E.66}$$

Therefore

$$\hat{a}_{ij} = \frac{p(i, j \mid \hat{\mathbf{A}}^{(k)})}{\lambda}. \tag{E.67}$$

To find the value of the Lagrange multiplier λ plug (E.67) into the constraint (E.66)

$$\sum_{\ell} \hat{a}_{ij} = \frac{1}{\lambda} \sum_{\ell} p(i, \ell \mid \hat{\mathbf{A}}^{(k)}) = 1 \tag{E.68}$$

$$\lambda = \sum_{\ell} p(i, \ell \mid \hat{\mathbf{A}}^{(k)}) \tag{E.69}$$

The new estimates of the transition probabilities a_{ij} thus becomes

$$\boxed{\hat{a}_{ij} = \frac{p(i, j \mid \hat{\mathbf{A}}^{(k)})}{\sum_{\ell} p(i, \ell \mid \hat{\mathbf{A}}^{(k)})}} \tag{E.70}$$

E.6 Computing the expectations of Table 5.5

In this section it is explained how the expectations in the message passing EM update rules of Table 5.5 are computed from the sum-product messages when the incoming messages are Gaussian.

E.6.1 Scalar case

The expectation $\mathsf{E}[X_1 X_2|\hat{a}^{(k)}]$ is derived from the density $p(x_1, x_2|\hat{a}^{(k)})$ which is given by

$$p(x_1, x_2|\hat{a}^{(k)}) \propto \mu_{X_1\to f}(x_1) f(x_1, x_2, \hat{a}^{(k)}) \mu_{X_2\to f}(x_2) \tag{E.71}$$

$$\propto \mathcal{N}(\mathbf{x} \mid \mathbf{m}_{1,2}, \mathbf{W}_{1,2}) \tag{E.72}$$

where $\mathbf{x} \triangleq (x_2, x_1)^{\mathsf{T}}$, the node function is

$$f(x_1, x_2, \hat{a}^{(k)}) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x_2 - \hat{a}^{(k)} x_1)^2}{2s}\right) \tag{E.73}$$

and the incoming message $\mu_{X_1\to f}(x_1)$ is Gaussian with mean m_1 and weight w_1 and similarly m_2 and w_2 for $\mu_{X_2\to f}(x_2)$. (E.71) can be interpreted as propagating three augmented messages through an equality-node. We, therefore, can apply the update rule of Table 4.1-1 extended to three incoming messages. The mean vectors and weight matrices of the augmented messages are

$$\tilde{\mathbf{m}}_1 = (0, m_1)^{\mathsf{T}} \tilde{\mathbf{W}}_1 = \begin{bmatrix} 0 & 0 \\ 0 & w_1 \end{bmatrix} \tag{E.74}$$

$$\tilde{\mathbf{m}}_2 = (m_2, 0)^{\mathsf{T}} \tilde{\mathbf{W}}_2 = \begin{bmatrix} w_2 & 0 \\ 0 & 0 \end{bmatrix} \tag{E.75}$$

$$\tilde{\mathbf{m}}_f = \mathbf{0} \qquad \tilde{\mathbf{W}}_f = \frac{1}{s} \begin{bmatrix} 1 & -a \\ -a & a^2 \end{bmatrix} \tag{E.76}$$

The vector and the matrix (E.76) are found by comparing the coefficients of

$$\frac{(x_2 - a x_1)^2}{s} = (\mathbf{x} - \tilde{\mathbf{m}}_f)^{\mathsf{T}} \tilde{\mathbf{W}}_f (\mathbf{x} - \tilde{\mathbf{m}}_f) \tag{E.77}$$

The joint density (E.71) is a Gaussian with mean vector and weight matrix

$$\mathbf{m}_{1,2} = (\mathbf{W}_1 + \mathbf{W}_f + \mathbf{W}_2)^{-1}(\mathbf{W}_1\mathbf{m}_1 + \mathbf{W}_f\mathbf{m}_f + \mathbf{W}_2\mathbf{m}_2) \quad \text{(E.78)}$$

$$= \mathbf{W}_{1,2}^{-1} \begin{pmatrix} w_2 m_2 \\ w_1 m_1 \end{pmatrix} \quad \text{(E.79)}$$

$$\mathbf{W}_{1,2} = \mathbf{W}_1 + \mathbf{W}_f + \mathbf{W}_2. \quad \text{(E.80)}$$

The correlation matrix finally is

$$\begin{bmatrix} \mathsf{E}[X_2^2] & \mathsf{E}[X_1 X_2] \\ \mathsf{E}[X_2 X_1] & \mathsf{E}[X_1^2] \end{bmatrix} = \mathbf{W}_{1,2}^{-1} + \mathbf{m}_{1,2}\mathbf{m}_{1,2}^{\mathsf{T}}. \quad \text{(E.81)}$$

E.6.2 Vector case

Here the joint density of X_1 and X_2 given $\hat{\mathbf{a}}^{(k)}$ is

$$p(\mathbf{x}_1, \mathbf{x}_2 | \hat{\mathbf{a}}^{(k)}) \propto \mu_{\mathbf{X}_1 \to f}(\mathbf{x}_1) f(\mathbf{x}_1, \mathbf{x}_2, \hat{\mathbf{a}}^{(k)}) \mu_{\mathbf{X}_2 \to f}(\mathbf{x}_2) \quad \text{(E.82)}$$

$$\propto \mathcal{N}(\mathbf{x} \mid \mathbf{m}_{1,2}, \mathbf{W}_{1,2}) \quad \text{(E.83)}$$

where $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}$ are the following vectors:

$$\mathbf{x}_1 = (x_{n-1}, x_{n-2}, \ldots, x_{n-M})^{\mathsf{T}} \quad \text{(E.84)}$$

$$\mathbf{x}_2 = (x_n, x_{n-1}, \ldots, x_{n-M+1})^{\mathsf{T}} \quad \text{(E.85)}$$

$$\mathbf{x} = (x_n, x_{n-1}, \ldots, x_{n-M+1}, x_{n-M})^{\mathsf{T}} \quad \text{(E.86)}$$

Because of the special structure of the transition matrix $\mathbf{A}$ (cf. (3.10)) the augmented mean vectors and weight matrices are

$$\tilde{\mathbf{m}}_1 = (0, \mathbf{m}_1)^{\mathsf{T}} \tilde{\mathbf{W}}_1 = \begin{bmatrix} 0 & \mathbf{0}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{W}_1 \end{bmatrix} \quad \text{(E.87)}$$

$$\tilde{\mathbf{m}}_2 = (\mathbf{m}_2, 0)^{\mathsf{T}} \tilde{\mathbf{W}}_2 = \begin{bmatrix} \mathbf{W}_2 & \mathbf{0}^{\mathsf{T}} \\ \mathbf{0} & 0 \end{bmatrix} \quad \text{(E.88)}$$

$$\tilde{\mathbf{m}}_f = \mathbf{0} \quad \text{(E.89)}$$

$$\tilde{\mathbf{W}}_f = \frac{1}{s} \left[\begin{array}{c|ccc} 1 & -a_1 & \cdots & -a_n \\ \hline -a_1 & a_1^2 & a_1 a_2 & \cdots \\ \vdots & a_1 a_2 & \ddots & \vdots \\ & \vdots & & a_{n-1}^2 \quad a_n a_{n-1} \\ -a_n & & \cdots & a_n a_{n-1} \quad a_n^2 \end{array}\right] = \left[\begin{array}{c|c} 1 & -\mathbf{a}^{\mathsf{T}} \\ \hline -\mathbf{a} & \mathbf{a}\mathbf{a}^{\mathsf{T}} \end{array}\right] \quad \text{(E.90)}$$

The vector (E.89) and the matrix (E.90) are found by comparing the coefficients of

$$\frac{(x_n - \sum_{k=1}^{n} a_k x_{n-k})^2}{s} = (\tilde{\mathbf{x}} - \tilde{\mathbf{m}}_f)^{\mathsf{T}} \tilde{\mathbf{W}}_f (\tilde{\mathbf{x}} - \tilde{\mathbf{m}}_f) \tag{E.91}$$

The joint density (E.82) is Gaussian with mean vector and weight matrix

$$\mathbf{m}_{1,2} = (\tilde{\mathbf{W}}_1 + \tilde{\mathbf{W}}_f + \tilde{\mathbf{W}}_2)^{-1} (\tilde{\mathbf{W}}_1 \tilde{\mathbf{m}}_1 + \tilde{\mathbf{W}}_f \tilde{\mathbf{m}}_f + \tilde{\mathbf{W}}_2 \tilde{\mathbf{m}}_2) \tag{E.92}$$

$$= \mathbf{W}_{1,2}^{-1} \left[\begin{pmatrix} 0 \\ \mathbf{W}_1 \mathbf{m}_1 \end{pmatrix} + \begin{pmatrix} \mathbf{W}_2 \mathbf{m}_2 \\ 0 \end{pmatrix} \right] \tag{E.93}$$

$$\mathbf{W}_{1,2} = \tilde{\mathbf{W}}_1 + \tilde{\mathbf{W}}_f + \tilde{\mathbf{W}}_2. \tag{E.94}$$

The correlation matrix finally is

$$\begin{bmatrix} \mathsf{E}[X_2^2] & \mathsf{E}[\mathbf{X}_1 X_2] \\ \mathsf{E}[X_2 \mathbf{X}_1^{\mathsf{T}}] & \mathsf{E}[\mathbf{X}_1 \mathbf{X}_1^{\mathsf{T}}] \end{bmatrix} = \mathbf{W}_{1,2}^{-1} + \mathbf{m}_{1,2} \mathbf{m}_{1,2}^{\mathsf{T}}. \tag{E.95}$$

Abbreviations

AR	Autoregression
AWGN	Additive White Gaussian Noise
EM	Expectation Maximisation
FFG	Forney-style Factor Graph
FIR	Finite Impulse Response
GM	Gaussian Mixture
HMM	Hidden Markov Model
ICM	Iterative Conditional Modes
i.i.d.	independent identically distributed
LMS	Least Mean Squares
LPC	Linear Predictive Coding
MAP	Maximum A Posteriori
ML	Maximum-Likelihood
MMSE	Minimum Mean Squared Error
MSE	Mean Squared Error
pdf	probability density function
pmf	probability mass function
RLS	Recursive Least Squares
r.v.	random variable
SNR	Signal-to-Noise Ratio
SPA	Summary-Propagation Algorithm
SVD	Singular Value Decomposition
w.r.t.	with respect to

Seite Leer /
Blank leaf

List of Symbols

Elementary

X	Random variable
x	Value of random variable X
$\hat{x}$	Estimate of variable (or parameter) x
$\mathbf{x}$	Vector
$\mathbf{A}$	Matrix
$\triangleq$	Definition
$\propto$	Proportional to
$\stackrel{!}{=}$	Set to equality

Algebra

$\mathbb{Z}$	Ring of integers
$\mathbb{R}$	Field of real numbers
$\mathbb{R}^+$	Field of positive real numbers
$\mathbb{R}_0^+$	Field of non-negative real numbers
$\mathbb{C}$	Field of complex numbers

Linear Algebra

$\mathbf{A}^{\mathsf{T}}$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^{\mathsf{H}}$	Hermitian transpose of matrix $\mathbf{A}$
$\mathrm{diag}\,(\cdots)$	Diagonal matrix
$\mathrm{tr}\,(\cdot)$	Trace operator
$\mathbf{I}$	Identity matrix

Probability

$\mathcal{N}(\,.\mid \mu, \sigma^2)$	Scalar Gaussian distribution with mean μ and variance σ^2
$\mathcal{N}(\,.\mid \mathbf{m}, \mathbf{V})$	Multivariate Gaussian distribution with mean vector $\mathbf{m}$ and covariance matrix $\mathbf{V}$
$\mathcal{N}_W(\,.\mid \mathbf{m}, \mathbf{W})$	Multivariate Gaussian distribution with mean vector $\mathbf{m}$ and weight matrix $\mathbf{W}$
$\mathcal{N}_{\xi,W}(\,.\mid \mathbf{m}, \mathbf{W})$	Multivariate Gaussian distribution with weighted mean vector $\boldsymbol{\xi}$ and weight matrix $\mathbf{W}$
$\mathbf{W}$	Weight matrix of a Gaussian distribution
$\mathbf{V}$	Covariance matrix of a Gaussian distribution
$\mathbf{m}$	Mean vector of a Gaussian distribution
$\boldsymbol{\xi}$	Weighted mean vector of a Gaussian distribution
$\mathrm{Ig}\,(\,.\mid \alpha, \beta)$	Inverted-gamma distribution with parameters α, β
$\mathsf{E}[X]$	Expectation of r.v. X
$\mathsf{V}[X]$	Variance of r.v. X
$\mathsf{M}[X]$	Mode (i.e. maximum) of the distribution of r.v. X
$\{X_n\}$	Stochastic process

Factor graph

$\mu_{f\to X}(x)$	Message leaving node f along edge X
$\mu_{X\to f}(x)$	Message arriving at node f along edge X
$\nabla \log f(\hat{\theta})$	Gradient message
$(x^{(i)}, \rho^{(i)})$	Particle at position $x^{(i)}$ with weight $\rho^{(i)}$
$h(\theta)$	Local EM message

Kalman filter

M	Order of the auto-regressive model
$\mathbf{A}$	State transition matrix defined by (3.10)
$\hat{\mathbf{x}}_{n\|\overrightarrow{n-1}}$	State estimate at time n given all observations up to time $n-1$ (prediction)
$\hat{\mathbf{x}}_{n\|\overrightarrow{n}}$	State estimate at time n given all observations up to time n (correction)
$\mathbf{V}_{n\|\overrightarrow{n-1}}$	State covariance matrix at time n given all observations up to time $n-1$.
$\mathbf{V}_{n\|\overrightarrow{n}}$	State covariance matrix at time n given all observations up to time n.
$\mathbf{W}_{n\|\overrightarrow{n-1}}$	Weight matrix of the state at time n given all observations up to time $n-1$.
$\mathbf{W}_{n\|\overrightarrow{n}}$	Weight matrix of the state at time n given all observations up to time n.
$e_{n\|\overrightarrow{n-1}}$	Kalman error at time n
$\mathbf{k}_n$	Kalman gain at time n
$\mathbf{g}_n$	Kalman gain for the information filter at time n
σ_U^2	Input noise variance
σ_W^2	Observation noise variance
$\hat{\mathbf{x}}_{n\|\overleftarrow{n+1}}$	State estimate at time n given all observations from time $n+1$ on (backward prediction)
$\hat{\mathbf{x}}_{n\|\overleftarrow{n}}$	State estimate at time n given all observations from time n on (backward correction)
$e_{n\|\overleftarrow{n+1}}$	Backward Kalman error at time n

Miscellaneous

$\hat{\theta}^{(k)}$	value of θ in the k-th iteration
$\mathbf{x}^{[i]}$	i-th element of vector $\mathbf{x}$
$\mathbf{A}^{[i,j]}$	Element of matrix $\mathbf{A}$ in row i and column j
$\mathcal{D}_x$	Domain of variable x

Seite Leer /
Blank leaf

List of Figures

Bibliography

[1] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Transactions on Information Theory*, vol. 46, no. 2, 2000.

[2] D. L. Alspach and H. W. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Transactions on Automatic Control*, vol. 17, no. 4, 1972.

[3] B. D. O. Anderson and J. B. Moore, *Optimal Filtering.* Prentice Hall, 1979.

[4] C. Andrieu and A. Doucet, "Online expectation-maximization type algorithms for parameter estimation in general state space models," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, pp. 69–72, 2003.

[5] C. Andrieu, A. Doucet, S. S. Singh, and V. B. Tadić, "Particle methods for change detection, system identification, and control," *Proceedings of the IEEE*, vol. 92, no. 3, 2004.

[6] C. Andrieu, A. Doucet, and V. B. Tadić, "Online sampling for parameter estimation in general state space models," *Proc. IFAC Symposium on System Identification*, 2003.

[7] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002.

[8] P. R. Bélanger, "Estimation of noise covariance matrices for a linear time-varying stochastic process," *Automatica*, vol. 10, pp. 267–275, 1974.

[9] J. M. Bernardo, *Bayesian Theory.* John Wiley&Sons, 2000.

[10] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," International Computer Science Institute, Berkeley, Tech. Rep. TR-97-021, 1998.

[11] ——, "Graphical models and automatic speech recognition," Department of Electrical Engineering, University of Washington, Tech. Rep. UWEETR-2001-0005, 2001.

[12] X. Boyen and D. Koller, "Approximate learning of dynamic models," *Neural Information Processing Systems Conference*, pp. 396–402, 1998.

[13] M. Brookes, "Matrix reference manual," *online available at http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html*, 2005.

[14] K.-J. Bry and J. le Roux, "Comparison of some algorithms for identifying autoregressive signals in the presence of observation noise," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 224–227, 1982.

[15] W. Buntine, "A guide to the literature on learning probabilistic networks from data," *IEEE Transactions Knowledge and Data Engineering*, vol. 8, no. 2, 1996.

[16] W. L. Buntine, "Operations for learning with graphical models," *Journal of Artificial Intelligence Research*, pp. 159–225, 1994.

[17] H. Cai, E. Grivel, and M. Najim, "Speech enhancement using AR model driven by white noise with time-varying variance," *3-rd COST 276 Workshop in Information and Knowledge Management for Integrated Media Communication, Budapest (Hungary)*, 2002.

[18] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," *IEE Proc.-Radar, Sonar Navig.*, vol. 146, no. 1, pp. 2–7, 1999.

[19] S. Chib and E. Greenberg, "Bayes inference in regression models with ARMA(p, q) errors," *Journal of Econometrics*, vol. 64, pp. 183–206, 1994.

[20] P. Dagum and M. Luby, "An optimal approximation algorithm for Bayesian inference," *Artificial Intelligence*, vol. 93, no. 1-2, 1997.

[21] J. Dauwels, H.-A. Loeliger, P. Merkli, and M. Ostojic, "On Markov-structured summary propagation and LFSR synchronization," *Proc. 42nd Allerton Conf. on Communication, Control, and Computing*, 2004.

[22] J. Dauwels, S. Korl, and H.-A. Loeliger, "Expectation maximization as message passing," *IEEE International Symposium on Information Theory (ISIT)*, 2005.

[23] ——, "Steepest descent on factor graphs," *IEEE ITSOC Information Theory Workshop on Coding and Complexity*, 2005.

[24] C. E. Davila, "A subspace approach to estimation of autoregressive parameters from noisy measurements," *IEEE Transactions on Signal Processing*, vol. 46, no. 2, 1998.

[25] F. Dellaert, "The expectation maximization algorithm," College of Computing, Georgia Institute of Technology, Tech. Rep. GIT-GVU-02-20, 2002.

[26] J. R. Deller Jr., J. Hansen, and J. G. Proakis, *Discrete-Time Processing of Speech Signals.* IEEE Press, 2000.

[27] J. P. Delmas, "An equivalence of the EM and ICE algorithm for exponential family," *IEEE Transactions on Signal Processing*, vol. 45, no. 10, 1997.

[28] A. Dembo and O. Zeitouni, "Maximum a posteriori estimation of time-varying ARMA processes from noisy observations," *IEEE Transactions on Speech and Audio Processing*, vol. 36, no. 4, 1988.

[29] 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, 1977.

[30] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez, "Particle filtering," *IEEE Signal Processing Magazine*, September 2003.

[31] G. Doblinger, "Smoothing of noisy AR signals using an adaptive Kalman filter," *European Signal Processing Conference*, pp. 781–784, 1998.

[32] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice.* Springer-Verlag, 2001.

[33] ——, "An introduction to sequential Monte Carlo methods," in *Sequential Monte Carlo Methods in Practice.* Springer-Verlag, 2001.

[34] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," Signal Processing Group, Department of Engineering, University of Cambridge, Tech. Rep., 2000.

[35] H. Driessen and Y. Boers, "An efficient particle filter for jump Markov nonlinear systems," *IEE, Professional Networks, Control&Automation*, 2004.

[36] A. W. Eckford, "Channel estimation in block fading channels using the factor graph EM algorithm," *22nd Biennial Symposium on Communications, Kingston, Ontario, Canada*, 2004.

[37] R. J. Elliott and V. Krishnamurthy, "New finite-dimensional filters for parameter estimation of discrete-time linear Gaussian models," *IEEE Transactions on Automatic Control*, vol. 44, no. 5, 1999.

[38] J. A. Fessler and A. O. Hero, "Space-alternating generalized expectation-maximization algorithm," *IEEE Transactions on Signal Processing*, vol. 42, no. 10, 1994.

[39] W. Fong, S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing with application to audio signal enhancement," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002.

[40] G. D. Forney, "Notes on Gaussian random vectors, Fourier transforms of Gaussian distributions, and belief propagation," unpublished.

[41] ——, "Codes on graphs: Normal realizations," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 520–548, 2001.

[42] B. J. Frey, P. Dayan, and G. E. Hinton, "A simple algorithm that discovers efficient perceptual codes," in *Computational and Biological Mechanisms of Visual Coding.* Cambridge University Press, 1997.

[43] B. J. Frey and N. Jojic, "A comparison of algorithms for inference and learning in probabilistic graphical models," University of Toronto, Tech. Rep. PSI-2003-22, 2003.

[44] N. Friedman, "Learning belief networks in the presence of missing values and hidden variables," *International Conference on Machine Learning*, 1997.

[45] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2-3, 1997.

[46] N. Friedman and M. Goldszmidt, "Learning Bayesian networks with local structure," in *Learning in Graphical Models*, M. I. Jordan, Ed. MIT Press, 1999.

[47] M. Gabrea, "Robust adaptive Kalman filtering-based speech enhancement algorithm," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. I, pp. 301–304, 2004.

[48] S. Gannot, D. Burshtein, and E. Weinstein, "Iterative and sequential Kalman filter-based speech enhancement algorithms," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 4, pp. 373–385, 1998.

[49] S. Gannot, "Algorithms for single microphone speech enhancement," Master's thesis, Tel Aviv-University, 1995.

[50] J. D. Gibson, B. Koo, and S. D. Gray, "Filtering of colored noise for speech enhancement and coding," *IEEE Transactions on Signal Processing*, vol. 39, no. 8, 1991.

[51] W. R. Gilks and C. Berzuini, "Following a moving target - Monte Carlo inference for dynamic Bayesian models," *J.R.Statist.Soc.B*, vol. 63, pp. 127–146, 2001.

[52] W. M. Hartmann, *Signals, Sound, and Sensation.* Springer-Verlag, 1998.

[53] M. K. Hasan, M. J. Hossain, and M. A. Haque, "Parameter estimation of multichannel autoregressive processes in noise," *Signal Processing*, vol. 83, no. 3, 2003.

[54] S. Haykin, *Adaptive filter theory.* Prentice Hall, 1996.

[55] S. Haykin, A. H. Sayed, J. R. Zeidler, P. Yee, and P. C. Wei, "Adaptive tracking of linear time-variant systems by extended RLS algorithms," *IEEE Transactions on Signal Processing*, vol. 45, no. 5, 1997.

[56] D. Heckerman, "A tutorial on learning with Bayesian networks," in *Learning in Graphical Models*, M. I. Jordan, Ed. MIT Press, 1999.

[57] C. Herzet, V. Ramon, and L. Vandendorpe, "Turbo synchronization: A combined sum-product and expectation-maximization algorithm approach," *IEEE Workshop on Signal Processing Advances in Wireless Communications*, 2005.

[58] T. Heskes and O. Zoeter, "Expectation propagation for approximate inference in dynamic Bayesian networks," *Proceedings Uncertainty in Artificial Intelligence*, pp. 216–223, 2002.

[59] C. Huang and A. Darwiche, "Inference in belief networks: A procedural guide," *International Journal of Approximate Reasoning*, vol. 11, pp. 1–45, 1994.

[60] A. T. Ihler, E. B. Sudderth, W. T. Freeman, and A. S. Willsky, "Efficient multiscale sampling from products of Gaussian mixtures," *Neural Information Processing Systems Conference*, 2003.

[61] M. Isard, "PAMPAS: real-valued graphical models for computer vision," *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[62] R. H. Jones, "Maximum likelihood fitting of ARMA models to time series with missing observations," *Technometrics*, vol. 22, no. 3, 1980.

[63] M. I. Jordan and Y. Weiss, "Graphical models: Probabilistic inference," EECS Computer Science Division, Tech. Rep., 2000.

[64] T. Kailath and A. H. Sayed, Eds., *Fast Reliable Algorithms for Matrices with Structure.* Society for Industrial and Applied Mathematics, 1999.

[65] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME-Journal of Basic Engineering*, vol. 82 (Series D), pp. 35–45, 1960.

[66] S. M. Kay, "Noise compensation for autoregressive spectral estimates," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 3, 1980.

[67] ——, *Fundamentals of Statistical Signal Processing, Estimation Theory.* Prentice Hall, 1993.

[68] S. M. Kay and V. Nagesha, "Maximum likelihood estimation of signals in autoregressive noise," *IEEE Transactions on Signal Processing*, vol. 42, no. 1, 1994.

[69] S. Korl, H.-A. Loeliger, and A. G. Lindgren, "AR model parameter estimation: From factor graphs to algorithms," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004.

[70] J. H. Kotecha and P. M. Djurić, "Gaussian particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, 2003.

[71] ——, "Gaussian sum particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, 2003.

[72] B. Kovačević, M. Milosavljević, and M. Veinović, "Robust recursive AR speech analysis," *Signal Processing*, vol. 44, pp. 125–138, 1995.

[73] P. J. Krause, "Learning probabilistic networks," Philips Research Laboratories, UK, Tech. Rep., 1998.

[74] 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, 2001.

[75] M. Kuropatwinski and W. Kleijn, "Minimum mean square error estimation of speech short-term predictor parameters under noisy conditions," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 96–99, 2003.

[76] C. Kwok, D. Fox, and M. Meilă, "Real-time particle filters," *Proceedings of the IEEE*, vol. 92, no. 3, 2004.

[77] S. L. Lauritzen, "The EM algorithm for graphical association models with missing data," *Computational Statistics & Data Analysis*, vol. 19, pp. 191–201, 1995.

[78] J. F. Leathrum, "On sequential estimation of state noise variances," *IEEE Transactions on Automatic Control*, vol. 26, no. 3, 1981.

[79] D. S. Lee and N. K. Chia, "A particle algorithm for sequential Bayesian parameter estimation and model selection," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002.

[80] K. Lee and S. Jung, "Time-domain approach using multiple Kalman filters and EM algorithm to speech enhancement with nonstationary noise," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 3, 2000.

[81] D. Lippuner and G. S. Moschytz, "The Kalman filter in the context of adaptive filter theory," *Int. J. Circ. Theor. Appl.*, vol. 32, pp. 223–253, 2004.

[82] J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[83] H.-A. Loeliger, "Least squares and Kalman filtering on Forney graphs," in *Codes, Graphs, and Systems*, R. Blahut and R. Koetter, Eds. Kluwer, 2002.

[84] ——, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, Jan. 2004.

[85] ——, "Some remarks on factor graphs," *3rd International Symposium on Turbo Codes & related topics*, 2003.

[86] H.-A. Loeliger, J. Dauwels, V. M. Koch, and S. Korl, "Signal processing with factor graphs: Examples," *International Symposium on Control, Communications and Signal Processing, Hammamet, Tunisia*, 2004.

[87] H. Lucke, "Which stochastic models allow Baum-Welch training?" *IEEE Transactions on Signal Processing*, vol. 44, no. 11, 1996.

[88] D. J. C. MacKay, "Introduction to Monte Carlo methods," in *Learning in Graphical Models*, ser. NATO Science Series, M. I. Jordan, Ed. Kluwer Academic Press, 1998, pp. 175–204.

[89] D. J. MacKay, J. S. Yedidia, W. T. Freeman, and Y. Weiss, "A conversation about the Bethe free energy and sum-product," Mitsubishi Electric Research Laboratories, Tech. Rep. TR-2001-18, 2001.

[90] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, 1975.

[91] U. E. Makov and A. F. M. Smith, "A quasi-Bayes unsupervised learning procedure for priors," *IEEE Transactions on Information Theory*, vol. 23, no. 6, 1977.

[92] M. B. Malik, "State-space recursive least-squares: Part I," *Signal Processing*, vol. 84, pp. 1709–1718, 2004.

[93] ——, "State-space recursive least-squares: Part II," *Signal Processing*, vol. 84, pp. 1709–1718, 2004.

[94] P. McGiffin and D. Murthy, "Parameter estimation for auto-regressive systems with missing observations," *International Journal of Systems Science*, vol. 11, no. 9, pp. 1021–1034, 1980.

[95] ——, "Parameter estimation for auto-regressive systems with missing observations—part II," *International Journal of Systems Science*, vol. 12, no. 6, pp. 657–663, 1981.

[96] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions.* John Wiley&Sons, 1997.

[97] R. K. Mehra, "On the identification of variances and adaptive Kalman filtering," *IEEE Transactions on Automatic Control*, vol. 15, no. 2, 1970.

[98] ——, "Approaches to adaptive filtering," *IEEE Transactions on Automatic Control*, vol. 17, no. 5, 1972.

[99] T. P. Minka, "Expectation propagation for approximate Bayesian inference," Statistics Department, Carnegie Mellon University, Tech. Rep., 2001.

[100] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, pp. 47–60, November 1996.

[101] K. P. Murphy, "From belief propagation to expectation propagation," University of British Columbia, Vancouver, Canada, Tech. Rep., 2001.

[102] ——, "An introduction to graphical models," University of British Columbia, Vancouver, Canada, Tech. Rep., 2001.

[103] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, 1999.

[104] K. P. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. dissertation, University of California, Berkely, 2002.

[105] K. A. Myers and B. D. Tapley, "Adaptive sequential estimation with unknown noise statistics," *IEEE Transactions on Automatic Control*, vol. 21, no. 4, 1976.

[106] R. M. Neal, "Probabilisitc inference using Markov chain Monte Carlo methods," Department of Computer Science, University of Toronto, Tech. Rep., 1993.

[107] R. M. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants." in *Learning in Graphical Models*, M. I. Jordan, Ed. MIT Press, 1999.

[108] F. Nordén and T. Eriksson, "Time evolution in LPC spectrum coding," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 3, 2004.

[109] J. A. O'Sullivan, "Alternating minimization algorithms: From Blahut-Arimoto to expectation-maximization," in *Codes, Curves, and Signals. Common Threads in Communications*, A. Vardy, Ed. Kluwer Academic Publishers, 1999.

[110] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers, 1988.

[111] K. Plarre and P. Kumar, "Extended message passing algorithm for inference in loopy Gaussian graphical models," *Ad Hoc Networks*, vol. 2, pp. 153–169, 2004.

[112] R. Prado, G. Huerta, and M. West, "Bayesian time-varying autoregressions: Theory, methods and applications," *Journal of the Institute of Mathematics and Statistics of the University of Sao Paolo, Special issue on Time Series and Related Topics*, 2002.

[113] J. L. Rojo-Álvarez, M. Martínez-Ramón, M. de Prado-Cumplido, A. Artés-Rodríguez, and A. R. Figueiras-Vidal, "Support vector method for robust ARMA system identification," *IEEE Transactions on Signal Processing*, vol. 52, no. 1, 2004.

[114] S. Roweis and Z. Ghahramani, "A unifying review of linear Gaussian models," *Neural Computation*, vol. 11, pp. 305–345, 1999.

[115] G. M. K. Saleh, M. Niranjan, and W. J. Fitzgerald, "The application of Bayesian inference to linear prediction of speech," Cambridge University Engineering Department, Tech. Rep., 1994.

[116] A. H. Sayed and T. Kailath, "A state-space approach to adaptive RLS filtering," *IEEE Signal Processing Magazine*, vol. 11, July 1994.

[117] Y. Shi and E. Chang, "Spectrogram-based formant tracking via particle filters," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 168–171, 2003.

[118] R. Shumway and D. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of Time Series Analysis*, vol. 3, no. 4, 1982.

[119] A. C. Singer, "Universal linear prediction by model order weighting," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, 1999.

[120] P. Smyth, "Belief networks, hidden Markov models, and Markov random fields: A unifying view," *Pattern Recognition Letters*, vol. 18, pp. 1261–1268, 1997.

[121] P. Smyth, D. Heckerman, and M. I. Jordan, "Probabilistic independence networks for hidden Markov probability models," *Neural Computation*, vol. 9, pp. 227–269, 1997.

[122] H. C. So, "LMS algorithm for unbiased parameter estimation of noisy autoregressive signals," *Electronics Letters*, vol. 37, no. 8, 2001.

[123] T. P. Speed and H. T. Kiiveri, "Gaussian Markov distributions over finite graphs," *Annals of Statistics*, vol. 14, no. 1, 1986.

[124] J. Spragins, "A note on the iterative application of Bayes' rule," *IEEE Transactions on Information Theory*, vol. 11, no. 4, 1965.

[125] P. Stoica and R. L. Moses, "On biased estimators and the unbiased Cramér-Rao lower bound," *Signal Processing*, vol. 21, pp. 349–350, 1990.

[126] P. Stoica and Y. Selén, "Cyclic minimizers, majorization techniques, and the expectation-maximization algorithm: A refresher," *IEEE Signal Processing Magazine*, Jan. 2004.

[127] G. Storvik, "Particle filters for state-space models with the presence of unknown static parameters," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002.

[128] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 605–612, 2003.

[129] E. B. Sudderth, M. J. Wainwright, and A. S. Willsky, "Embedded trees: Estimation of Gaussian processes on graphs with cycles," *IEEE Transactions on Signal Processing*, vol. 52, no. 11, 2004.

[130] B. Thiesson, D. M. Chickering, D. Heckerman, and C. Meek, "ARMA time-series modeling with graphical models," *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, pp. 552–560, 2004.

[131] S. Thrun, J. C. Langford, and D. Fox, "Monte Carlo hidden Markov models: Learning non-parametric models of partially observable stochastic processes," *International Conference on Machine Learning*, 1999.

[132] J. Vermaak, C. Andrieu, A. Doucet, and S. Godsill, "Non-stationary Bayesian modelling and enhancement of speech signals," Signal Processing Group, Department of Engineering, University of Cambridge, Tech. Rep., 1999.

[133] J. Vermaak, C. Andrieu, A. Doucet, and S. J. Godsill, "Particle methods for Bayesian modeling and enhancement of speech signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 3, 2002.

[134] P. O. Vontobel, "Algebraic coding for iterative decoding," Ph.D. dissertation, Signal and Information Processing Laboratory, ETH Zurich, 2003.

[135] ——, "Representations and updating of Gaussian messages in factor graphs," 2003, unpublished.

[136] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Computation*, vol. 12, pp. 1–41, 2000.

[137] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs," *IEEE Transactions on Information Theory*, vol. 42, no. 2, 2001.

[138] M. Welling and Y. W. Teh, "Linear response algorithms for approximate inference in graphical models," *Neural Computation*, vol. 16, pp. 197–221, 2004.

[139] G. Winkler, *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods.* Springer-Verlag, 2003.

[140] C. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1983.

[141] L. Xu and M. I. Jordan, "On convergence properties of the EM algorithm for Gaussian mixtures," *Neural Computation*, vol. 8, 1996.

[142] X.-S. Yang, "Comments on "on sequential estimation of state noise variance"," *IEEE Transactions on Automatic Control*, vol. 29, no. 8, 1984.

[143] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Bethe free energy, Kikuchi approximations and belief propagation algorithms," Mitsubishi Electric Research Laboratories, Tech. Rep. TR2001-16, 2001.

[144] ——, "Understanding belief propagation and its generalizations," in *Exploring Artificial Intelligence in the New Millennium.* Morgan Kaufmann, 2003.

[145] W. X. Zheng, "Estimation of the parameters of autoregressive signals from colored noise-corrupted measurements," *IEEE Signal Processing Letters*, vol. 7, no. 7, 2000.

[146] G. G. Zweig, "Speech recognition with dynamic Bayesian networks," Ph.D. dissertation, University of California, Berkeley, 1998.

Seite Leer /
Blank leaf

About the Author

Sascha Korl was born in Vienna, Austria, on 8 August 1974. He attended primary school in Feistritz/Drau and grammer school in Spittal/Drau. In 1993 he graduated with distinction from the College of Technology for Electronics Klagenfurt, Austria. Afterwards he joined the Graz University of Technology and the University of Music and Dramatic Arts in Graz, Austria, to study Electrical Engineering with specialisation in Sound Engineering. During his studies he gained working experience as software engineer and as freelancer for integrated circuit design. After a civilian service year in 2000/01 he started as research and teaching assistant at the Signal and Information Processing Laboratory (ISI) at the Swiss Federal Institute of Technology (ETH) in Zurich, Switzerland.

Series in Signal and Information Processing

edited by Hans-Andrea Loeliger

Vol. 1: Hanspeter Schmid, **Single-Amplifier Biquadratic MOSFET-C Filters.** ISBN 3-89649-616-6

Vol. 2: Felix Lustenberger, **On the Design of Analog VLSI Iterative Decoders.** ISBN 3-89649-622-0

Vol. 3: Peter Theodor Wellig, **Zerlegung von Langzeit-Elektromyogrammen zur Prävention von arbeitsbedingten Muskelschäden.** ISBN 3-89649-623-9

Vol. 4: Thomas P. von Hoff, **On the Convergence of Blind Source Separation and Deconvolution.** ISBN 3-89649-624-7

Vol. 5: Markus Erne, **Signal Adaptive Audio Coding using Wavelets and Rate Optimization.** ISBN 3-89649-625-5

Vol. 6: Marcel Joho, **A Systematic Approach to Adaptive Algorithms for Multichannel System Identification, Inverse Modeling, and Blind Identification.** ISBN 3-89649-632-8

Vol. 7: Heinz Mathis, **Nonlinear Functions for Blind Separation and Equalization.** ISBN 3-89649-728-6

Vol. 8: Daniel Lippuner, **Model-Based Step-Size Control for Adaptive Filters.** ISBN 3-89649-755-3

Vol. 9: Ralf Kretzschmar, **A Survey of Neural Network Classifiers for Local Wind Prediction.** ISBN 3-89649-798-7

Vol. 10: Dieter M. Arnold, **Computing Information Rates of Finite State Models with Application to Magnetic Recording.** ISBN 3-89649-852-5

Vol. 11: Pascal O. Vontobel, **Algebraic Coding for Iterative Decoding.** ISBN 3-89649-865-7

Vol. 12: Qun Gao, **Fingerprint Verification using Cellular Neural Networks.** ISBN 3-89649-894-0

Vol. 13: Patrick P. Merkli, **Message-Passing Algorithms and Analog Electronic Circuits.** ISBN 3-89649-987-4

Vol. 14: Markus Hofbauer, **Separation and Deconvolution of Acoustical Convolutive Mixtures.** ISBN 3-89649-996-3

Hartung-Gorre Verlag Konstanz – http://www.hartung-gorre.de