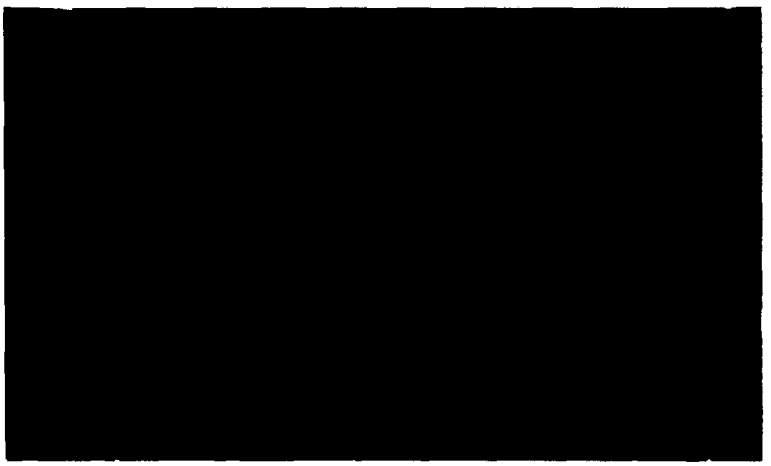
Bayesian linear regression for user-adaptive hearing aids

Özer, S.

Award date:
2007

U/e technische universiteit eindhoven



/ faculteit elektrotechniek

Eindhoven University of Technology
Department of Electrical Engineering
Signal Processing Systems

TU/e

Bayesian linear regression for user-adaptive hearing aids

by Serkan Özer

Master of Science thesis

Project period: 1-4-2006 - 29-8-2007
Report Number: 17-07

Commissioned by:
Prof.dr.ir. J.W.M. Bergmans

Supervisors:
Dr.ir. B. de Vries
Dr. A. Ypma

Additional Commission members:
Dr. T.M.H. Dijkstra

Preface

After graduating at the Avans Hogeschool in 's-Hertogenbosch, I started with my master study on Electrical Engineering at the Eindhoven University of Technology. This thesis is a conclusion of the work performed during my graduation project at this university in cooperation with GN-Resound in Eindhoven. I would like to thank the people from GNResound who gave me support and assistance. Without their help, I could not finish the project successfully.

At first, sincere gratitude goes to my supervisors: dr.ir. Bert de Vries and dr. Alexander Ypma. Their ideas and great enthusiasms made me enthusiastic in the machine learning world. Furthermore, I would like to thank the all members of the signal processing group. Also, I want to thank my colleagues at PT 3.31 for their great cooperation, but most of all for the nice working atmosphere.

Finally, I wish to express my thanks to my family and friends who have supported me during my time at the university. Especially my parents for giving me the chance to continue my study at the university.

Eindhoven, August 2007
Serkan Ozer

Abstract

Modern digital hearing aids are devices with a Digital Signal Processor (DSP) which allows implementation of advanced signal processing algorithms. These algorithms have a lot of parameters which should to be set to values that ideally match the preferences and the behavior of the user. The optimization procedure is a very complex task. As a result many parameters of a hearing aid algorithm are fixed by the manufacturer at a best guess value. A few of the user parameters are personalized by the hearing aid dispenser based on the nature of the hearing loss. Not every individual user preference can be put into the hearing aid: some particularities of the user may be hard to represent into the algorithm, the typical acoustic environments or user preference patterns may be changing.

We would like to personalize the hearing aid to the preferences of the user during usage. An adaptive model introduced in [Ypma et al., 2006] is used to learn the preferred parameters from the control operations of the user. In this report we have evaluated several methods for finding optimal linear models that can control the user preferred parameters in the adaptive hearing aid when the user preferences change. The underlying model that controls the adaptive hearing aid is based on the assumption that user preferred steering signal is linearly dependent on acoustic environmental features.

Under this assumption the project focused on analysis of a few properties of a multiple linear regression problem. This properties were *irrelevancy, redundancy, computational efficiency, probabilistic embedding* and *adaptivity.* These analysis leads to a deeper study for one specific algorithm, named Variational Bayesian Least Square (VBLS). One of the advantages of VBLS in comparison to other methods discussed in this report is its numerical robustness. This is due to the fact that VBLS requires no matrix inversions and has linear computational complexity in the number of dimensions d which results in an algorithm which scales easily to high dimensional data sets. The performance of VBLS was tested with experiments on artificial regression data and real data.

From our artificial regression data experiments, we concluded that VBLS does well at feature selection under the condition that a minimum number of samples is available for training. Data sets with large sample sizes and dimensions takes longer training time because of required computations increases. Therefore VBLS is a useful method for doing accurate regression and feature selection when we use it off-line. One further advantage of the VBLS method over the other analyzed methods is that one can compute a predictive distribution using the inferred parameter posteriors, allowing for a prediction with confidence levels. We can also extend VBLS algorithm into a sequential algorithm.

The VBLS method presented in this report is a suitable method that can handle with *irrelevancy, redundancy, computational efficiency, probabilistic embedding* and therefore it is widely applicable over a certain range of data sets in off-line usage. The required large training times for high dimensional data sets provides that the VBLS method is not efficient for on-line (adaptive) usage. For that reason a further research is necessary to the VBLS method in order to improve it on computational efficiency for sequential (on-line) usage.

Contents

Chapter 1

Introduction

1.1 Background

The work described in this report is the result of a research program on adaptive hearing aids started at GN Resound in co-operation with the Signal Processing Systems (SPS) group at Eindhoven University of Technology. The objective of the project was to evaluate methods for finding optimal linear models that can control the user preferred parameters in the adaptive hearing aid when the user preferences change. The underlying linear model that can controls the adaptive hearing aid will be based on the assumption that user preferences are linearly dependent on acoustic environmental features.

Because the vector of acoustic environmental features is likely high dimensional, computational complexity of our algorithms will be an issue in this report. The model input could possibly contain features that not contribute to the output. The optimal model parameters in the form of probabilistic distributions makes it possible to calculate a distribution over the output. Uncertainty in the output gives insight in the reliability of the model. The optimization method have to be applicable for a adaptive model. In this report different methods are considered in order to obtain an algorithm that can deal with above mentioned issues.

In section 1.2 we will give some information about hearing loss and hearing aids in order to get a better understanding of how hearing aids work. In section 1.3 the problem definition will be discussed in detail.

1.2 Hearing aids

Hearing loss is one of the most common conditions affecting many people in the world and can occur at any age. People with hearing loss have a difficult time believing, or accepting, that they have a hearing loss, because hearing loss often occurs gradually, and it's not always easy to notice at first.Persons with hearing loss discover their condition often from the reactions of others. According to Corwin [1998], one out of ten persons in US suffers from hearing loss. Hearing loss can be categorized by three classes:

- Type of hearing loss
- Degree of hearing loss
- Configuration of the hearing loss

There are three types of hearing losses. First one is conductive hearing loss. This type of loss occurs when the sounds in the outer or middle part of the ear become blocked and are not carried all the way to the inner ear. Conductive hearing loss is usually treatable with medical or surgical intervention. Sensorineural hearing loss is the second type of hearing loss and occurs when the auditory nerve or hair cells in the inner ear are damaged. Besides involving a reduction in sound level, sensorineural hearing loss also effects speech understanding or ability to hear clearly. Sensorineural hearing loss cannot be medically or surgically corrected. It is a permanent loss, but can be treated with hearing aids. In a few cases sensorineural hearing loss can occur in combination with conductive hearing loss. This is called mixed hearing loss. With mixed hearing loss, the conductive part may be treated, but the sensorineural part is usually permanent. Figure 1.1 shows a schematic overview of the ear, and shows in which parts of the ear the three types of hearing losses occur.

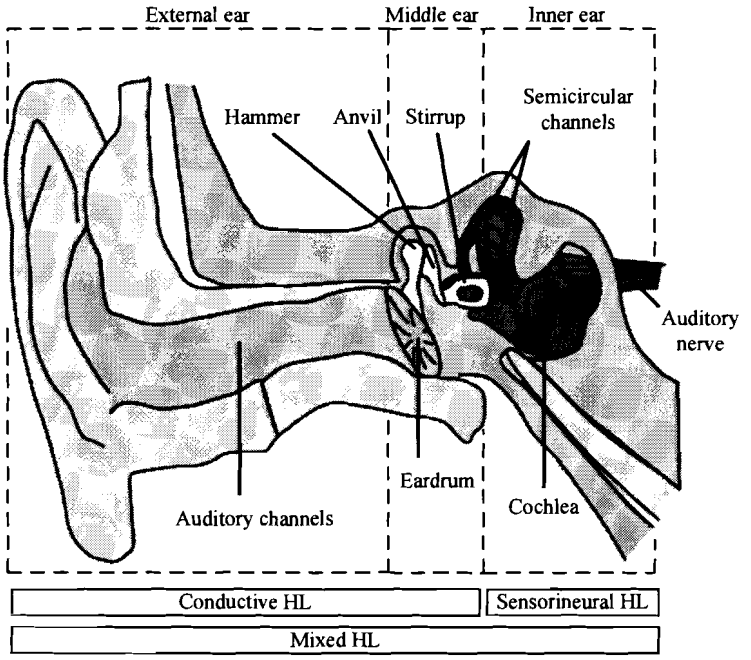


Figure 1.1: *Schematic overview of the ear. Conductive hearing loss is due to problems affecting sound transmission through the outer or middle ear. Sensorineural hearing loss arises in the inner ear. In mixed hearing loss both the middle and inner ear are involved.*

The degree of hearing loss refers to how large the loss is and is divided in six categories. A person who can hear sounds across a range of frequencies at 0 to 20 dB of Sound Pressure Level (SPL) is considered to have normal hearing. The level at which a person cannot hear a sound of a certain frequency, is known as their auditory threshold. A person who experiences hearing loss can have a range of tests, usually at an audiology clinic of the local hospital or health center. Hearing loss can be visualized with the help of a hearing test on an audiogram. An audiogram can show the hearing sensitivity for different frequencies at different intensities. In an audiogram the amount of the loss is measured in decibels of Hearing Loss (dB HL) at six frequencies: 250, 500, 1000, 2000, 4000 and 8000 Hz. A decibel is a logarithmic measurement of the intensity. Table 1.1 lists the thresholds for the six categories that generally range as follows. Physically, every 6 dB increase represents a doubling of sound pressure level.Perceptually, every 3 dB corresponds to doubling of sound intensity. Figure 1.2 shows an audiogram of a person with a moderate hearing loss.

Table 1.1: *Degree of hearing loss.*

Hearing threshold(dB)	Degree of hearing loss	Ability to hear speech
0-25	None	No significant difficulty.
26-40	Mild	Difficulty hearing soft speech and conversations.
41-55	Moderate	Difficulty understanding conversational speech, especially when there is background noise. Higher volume levels are required for hearing TV and radio.
56-70	Moderate to Severe	Clarity of speech is significantly affected. Speech must be loud and you may have difficulty in group conversations.
71-90	Severe	Normal conversational speech is inaudible. You may also have difficulty with loud speech or only be able to understand shouted or amplified speech.
91+	Profound	Unable to clearly understand even amplified speech.

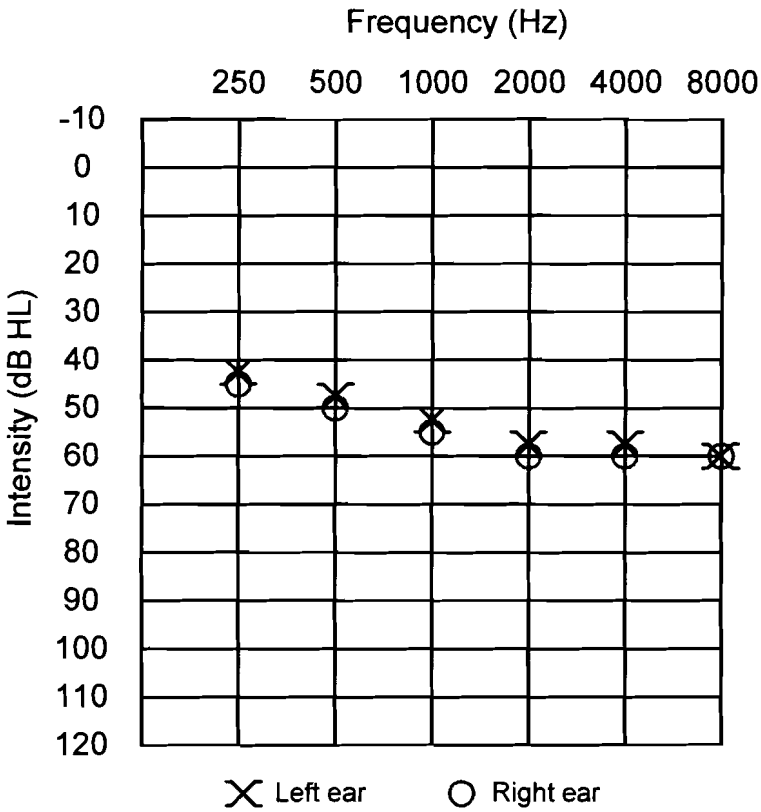


Figure 1.2: *Audiometric thresholds ranging from 40-60 db HL constitute a moderate hearing loss.*

The configuration or shape of the hearing loss refers to the extent of hearing loss at each frequency and the overall picture of hearing that is created. A hearing loss that only affects the high frequencies would be described as a high-frequency loss. Its configuration would show good hearing in the low frequencies and poor hearing in the high frequencies. On the other hand, if only the low frequencies are affected, the configuration would show poorer hearing for low tones and better hearing for high tones. Some hearing loss configurations are flat, indicating the same amount of hearing loss for low and high tones. As described before sensorineural hearing loss is a permanent loss and cannot be medically or surgically corrected. In permanent hearing loss, hearing aids can often improve how well patients hear and communicate.

Hearing aids are able to restore the audibility by using compressive amplification. Compressive amplification amplifies the weak sounds more than the strong ones, so that a wide range of input signals is compressed into a smaller range at the output. There are many types of hearing aids and they have mostly similar basic parts. All hearing aids consist of three basic components. The first component is the microphone that picks up sounds from the air and converts them to electrical signals. The amplifier increases the intensity of the signals from the microphone which are relevant for the person. Third component is the receiver. It converts electrical signals into acoustic signals.

Furthermore, in digital hearing aids a DSP (Digital Signal Processor) can be programmed to manipulate the signals to fit the hearing loss of the hearing impaired person. Some hearing aids are equipped with volume controls and other control functions, which can be used for individual adjustments. Hearing aids are available in many different types. Figure 1.3 shows the most popular types of hearing aids.

Figure 1.3: *The types of hearing aids can be divided into four categories.*

- **Completely in the canal (CIC):** The smallest hearing aid which is almost invisible in the ear. These hearing aids are restricted to people with ear canals large enough to accommodate the whole hearing aid.
- **In the canal (ITC):** This hearing aid is a bit larger than the completely in the canal type. This kind of hearing aid is normally used in cases of mild to moderate hearing loss.
- **In the ear (ITE):** In the ear hearing aid are larger, and therefore have more features. It can also accommodate larger sound amplifiers. This type of hearing aid is also easier to accommodate than the previous two.
- **Behind the ear (BTE):** The electronics are housed in a case that is fitted behind the ear. Because of their robust design, this type is especially recommended for children.

The modern digital hearing aid can provide a hearing-impaired person with an improved and more pleasant sound picture, but it cannot fully bring back normal hearing.

1.3 Problem definition

1.3.1 Problem context

Modern digital hearing aids are devices with a Digital Signal Processor (DSP) which allows implementation of advanced signal processing algorithms. These algorithms have a lot of parameters which should to be set to values that ideally match the preferences and the behavior of the user. Because of the possibly large dimensionality of the parameter space and unknown user satisfaction determinants, the optimization procedure becomes a very complex task. As a result many parameters of a hearing aid algorithm are fixed by the manufacturer at a best guess value. A few of the user parameters are personalized by the hearing aid dispenser based on the nature of the hearing loss. Not every individual user preference can be put into the hearing aid: some particularities of the user may be hard to represent into the algorithm, the typical acoustic environments or user preference patterns may be changing. Therefore one would like to personalize the hearing aid to the preferences of the user during usage.

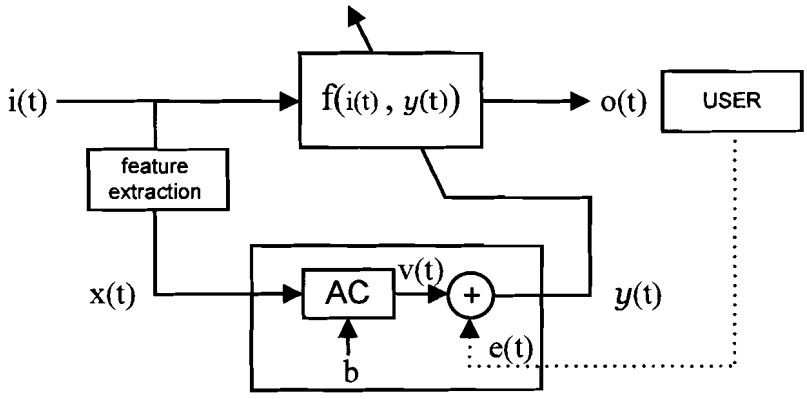


Figure 1.4: *Hearing aid personalization from user control*

An adaptive model introduced in Ypma et al. [2006] is able to learn the user preferred parameters from the control operations of the user. The situation is illustrated in figure 1.4. The hearing aid performs as a sound processing function $f(i_t, y_t)$ where y_t is vector for tuning parameters. In a conventional hearing aid tuning parameters are fixed at a certain value. The feedback signal e_t represents the adjustments from the user e.g. through a control wheel or button on the hearing aid. In this adaptive learning model the user feedback e_t is absorbed by a parameter vector $\mathbf{b}$ in order to obtain smaller corrections from the user in the future. The Automatic Control (AC) unit takes input $\mathbf{x_t}$, which holds a vector of acoustic features from the input signal i_t. For instance $\mathbf{x_t}$ could hold power (RMS) and Signal to Noise Ratio (SNR) estimates of signal i_t from the acoustic environment. The output v_t is a prediction of the user preferred parameter value y_t. In this model an assumption is made that the prediction of the user preferred parameter vector y_t is linearly dependent on the acoustic environment through a set of acoustic features $\mathbf{x_t}$ given by

$$y_t = \mathbf{x_t}\mathbf{b} + e_t \tag{1.1}$$

When the user start with adjustment, we will accept that he is not content at that moment (dissent moment). A learning moment occurs right after the user has stopped with adjustment. At this time the user has found satisfying tuning parameters; we will call this the consent moment. The

assumptions that we have made is only valid at consent moments. The index t in equation (1.1) is replaced by index n for consent moments and is given by.

$$y_n = \mathbf{x_n}\mathbf{b} + e_n \tag{1.2}$$

where the user corrections e_n are assumed to be independent and identically distributed (i.i.d.). Now the described link between y_n and $\mathbf{x_n}$ can be seen in the context of a multiple linear regression problem. We will assume to have acquired data set $D = \{(\mathbf{x_n}, y_n)\}_{n=1}^{N}$ where n denote each consent moment. In multiple linear regression model the target variable y is a function of d input variables $x_1, ..., x_d$ and is given by

$$y_n = \mathrm{b}_1 x_{n1} + \mathrm{b}_2 x_{n2} + ... + \mathrm{b}_d x_{nd} + e_n \quad n = 1, ..., N \tag{1.3}$$

where e_n's are i.i.d. random variables with zero mean and constant variance σ^2 for all $n = 1, ..., N$. We will describe multiple linear regression model in detail in Chapter 2.

1.3.2 Problem statement

The goal of this research is to evaluate methods for finding the optimal model parameters $\mathbf{b}$ for the multiple linear regression model that can explain user preferred parameter values y_n from a possibly high dimensional input $\mathbf{x_n}$. But, there are some issues that can complicate the problem:

1. **Irrelevancy:** The high dimensional input $\mathbf{x_n}$ could possibly contain a lot of irrelevant features. Irrelevant features are those that do not contribute to predict the desired output.
2. **Redundancy:** The input $\mathbf{x_n}$ could also contain redundant features which are correlated to other features.
3. **Computational efficiency:** The high dimensional behavior of the input $\mathbf{x_n}$ makes optimization procedure a complex task. The numerical solutions of the matrix inversions become expensive by increasing number of inputs.
4. **Probabilistic embedding:** GN Resound prefers probabilistic models because in this report determined probabilistic model can be used to weight this with other probabilistic models determined before by GN Resound scientific staff in order to get most promising one.
5. **Adaptivity:** The optimization procedure may have to occur during the usage of hearing aid. This means that the used method has to be applicable for sequential processing.

To construct the best model that can predict the user preferred parameters y_n, we analyzed several methods in order to get one that can handle all mentioned points above. After analyzing of different methods we have chosen for the most promising one. First, chosen method is reported in detail to get more insight on, second, the performance of the chosen method is tested on synthetic an real world data to validate the model.

1.4 Report outline

In order to construct the best model, several steps have to be performed in this report. This report has been split up in following parts:

- **Literature study:** In chapter 2 some literature study will be reported. We will discuss linear models, adaptive linear models and dimensionality reduction for regression.
- **Variational Bayesian Least Squares:** In chapter 3 we will construct the Variational Bayesian Least Squares method from an existing method called Backfitting. First we will construct the probabilistic backfitting method after that Variational Bayesian Least Squares method will be introduced from the probabilistic version of backfitting.

- **Experiments:** In chapter 4 the performance of VBLS will be tested with experiments on artificial regression data and real data. A real time platform is used in order to obtain real data. This platform contains a virtual hearing aid interface with an adjustable gain and noise. This interface can register user behavior in a certain acoustic environment and can collect the corresponding parameter values in a database.
- **Conclusions and recommendations:** Finally, chapter 5 will summarize the conclusions of the project. This chapter will also provide some recommendations for further research.

Chapter 2

Linear models for regression

We have to evaluate different models in order to find most promising one that can handle with all mentioned points in subsection 1.3.2. Therefore first the underlying theory has to be understood. However, not all parts of the underlying theory will be dealt with in detail. Instead, only the relevant parts will be discussed in this chapter. Section 2.1 deals with general topics in machine learning. Section 2.2 describes the simple linear regression and the multiple linear regression models. In this section we will also explain an regression method called Least Square (LS). In section 2.3 adaptive linear models Least Mean Square (LMS) and Recursive Least Square(RLS) are explained in detail. In the last section 2.4 dimensionality reduction for regression will be explained.

2.1 Machine learning

Learning is a broad concept that it is very difficult to define precisely [Catania, 1992]; [Hergenhahn, 1988]; [Mazur, 1990]. According to Webster [1999] the term learning is phrased as "to gain knowledge, or understanding of, or skill in, by study, instruction or experience". For a long time, scientists in biology study learning in animals and humans to get understanding of it. This report focuses on learning in machines. A machine learns whenever it changes its structure, program or data (input/output) in such a manner that its expected future performance improves. Machine learning usually refers to the changes in systems that perform tasks like recognition, identification and diagnosis and robot control. Why is machine learning so important? There are several reasons that make machine learning important.

- A system with unknown transfer function: If we can specify input/output pairs but not the relation between the inputs and outputs. We need machines that can adjust their internal structure to produce correct outputs (classification or regression).
- It is possible that hidden among large piles of data are important relationships and correlations. Machine learning can be used to extract this relationship (data mining).
- The amount of knowledge available about certain task might be too large for explicit encoding by humans. A machine that learns this knowledge might be able to capture the essence of it(compression).
- For many real world problems there is uncertainty about the variables in the system and their structure. A way to combine prior expectations with actual (but perhaps noisy) measurements is crucial for learning meaningful models.

There are different kinds of computational structures that can be learned. Some of these computational structures are functions, logic programs, finite-state machines. In this report we put more emphasis on learning functions. There are two major settings in which we wish to learn a

function. These major settings can be divided in supervised and unsupervised learning.

Supervised learning is a machine learning technique for learning a (perhaps nonparametric)functional form for the process that may have generated a set of training data. The training data consist of inputs and desired output vectors, the targets. If the output of the function for a given input has a continuous value then the used algorithm is called Regression. Another task in supervised learning is called Classification. With classification methods the output of the function can predict a class label of the input vector. The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples.

Unsupervised learning is used for problems where a model is fit to observations, but there is no a target output. Unsupervised learning treats input vectors as a draw from a set of random variable therefore for the data set a joint density model for data and other model variables (e.g. the generating variables) is built. Examples of unsupervised learning: Clustering and Dimensionality Reduction. Clustering is the process of assigning data to different groups, such that all data points in a group share same common properties. Dimensionality reduction is e.g. used by compression. In this method the number of variables with which a set of input vectors is described is decreased while still retaining most of the predicting power for the outcome.

In this report supervised learning is used in order to find the best model that can handle in subsection 1.3.2 mentioned five points. As the learning method we chose for multiple linear regression model.

2.2 Linear models

Linear regression models are statistical models which are able to make predictions for the target variable Y given some new value of the input variable X. A set of training data that contains N observations from $\mathrm{Y} = (y_1, ..., y_N)$ and $\mathrm{X} = (x_1, ..., x_N)$ are used to learn the underlying functional mapping with a model that is linear in the regression parameters to predict the new values of X. Although linear regression models are simple models they are very important because, these models are able to transform underlying non-linear relationships between the target variables Y and the input variables X to a new pair of variables with a linear relationship.

2.2.1 Simple linear regression model

The relationship between target variables Y and the input variables X can be described as a linear function and some random fluctuations. Suppose that we have a target variable Y and an input variable X then the simple linear regression model is given by

$$y_n = \mathrm{b}x_n + \epsilon_n \quad n = 1, ..., N \tag{2.1}$$

where b is the unknown coefficient and ϵ_n's are independent and identically distributed (iid) random variables with zero mean and constant variance σ^2 for all $n = 1, ..., N$. The line $\mathrm{b}x_n$ in (2.1)is called linear predictor and the terms ϵ_n are random errors. The random error $\epsilon_n = y_n - \mathrm{b}x_n$ is the term which accounts for the variation of the n th target variable y_n away from the linear predictor $\mathrm{b}x_n$ at the point x_n. Now we can express our uncertainty over the value of the target variable using a probability distribution. Assume that, given the value of x, the corresponding value of y has a Gaussian distribution with mean value $\mathrm{b}x$ and variance σ^2. We obtain the following conditional distribution given by

$$p(y|x, \mathrm{b}, \sigma^2) = \mathcal{N}(y|\mathrm{b}x, \sigma^2) \tag{2.2}$$

The Gaussian conditional distribution for y given x is illustrated schematically in Figure 2.1. We have now two unknown parameters b and σ^2 and we want to estimate them using the given

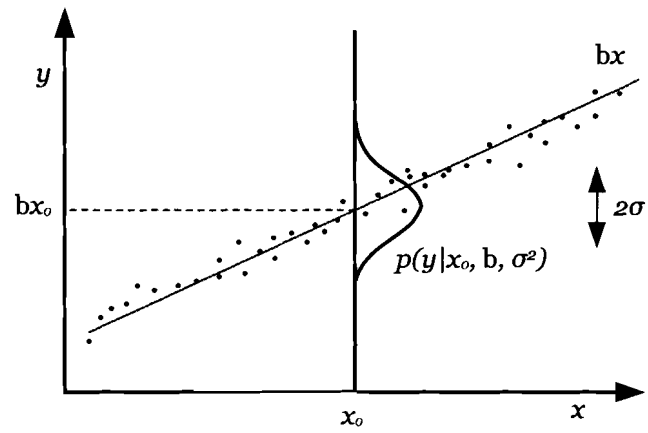


Figure 2.1: Gaussian conditional distribution for y given x. The mean is given by linear predictor bx and the precision is given by variance σ^2.

training data set (X,Y). For the determination of values of the unknown parameters maximum likelihood is used. If we assume that data from the distribution (2.2) is drawn independently then the likelihood function is given by

$$p(\mathrm{Y}|\mathrm{X},\mathrm{b},\sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y_n|\mathrm{b}x_n,\sigma^2) \tag{2.3}$$

The unknown parameter values can now be determined by maximizing the logarithm of the likelihood function. If we take the logarithm of both sides of the equation (2.3) we obtain the following log likelihood function in the form

$$\ln p(\mathrm{Y}|\mathrm{X},\mathrm{b},\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \mathrm{b}x_n)^2 - \frac{N}{2}\ln\sigma^2 \tag{2.4}$$

The unknown coefficient is determined by maximizing (2.4) with respect to b. The coefficients that maximize the log-likelihood function, will be denoted by b_{ML}. We can also use maximum likelihood to determine the variance of the Gaussian conditional distribution by maximizing (2.4) with respect to σ^2. The solutions for b_{ML} and σ^2_{ML} are given by

$$\begin{aligned} \mathrm{b}_{\mathrm{ML}} &= \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2} \\ \sigma^2_{\mathrm{ML}} &= \frac{1}{N}\sum_{n=1}^{N}(y_n - \mathrm{b}_{\mathrm{ML}}x_n)^2 \end{aligned} \tag{2.5}$$

Now the determined parameters b_{ML} and σ^2_{ML} can be used for making predictions for new values of x^*. We may assign to our probabilistic model a predictive distribution that gives the probability distribution over y^*. Substituting the maximum likelihood parameters into (2.2) gives

$$p(y^*|x^*,\mathrm{b}_{\mathrm{ML}},\sigma^2_{\mathrm{ML}}) = \mathcal{N}(y^*|\mathrm{b}_{\mathrm{ML}}x^*,\sigma^2_{\mathrm{ML}}) \tag{2.6}$$

In this subsection the simple linear regression model has been discussed. We have described a method, based on maximum likelihood, for fitting simple linear regression models to data.

2.2.2 Multiple linear regression model

In this subsection multiple linear regression models will be discussed by extending the concept of simple linear regression models to multiple linear regression models. In multiple linear regression model the target variable y is a function of d input variables $(x_1, ..., x_d)$. This relationship can be written as

$$y_n = \mathrm{b}_1 x_{n1} + \mathrm{b}_2 x_{n2} + ... + \mathrm{b}_d x_{nd} + \epsilon_n \quad n = 1, ..., N \tag{2.7}$$

where ϵ_n's are i.i.d. random variables with zero mean and constant variance σ^2 for all $n = 1, ..., N$. Because of the multidimensional behavior of the linear regression model, the parameter estimation expressions become complex and tedious to write out. Therefore it is customary to use a matrix notation that simplifies the notation of the multidimensional linear regression model. Let y_n denote the n th observation and $\mathbf{y} = [y_1, ..., y_N]^T$ the column vector containing the y_n's. Further $\mathbf{x}_n = [x_{n1}, ..., x_{nd}]$ denotes d dimensional row vector which contains all values of input variable $\mathbf{x}$ for the n th observation. If we stack the N row vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$, we get the design matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]^T$. Now we can express the multiple linear regression model in matrix form as

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \boldsymbol{\epsilon} \tag{2.8}$$

where $\mathbf{b} = (\mathrm{b}_1, \mathrm{b}_2, ..., \mathrm{b}_d)^T$ is the parameter vector and $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, ..., \epsilon_N)^T$ is column vector containing N random variables. We assume all ϵ_n's are i.i.d. distributed as $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$. Our goal is to estimate the optimal linear coefficients $\mathbf{b} = (\mathrm{b}_1, \mathrm{b}_2, ..., \mathrm{b}_d)^T$ which combine the input dimensions to produce the output y_n given the observed data set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$. Figure 2.2 shows the graphical model for multiple linear regression which corresponds to the the generative model given by equation (2.8).

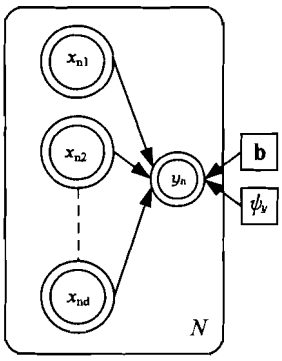


Figure 2.2: Graphical model for multiple linear regression.

We now return to the Gaussian conditional distribution for y given $\mathbf{x}$. It is easy to see that under our graphical model, the estimation of the optimal regression parameters can be done by maximum likelihood, where the corresponding likelihood function is given by

$$p(y|\mathbf{x}, \mathbf{b}, \sigma^2) = \mathcal{N}(y|\mathbf{x}\mathbf{b}, \sigma^2) \tag{2.9}$$

where $\mathbf{x}\mathbf{b} = \mathrm{b}_1 x_1 + \mathrm{b}_2 x_2 + ... + \mathrm{b}_d x_d$ and σ^2 is the variance of the likelihood function. In order to find the maximum likelihood solution we first need to maximize $\ln p(\mathbf{y}|\mathbf{X}, \mathbf{b}, \psi_y)$ (log likelihood function) with respect to $\mathbf{b}$ and ψ_y where $\psi_y = \sigma^2 I$. If we assume that data from (2.9) is i.i.d. then the log likelihood function is given by

$$\ln p(\mathbf{y}|\mathbf{X}, \mathbf{b}, \psi_y) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n \mathbf{b})^2 - \frac{N}{2} \ln \sigma^2 \tag{2.10}$$

Maximizing the log likelihood function with respect to $\mathbf{b}$ and σ^2 gives the following maximum likelihood solutions.

$$\mathbf{b}_{\mathrm{ML}} = \frac{\sum_{n=1}^{N} \mathbf{x}_n^T y_n}{\sum_{n=1}^{N} \mathbf{x}_n^T \mathbf{x}_n} \qquad (2.11)$$

$$\sigma^2_{\mathrm{ML}} = \frac{1}{N} \sum_{n=1}^{N} (y_n - \mathbf{x}_n \mathbf{b}_{\mathrm{ML}})^2$$

The determined parameters $\mathbf{b}_{\mathrm{ML}}$ and $\psi_{y(\mathrm{ML})} = \sigma^2_{\mathrm{ML}} I$ can assign a predictive distribution to our multiple linear regression graphical model (figure 2.2) that gives the probability distribution over y^* given new values of $\mathbf{x}^*$. Substituting the maximum likelihood parameters into (2.9) gives

$$p(y^* | \mathbf{x}^*, \mathbf{b}_{\mathrm{ML}}, \sigma^2_{\mathrm{ML}}) = \mathcal{N}(y^* | \mathbf{x}^* \mathbf{b}_{\mathrm{ML}}, \sigma^2_{\mathrm{ML}}) \qquad (2.12)$$

In this subsection we have described the maximum likelihood method for fitting multiple linear regression graphical models to data. Another linear regression method called Least Squares (LS) will be explained in detail in the next subsection.

2.2.3 Least squares

The method of least squares assumes that the best-fit line of a given type is the line that has the minimal sum of the deviations squared (least square error) from a given set of data. From a given data set (x_1, y_1),(x_2, y_2),...,(x_N, y_N) we can determine the best fitting line $f(x)$ that has the minimum error e from each data point, i.e., $e_1 = y_1 - f(x_1)$, $e_2 = y_2 - f(x_2)$,..., $e_N = y_N - f(x_N)$ where x is the input variable and y is the target variable. The fitting line $f(x)$ is controlled by one independent parameter b as $f(x) = bx$. According to the the method of least squares, the best fitting line has the minimum value for sum of squares error (SSE) given by

$$SSE = e_1^2 + e_1^2 + ... + e_N^2 = \sum_{n=1}^{N} e_n^2 = \sum_{n=1}^{N} [y_n - f(x_n)]^2 \qquad (2.13)$$

Multiple regression least squares estimates the outcomes which may be affected by more than one independent parameter $(b_1, b_2, ...)$. An example is d input variables $x_1, x_2, ..., x_d$ and one target variable y which is given by

$$y = b_1 x_1 + b_2 x_2 + ... + b_d x_d \qquad (2.14)$$

For a given data set $(x_{11}, x_{12}, ..., x_{1d}, y_1), (x_{21}, x_{22}, ..., x_{2d}, y_2), ..., (x_{N1}, x_{N2}, ..., x_{Nd}, y_N)$, where $N \geq d$, the best fitting line $f(x)$ has the least square error given by

$$SSE = \sum_{n=1}^{N} [y_n - f(x_{n1}, x_{n2}, ..., x_{nd})]^2 = \sum_{n=1}^{N} [y_n - (b_1 x_{n1} + b_{n2} x_2 + ... + b_d x_{nd})]^2 \qquad (2.15)$$

The coefficients $(b_1, b_2, ..., b_d)$ are unknown while all $x_{n1}, x_{n2}, ..., x_{nd}$ and y_n are given for $n = 1, ..., N$. To obtain the least square error, the unknown coefficients $(b_1, b_2, ..., b_d)$ must yield zero first derivatives and is given by

$$
\begin{aligned}
\frac{\partial SSE}{\partial b_1} &= 2\sum_{n=1}^{N} x_{n1}[y_n - (b_1 x_{n1} + b_2 x_{n2} + ... + b_d x_{nd})] = 0 \\
\frac{\partial SSE}{\partial b_2} &= 2\sum_{n=1}^{N} x_{n2}[y_n - (b_1 x_{n1} + b_2 x_{n2} + ... + b_d x_{nd})] = 0 \\
&\cdot \\
&\cdot \\
&\cdot \\
\frac{\partial SSE}{\partial b_d} &= 2\sum_{n=1}^{N} x_{nd}[y_n - (b_1 x_{n1} + b_2 x_{n2} + ... + b_d x_{nd})] = 0
\end{aligned}
\tag{2.16}
$$

After expanding of equations (2.16) we end up with d linear equations (2.17) which can used to calculate the unknown coefficients $(b_1, b_2, ..., b_d)$.

$$
\begin{aligned}
\sum_{n=1}^{N} x_{n1} y_n &= b_1 \sum_{n=1}^{N} x_{n1}^2 + b_2 \sum_{n=1}^{N} x_{n1} x_{n2} + ... + b_d \sum_{n=1}^{N} x_{n1} x_{nd} \\
\sum_{n=1}^{N} x_{n2} y_n &= b_1 \sum_{n=1}^{N} x_{n2} x_{n1} + b_2 \sum_{n=1}^{N} x_{n2}^2 + ... + b_d \sum_{n=1}^{N} x_{n2} x_{nd} \\
&\cdot \\
&\cdot \\
&\cdot \\
\sum_{n=1}^{N} x_{nd} y_n &= b_1 \sum_{n=1}^{N} x_{nd} x_{n1} + b_2 \sum_{n=1}^{N} x_{nd} x_{n2} + ... + b_d \sum_{n=1}^{N} x_{nd}^2
\end{aligned}
\tag{2.17}
$$

We can express these linear equations using input matrix $\mathbf{X}$, target vector $\mathbf{y}$ and parameter vector $\mathbf{b}$. The nth row of $\mathbf{X}$ and $\mathbf{y}$ will contain the x and y value for the nth data sample. The calculated maximum likelihood solutions (2.11) and least square solutions (2.17) for coefficient vector $\mathbf{b}$ end up with the same solution and can be written in the form of input matrix $\mathbf{X}$ and target vector $\mathbf{y}$ given by

$$
\mathbf{b}_{\text{OLS}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{2.18}
$$

Least squares solutions are calculated by fitting a regression line to the points in a plot. The line is formed by regressing the data on the transformed percent. Maximum likelihood solutions are calculated by maximizing the likelihood function. The likelihood function describes for each set of distribution parameters the chance that the true distribution has these parameters based on the sample.

It is easy to see that the solution for optimal estimate of parameters $\mathbf{b}$ is given by equation (2.18) and called Ordinary Least Square (OLS) solution. Increasing the input dimensionality d in the graphical model makes required matrix inversion $(\mathbf{X}^T \mathbf{X})^{-1}$ computationally instable (approximately $O(d^3)$). The determinant of matrix multiplication $(\mathbf{X}^T \mathbf{X})$ will be zero if there are redundant dimensions which makes matrix inversion also instable. According to Ting et al. [2006] there are efficient methods for matrix inversions, but these methods are unable to model noise in input data and require manual tuning of parameters, which can strongly influence the quality of the estimation results. Solution (2.18) can be improved by several existing methods such as stepwise regression [Draper and Smith, 1981] and partial least square regression (PLS) [Wold, 1975] but these methods may suffer from overfitting. The overfitting problem can be diminished by estimating regression parameters $\mathbf{b}$ in a Bayesian framework. This has the advantages that

confidence intervals can be produced for parameter estimates without the use of asymptotic approximations and prior information can be incorporated into the analysis. In the next chapter we will describe a linear regression method in Bayesian framework in detail. Now adaptive linear models Least Mean Squares (LMS) and Recursive Least Squares (RLS) are described in detail in the next section.

2.3 Adaptive linear models

Adaptive algorithms are very important for Digital Signal Processing (DSP). They are used in different applications, acoustic echo cancelation, radar guidance systems, and wireless channel estimation. Adaptive algorithms are used to estimate a time varying signals. In the literature there are many adaptive algorithms such as Recursive Least Square (RLS), Kalman filtering and Least Mean Square (LMS). In this section we will only describe the most commonly used algorithms namely, LMS in the first subsection and RLS in the second subsection.

2.3.1 Least Mean Square (LMS)

The Least Mean Square (LMS) algorithm, introduced by Widrow and Hoff in 1959 [Widrow and Hoff, 1960] is an adaptive algorithm, which uses a gradient-based method of steepest decent [Haykin, 1985]. LMS algorithm is used in adaptive filters to find the filter coefficients by taking the least mean squares of the difference between the desired and the actual signal (error signal). The gradient descent method minimize the error signal by adjusting the filter coefficients. This is done by taking steps in the direction negative of the gradient. Linear adaptive filter with LMS algorithm is formulated using a block diagram shown in figure 2.3.

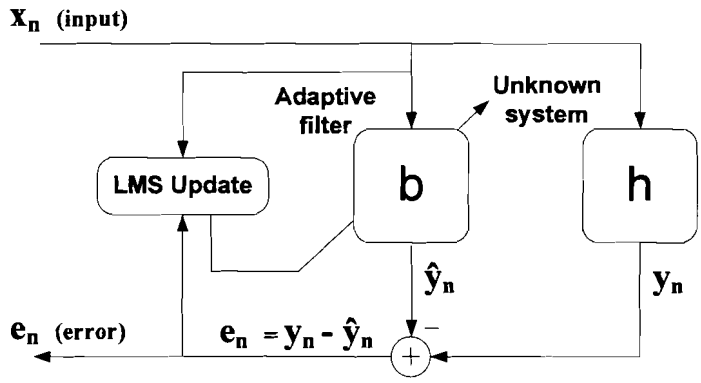


Figure 2.3: Linear adaptive filter with Least Mean Square algorithm.

An unknown system h has to be identified by adapting the filter **b** to make it as close as possible using observable signals $\mathbf{x}_n$, $\mathbf{y}_n$ and $\mathbf{e}_n$ which eventually leads to the minimum mean square error. For that purpose we use the method of steepest descent to find the weight vector **b**, the weight vector equation is given by

$$\mathbf{b}_{n+1} = \mathbf{b}_n + \frac{1}{2}\mu[-\nabla(E\{\mathbf{e}_n^2\})] \tag{2.19}$$

Where μ is the step-size parameter and controls the convergence characteristics of the LMS algorithm; $E\{\mathbf{e}_n^2\}$ is the mean square error between the desired signal $\mathbf{y}_n$ and actual signal $\hat{\mathbf{y}}_n$ given by formula

$$E\{\mathbf{e}_n^2\} = E\{[\mathbf{y}_n - \hat{\mathbf{y}}_n]^2\} = E\{[\mathbf{y}_n - \mathbf{b}^T\mathbf{x}_n]^2\} \tag{2.20}$$

We can minimize the mean square error term in equation (2.19) by taking the partial derivatives with respect to the individual entries of the filter coefficient vector $\mathbf{b}$. The computed gradient vector is given by

$$\nabla_{\mathbf{b}}(E\{\mathbf{e}_n^2\}) = -2\mathbf{r} + 2\mathbf{R}\mathbf{b}_n \tag{2.21}$$

In the method of steepest descent it is difficult to find values for $\mathbf{r}$ and $\mathbf{R}$ matrices in real time. The LMS algorithm prevent this problem by computing the instantaneous values of covariance matrices $\mathbf{r}$ and $\mathbf{R}$ given by

$$\begin{aligned} \mathbf{r}_n &= \mathbf{y}_n\mathbf{x}_n \\ \mathbf{R}_n &= \mathbf{x}_n\mathbf{x}_n^T \end{aligned} \tag{2.22}$$

If we substitute the computed gradient vector into equation (2.19), we obtain the following formula for weight update

$$\mathbf{b}_{n+1} = \mathbf{b}_n + \mu\mathbf{x}_n[\mathbf{y}_n - \mathbf{x}_n^T\mathbf{b}_n] = \mathbf{b}_n + \mu\mathbf{x}_n\mathbf{e}_n \tag{2.23}$$

The LMS algorithm is initiated with an arbitrary value $\mathbf{b}_0$ for the weight vector at $n = 0$. The successive corrections of the weight vector eventually leads to the minimum value of the mean squared error [Shetty, 2004]. The LMS algorithm can be summarized in following equations given by

$$\begin{aligned} \textit{Output:}\quad \mathbf{y}_n &= \mathbf{b}^T\mathbf{x}_n \\ \textit{Error:}\quad \mathbf{e}_n &= \mathbf{y}_n - \hat{\mathbf{y}}_n \\ \textit{Weight:}\quad \mathbf{b}_{n+1} &= \mathbf{b}_n + \mu\mathbf{x}_n\mathbf{e}_n \end{aligned} \tag{2.24}$$

The LMS algorithm initiated with some arbitrary value for the weight vector will converge after number of iterations. The stability criterion for LMS algorithm is that step-size parameter μ must have values between zero and $1/\lambda_{max}$ ($0 < \mu < 1/\lambda_{max}$). λ_{max} is the largest eigenvalue of the correlation matrix $\mathbf{R}$. The convergence of the algorithm is inversely proportional to the eigenvalue spread of the correlation matrix $\mathbf{R}$. The eigenvalue spread of the correlation matrix is estimated by computing the ratio of the largest eigenvalue to the smallest eigenvalue of the matrix. R with widespread eigenvalues causes a slow converge. If μ is chosen to be very small then the algorithm converges very slowly. A large value of μ may lead to a faster convergence but may be less stable around the minimum value.

The LMS algorithm is most commonly used adaptive algorithm because of its simplicity and a reasonable performance. LMS algorithm is an adaptive algorithm therefore it can be used in a highly time-varying signal environment. It has a stable and robust performance against different signal conditions. Beside all mentioned advantages LMS has also shortcomings, it may not have a really fast convergence speed compared other complicated algorithms like the Recursive Least Square (RLS).

2.3.2 Recursive Least Square (RLS)

The Recursive Least Squares (RLS) algorithm is another algorithm for determining the coefficients of an adaptive filter. If we compare RLS with LMS algorithm, the RLS algorithm uses information from all past input samples (LMS uses only the current input samples) to estimate the inverse of the autocorrelation matrix $\mathbf{R}$ of the input vector. To decrease the influence of input samples from the far past, a weighting factor for the influence of each sample is used. Now we will introduce in a cost function J the weighting factor ρ and is given by formula

$$J_n = \sum_{i=1}^{n} \rho^{n-i} E\{\mathbf{e}_{i,n}^2\} \tag{2.25}$$

where the error signal $\mathbf{e}_{i,n}$ is computed for all times $1 \leq i \leq n$ using the current filter coefficients $\mathbf{b}_n$ and is given by

$$\mathbf{e}_{i,n} = \mathbf{y}_i - \mathbf{b}_n^T \mathbf{x}_i \tag{2.26}$$

If we choice for $\rho = 1$ the squared error for all sample times up to current time is considered in the cost function equally. If $0 \leq \rho \leq 1$ the influence of past error values decays exponentially. In this special case the weighting factor ρ is called *forgetting factor*. Instead of the gradient of the mean square error function (this is done by LMS) we will compute the gradient of the cost function J with respect to filter coefficient vector $\mathbf{b}$. In RLS cost function J is a weighted error function with weighting factor ρ. The gradient of the mean cost function J is given by

$$\begin{aligned} \nabla_{\mathbf{b}} J_n &= \textstyle\sum_{i=1}^{n} \rho^{n-i} E\{-2\mathbf{x}_i[\mathbf{y}_i - \mathbf{b}_n^T \mathbf{x}_i]\} \\ \nabla_{\mathbf{b}} J_n &= \textstyle\sum_{i=1}^{n} \rho^{n-i} (-2E\{\mathbf{y}_i \mathbf{x}_i\} + 2E\{\mathbf{x}_i \mathbf{x}_i^T\} \mathbf{b}_n^T) \end{aligned} \tag{2.27}$$

We now do not use a gradient descent method, but immediately search for the minimum of the cost function J_n by setting its gradient to zero. The resulting equation for the optimum filter coefficients at time n is

$$\begin{aligned} \Phi_n \mathbf{b}_n &= \mathbf{z}_n \\ \mathbf{b}_n &= \Phi_n^{-1} \mathbf{z}_n \end{aligned} \tag{2.28}$$

with $\Phi_n = \sum_{i=1}^{n} \rho^{n-i} \mathbf{x}_i \mathbf{x}_i^T$ and $\mathbf{z}_n = \sum_{i=1}^{n} \rho^{n-i} \mathbf{y}_i \mathbf{x}_i$. We can compute Φ_n and $\mathbf{z}_n$ recursively given by

$$\begin{aligned} \Phi_n &= \rho \Phi_{n-1} + \mathbf{x}_i \mathbf{x}_i^T \\ \mathbf{z}_n &= \rho \mathbf{z}_{n-1} + \mathbf{y}_i \mathbf{x}_i \end{aligned} \tag{2.29}$$

We need for the calculation of coefficient vector $\mathbf{b}_n$ the inverse matrix Φ_n^{-1}. Using a matrix inversion lemma [Haykin, 1996] a recursive update equation for $\mathbf{P}_n = \Phi_n^{-1}$ is found as

$$\begin{aligned} \mathbf{P}_n &= \rho^{-1} \mathbf{P}_{n-1} + \rho^{-1} \mathbf{k}_n \mathbf{x}_n \\ \mathbf{k}_n &= \tfrac{\rho^{-1} \mathbf{P}_{n-1} \mathbf{x}_n}{1 + \rho^{-1} \mathbf{x}_n^T \mathbf{P}_{n-1} \mathbf{x}_n} \end{aligned} \tag{2.30}$$

We end up with weight update given by

$$\mathbf{b}_n = \mathbf{b}_{n-1} + \mathbf{k}_n (\mathbf{y}_n - \mathbf{x}_n^T \mathbf{b}_{n-1}] \tag{2.31}$$

The RLS algorithm is computationally more complex than the LMS but RLS has significantly faster convergence behavior than LMS. Instead of computationally intensive matrix inversions we use recursive updating. This method decreases the computational complexity of RLS algorithm.

2.4 Dimensionality reduction for regression

In this section we will discuss two methods [D'Souza, 2004] that performs dimensionality reduction on the input space $\mathbf{x}$, resulting lower dimensional input that captures sufficient information to accurately predict the output y. In the first subsection principal component regression will be explained in detail. In the second subsection joint-space factor analysis for regression will be discussed.

2.4.1 Principal Component Regression (PCR)

Principal component regression (PCR) Massey [1965] is a method that selects the only the dimensions from input x with most variance. The selected dimensions captures the most essential information required to predict y. First we start with computation of the empirical covariance matrix $\mathbf{\Sigma_{PCR}}$ of the input data $\mathbf{x}$ and is given by

$$\mathbf{\Sigma_{PCR}} = \frac{1}{N-1}\sum_{n=1}^{N} \mathbf{x_n x_n^T} \tag{2.32}$$

we compute its eigen-decomposition:

$$\mathbf{\Sigma_{PCR} v_n} = \lambda_n \mathbf{v_n} \tag{2.33}$$

where $\mathbf{v_n}$ is each eigenvector and λ_n the corresponding eigenvalue. By projecting the input x on the principal K eigenvectors with largest eigenvalues in the columns of projection matrix $\mathbf{U} = [\mathbf{v_1 v_2 ... v_K}]$, the regression solution can be computed as follows:

$$\mathbf{b_{PCR}} = (\mathbf{U^T X^T X U})^{-1}\mathbf{U^T X^T y} \tag{2.34}$$

Because of the matrix $(\mathbf{U^T X^T X U})$ in equation (2.38) is a diagonal matrix, the inversion of this matrix become computationally convenient. PCR reduces the multivariate regression problem to a set of independent univariate regressions along each of the orthogonal principal component dimensions. This makes possible to do regression with dimensionality reduction but computation of eigen-decomposition is still a process that computationally expensive.

2.4.2 Joint-space factor analysis for regression

Factor analysis (FA) Everit [1984], Ghahramani and Hinton [1997] is a density estimation method. This method assumes that observed data $\mathbf{z}$ is generated from a lower dimensional process characterized by K hidden variables $\mathbf{v}$ and is given by

$$\mathbf{z_n} = \mathbf{W v_n} + \epsilon_\mathbf{n} \quad 1 \leq n \leq N \tag{2.35}$$

An assumption is made that hidden variables $\mathbf{v}$ and noise ϵ are normally independently distributed and is given by

$$\begin{aligned} \mathbf{v_n} &\sim \mathcal{N}(\mathbf{v_n}; 0, I) \\ \epsilon_\mathbf{n} &\sim \mathcal{N}(\epsilon_\mathbf{n}; 0, \mathbf{\Psi}) \end{aligned} \tag{2.36}$$

The factor W and the noise variance matrix $\mathbf{\Psi}$ can be easily estimated using Expectation-Maximization (EM) [Ghahramani and Hinton, 1997] algorithm or Bayesian [Ghahramani and Beal, 2000] techniques. For joint-space factor analysis for regression, first we define three matrixes $\mathbf{z}, \mathbf{W}$ and $\mathbf{\Psi}$ given by

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} \mathbf{W_x} \\ \mathbf{W}_y \end{bmatrix} \qquad \mathbf{\Psi} = \begin{bmatrix} \mathbf{\Psi_x} & \mathbf{0} \\ \mathbf{0^T} & \psi_y \end{bmatrix} \tag{2.37}$$

After estimation of $\mathbf{W}$ and $\mathbf{\Psi}$ for the joint space of $\mathbf{z}$ we can condition y on $\mathbf{x}$ by integrating out the hidden variable $\mathbf{v}$. We obtain the following

$$\mathbf{b}_{\mathrm{JFR}} = \mathbf{\Psi_x^{-1}} \mathbf{W_x} (\mathbf{I} + \mathbf{W_x^T} \mathbf{\Psi_x^{-1}} \mathbf{W_x})^{-1} \mathbf{W}_y^{\mathbf{T}} \tag{2.38}$$

The required matrix inversion for $(\mathbf{I} + \mathbf{W_x^T} \mathbf{\Psi_x^{-1}} \mathbf{W_x})$ is in the order of K dimensional. This makes joint-space factor analysis for regression method computationally attractive ($K \ll d$) for problems which underlying hidden variable model is low dimensional.

2.5 Conclusions

In order to find the best method that can predict the optimal model parameters from a multiple linear regression problem we have evaluated different methods in this chapter. As explained before in subsection 1.3.2 there are some points that we have to take care of during the evaluation of this methods. We need a method that can handle with five mentioned points in the problem statement subsection. This points were, *irrelevancy, redundancy, computational efficiency, probabilistic embedding and adaptivity.*

In section 2.2 we explained two standard methods Maximum likelihood and Least Squares (LS) method for our multiple linear regression problem. These two methods are most commonly used methods in the literature for point estimations. In our case we need a method that can do feature selection in order to discard the irrelevant and redundant dimensions. These two methods can not handle with feature selection. The computational complexity for both methods is very high (approximately $O(d^3)$) therefore they are not computationally efficient. The Maximum likelihood and Least Squares (LS) methods can also not handle with probabilistic embedding and adaptivity because these methods are not adaptive methods with probabilistic interpretation.

In section 2.3 we explained two most commonly used adaptive models, namely LMS and RLS. These two methods do not have the ability to discard the irrelevant and redundant dimensions. The computational complexity for both is not very high, approximately $O(d)$ therefore they are very efficient but they do not have probabilistic interpretation. If we compare the performance of two methods we can conclude that RLS is computationally more complex than the LMS but RLS has significantly faster convergence behavior than LMS.

All explained methods in sections 2.2 and 2.3 were methods without feature selection. In section 2.4 we explained two methods Principal Component Regression (PCR) and Joint-space factor analysis for regression (JFR). These methods can reduce the input dimensions such a way that the computational complexity will decreases for the regression problem. Is is also possible to integrate both methods in an adaptive model. We see that in this section explained methods can handle with 4 mentioned points above, namely *irrelevancy, redundancy, computational efficiency and adaptivity* but they do not have probabilistic interpretations.

Up to now we have explained different methods with different properties. We constructed a table (2.1) that represents the evaluation results on the methods OLS, LMS, RLS, PCR and JFR. All these methods were methods without probabilistic interpretation therefore we have to go on with evaluating methods for a multiple linear regression problem in Bayesian framework. Because Bayesian methods have the advantages that confidence intervals can be produced for parameter

estimates without the use of asymptotic approximations and prior information can be incorporated into the analysis. These means that Bayesian methods have probabilistic interpretation, now we need a Bayesian Linear regression method that can handle with *irrelevancy, redundancy, computational efficiency and adaptivity.* In the next chapter we will evaluate step by step a Bayesian linear regression method in order to get one that can handle with all mentioned points above.

Table 2.1: *Evaluation results of described methods on irrelevancy, redundancy, computational efficiency, probabilistic embedding and adaptivity. The character d is the number of dimensions of the input matrix given that* $K \ll d$. $(-)$ *can not handle,* $(+)$ *good,* $(++)$ *better.*

Criteria	OLS	LMS	RLS	PCR	JFR
irrelevancy	$-$	$-$	$-$	$+$	$++$
redundancy	$-$	$-$	$-$	$+$	$++$
computational complexity	$O(d^3)$	$O(d)$	$O(d^2)$	$O(Kd^2)$	$O(K^3)$
probabilistic embedding	$-$	$-$	$-$	$-$	$-$
adaptivity	$-$	$+$	$++$	$-$	$-$

Chapter 3

Bayesian linear regression

Bayesian methods have probabilistic interpretation therefore in this chapter we are going to construct a Bayesian linear regression method by adapting an existing method namely, backfitting method [Hastie and Tibshirani, 1990]. In the first section (3.1) a short introduction on backfitting method will be given. Before we modify the backfitting model, the graphical model theory will be explained. We also discuss the well-known expectation-maximization (EM) algorithm [Dempster et al., 1977] that are used to perform Bayesian inference in graphical models. In the second section (3.2) probabilistic backfitting model will be introduced. Probabilistic backfitting model are able to decompose the multivariate regression problem into successive univariate regressions to prevent the difficulties in matrix inversions. In the last section (3.3) of this chapter, probabilistic backfitting will be used in Bayesian framework to control the complexity (regularization) of the model.

3.1 Introduction

In the previous chapter (2) we have evaluated different methods in order to find one that can handle in the problem statement (subsection 1.3.2) mentioned five points: *Irrelevancy, redundancy, computational efficiency, probabilistic embedding and adaptivity.* Only in the subsection 2.4 described methods were methods with best performance. They were able to handle with four of the five mentioned points but *probabilistic embedding* was still a problem. Therefore we will construct in this chapter a statistical model in Bayesian framework that can handle with all mentioned five points above by modifying the backfitting method. First we will start with explaining the backfitting method.

3.1.1 Backfitting method

Backfitting [Hastie and Tibshirani, 1990] is a method for estimating additive models of the form $y(\mathbf{x}) = \sum_{m=1}^{d} g_m(\mathbf{x}; \theta_m)$, where the functions g_m absorb the parameters b_m in form of $g_m = b_m f_m$. The basis function g_m also contain the parameter θ_m which can be used to adjust the shape of a nonlinear basis function. Backfitting decomposes the statistical estimation problem in to d individual estimation problems by creating "fake supervised targets" for each g_m function. This method effectively decouples inference in each individual dimension leading to a highly efficient algorithm given in algorithm 1.

Although backfitting is an efficient algorithm, there are two serious drawbacks. Firstly, backfitting has no probabilistic interpretation, therefore it is difficult to insert backfitting into the framework of Bayesian statistical learning. Secondly, even the simplest case of linear regression, backfitting provides no guarantees of convergence. Therefore we are going to insert the backfitting method

Algorithm 1 Backfitting

1: Init: $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_\mathbf{N})^\mathbf{T}, \mathbf{y} = (y_1, y_2, ..., y_N)^\mathbf{T}, g_{m,n} = g_m(\mathbf{x_n}; b_m, \theta_m), \mathbf{g}_m = (g_{m,1}, ..., g_{m,N})^\mathbf{T}$
2: **repeat**
3: **for** $m = 1$ to d **do**
4: $\mathbf{r}_m \Leftarrow \mathbf{y} - \sum_{k \neq m} \mathbf{g}_k$
5: $\mathbf{g}_m \Leftarrow argmin_{b_m, \theta_m} (\mathbf{g}_m - \mathbf{r}_m)^2$
6: **end for**
7: **until** convergence

in a statistical EM framework in order to get a algorithm with probabilistic interpretation that provides guarantees of convergence. We will explain the EM algorithm in detail but before we start with EM we will first explain the graphical models that we are going to use regularly in this chapter.

3.1.2 Graphical models

Graphical models are simple tool for representing the structure of a statistical model and playing an increasingly important role in the design and analysis of machine learning algorithms. The basic idea of a graphical model is to build a complex system by combining of simpler parts. Probability theory provides the glue whereby the parts are combined, ensuring that the system as a whole is consistent, and providing ways to interface models to data. Probabilistic interpretation of graphical models makes it possible to eliminate the nuisance parameters in the framework of Bayesian inference.

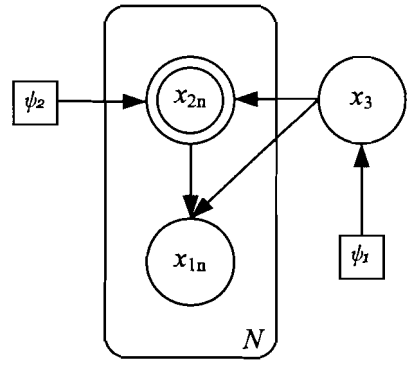


Figure 3.1: Graphical model example.

Consider the following example shown in figure 3.1. Random variables x_1, x_2, x_3 in the model are drawn as circular nodes. We will distinguish between observed and unobserved variables by giving nodes corresponding to observed variables a double border. The collection of nodes with double border represents the observed data set $\mathbf{x}_\mathcal{D} = \{x_2\}$.

Our main goal in the graphical model is to use Bayesian inference in order to obtain the marginal posterior distribution over unobserved (hidden) random variables. These unobserved variables are referred using by the symbol $\mathbf{x}_\mathcal{H} = \{x_1, x_3\}$. The parameters $\{\psi_1, \psi_2\}$ in the graphical model do not have distributions and are given by square nodes. Edges in the graph are used to indicate dependencies between variables. A directed edge from x_2 to x_1 implies that the distribution of x_1 is conditioned on x_2. Repeated sets of variables (x_1 and x_2 in figure 3.1) are denoted by the use

of a rectangular box around the variables. These variables have subscripts denoting iteration over the repetitions. Each variable in the rectangular box is conditionally i.i.d given all parent nodes outside the box. We have explained graphical models in detail now. In the next section (3.1.3) EM algorithm for parameter estimation will be explained in detail.

3.1.3 EM Algorithm for parameter estimation

The aim of construction of a graphical model is to estimate the marginal distributions over sets of variables. We are also often interested in optimizing the parameter values $\boldsymbol{\phi}$ of the model itself such that we are most likely to generate our observations $\mathbf{x}_{\mathcal{D}}$. These can be reached by maximizing the the marginal likelihood function $p(\mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi})$ and is called *maximum likelihood* (ML) framework.

The quantity $p(\mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi})$ in the ML framework refers to *incomplete* likelihood under the assumption that only a subset of the nodes in the model corresponding to $\mathbf{x}_{\mathcal{D}}$ are observed. The quantity $p(\mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{H}}; \boldsymbol{\phi})$ is called *complete* likelihood since we must assume that the values of the hidden variables $\mathbf{x}_{\mathcal{H}}$ are known as well. Given the underlying graphical model, observed data $\mathbf{x}_{\mathcal{D}}$ and the mathematical form of the distributions we can find the parameter values that maximize the incomplete likelihood $\boldsymbol{\phi}^* = \text{argmax}_{\boldsymbol{\phi}}\, p(\mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi})$, but this is not a trivial task for the following reasons:

- We need to marginalize the the quantity $p(\mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{H}}; \boldsymbol{\phi})$ over hidden variables $\mathbf{x}_{\mathcal{H}}$ in order to obtain the quantity $p(\mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi})$ but this is difficult because $\mathbf{x}_{\mathcal{H}}$ variables are not known.

$$p(\mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi}) = \int p(\mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{H}}; \boldsymbol{\phi}) d\mathbf{x}_{\mathcal{H}} \tag{3.1}$$

- Even when the quantity is analytically expressed, for the optimization of parameters $\boldsymbol{\phi}$ we have to solve the equation (3.2). This is also a trivial task because the parameters $\boldsymbol{\phi}$ are linked in a complex manner so it is not possible to obtain a expression for $p(\mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi})$ in terms of the observed variables $\mathbf{x}_{\mathcal{D}}$.

$$\left. \frac{\partial p(\mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \right| = 0 \tag{3.2}$$

The expectation-maximization (EM) algorithm [Dempster et al., 1977] is a parameter optimization algorithm that operates within the framework of ML parameter estimation. Instead of computation of marginalized likelihood integral of equation (3.1) it is much more easier to express the complete likelihood $p(\mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{H}}; \boldsymbol{\phi})$ as a product of conditional distributions obtained from the graphical model.

The EM algorithm is able to perform the optimization of the model parameters $\boldsymbol{\phi}$ working with the analytical convenience of the complete likelihood $p(\mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{H}}; \boldsymbol{\phi})$. Several derivations of EM algorithm exist, in this report we shall use an EM algorithm which relies on creating a lower bound to the incomplete likelihood using Jensen's inequality. For this purpose first let us hypothesize the existence of an arbitrary distribution $Q(\mathbf{x}_{\mathcal{H}})$ over the unobserved variables $\mathbf{x}_{\mathcal{D}}$ which allows us to derive the following lower bound on the incomplete log likelihood given by

$$
\begin{aligned}
\ln p(\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi}) &= \ln \int p(\mathbf{x}_{\mathcal{D}},\mathbf{x}_{\mathcal{H}};\boldsymbol{\phi})d\mathbf{x}_{\mathcal{H}} \\
&= \ln \int Q(\mathbf{x}_{\mathcal{H}})[\frac{p(\mathbf{x}_{\mathcal{D}},\mathbf{x}_{\mathcal{H}};\boldsymbol{\phi})}{Q(\mathbf{x}_{\mathcal{H}})}]d\mathbf{x}_{\mathcal{H}} \\
&\geq \int Q(\mathbf{x}_{\mathcal{H}}) \ln [\frac{p(\mathbf{x}_{\mathcal{D}},\mathbf{x}_{\mathcal{H}};\boldsymbol{\phi})}{Q(\mathbf{x}_{\mathcal{H}})}]d\mathbf{x}_{\mathcal{H}} \quad \text{(Jensen's inequality)} \qquad (3.3) \\
&= \langle \ln p(\mathbf{x}_{\mathcal{D}},\mathbf{x}_{\mathcal{H}};\boldsymbol{\phi})\rangle_{Q} + \mathcal{H}[Q] \\
&= \mathcal{F}(Q,\boldsymbol{\phi})
\end{aligned}
$$

where $\langle . \rangle_Q$ denotes expectation with respect to the distribution Q, and $\mathcal{H}[Q]$ denotes its entropy. For a special case lower bound $\mathcal{F}(Q,\boldsymbol{\phi})$ in equation (3.3) becomes an equality when the distribution $Q(\mathbf{x}_{\mathcal{H}}) = p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$ (the posterior distribution of the hidden variables under the current parameter settings). This is verified in Appendix A.1 by substituting $p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$ for Q in equation (3.3). It is easy to exactly quantify the tightness of the bound as being the Kullback-Leibler (KL) divergence between $Q(\mathbf{x}_{\mathcal{H}})$ and $p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$ given by

$$
\ln p(\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi}) = \mathcal{F}(Q,\boldsymbol{\phi}) + KL[Q(\mathbf{x}_{\mathcal{H}})\|p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})] \qquad (3.4)
$$

Algorithm 2 Expectation-Maximization

1: Init: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}_0$
2: **repeat**
3: E-step
$l_c(\boldsymbol{\phi}) \leftarrow \langle p(\mathbf{x}_{\mathcal{D}},\mathbf{x}_{\mathcal{H}};\boldsymbol{\phi})\rangle_{p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})}$
4: M-step
$\boldsymbol{\phi} \leftarrow \text{argmax}_{\boldsymbol{\phi}}\, l_c(\boldsymbol{\phi})$
5: **until** convergence

The EM algorithm (algorithm 2) maximize the complete log likelihood $\ln p(\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$ not directly. EM operates on the functional $\mathcal{F}(Q,\boldsymbol{\phi})$ by alternating between an E-step, which infers posterior distributions over hidden variables $Q(\mathbf{x}_{\mathcal{H}})$ given a current parameter setting, and an M-step, which maximizes $l_c(\boldsymbol{\phi})$ with respect to $\boldsymbol{\phi}$ given the statistics gathered from E-step.

By evaluating the expectation $\langle p(\mathbf{x}_{\mathcal{D}},\mathbf{x}_{\mathcal{H}};\boldsymbol{\phi})\rangle_Q$ under the specific distribution $Q(\mathbf{x}_{\mathcal{H}}) = p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$ the E-step ensures that the functional lower bound $\mathcal{F}$ is raised to an equality. The M-step then maximizes $\mathcal{F}$ with respect to $\boldsymbol{\phi}$. Since the E-step makes the bound an equality, we are guaranteed that maximizing $\mathcal{F}$ in the M-step also increases the incomplete log likelihood $\ln p(\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$. According to Wu [1983] under general conditions EM is guaranteed to converge to a local maximum in the incomplete likelihood $p(\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$. The ability to produce the desired result in the M-step depends on being able to successfully compute the expectation of the complete log likelihood $\ln p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$ with respect to the *true posterior* $p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$. The *true posterior* will raise the bound $\mathcal{F}$ to an equation. Raising the bound $\mathcal{F}$ means that the M-step is actually maximizing incomplete likelihood $p(\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$. It can be occur that we are not able to express the posterior $p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$ analytically. When this happen we can prevent this intractability by using of several approximation methods. In this report we have chosen for factorial variational approximation and will be described in the next subsection (3.1.4).

3.1.4 The Factorial Variational Approximation

As explained in previous subsection (3.1.3) the marginal posterior distributions over variables is analytically intractable due to the structure of the individual conditional distributions. An alternative way to prevent the computational intractability is to make the a priori assumption that the posterior distribution factorizes over certain sets of variables in the model [Beal, 2003]. This approximation is known in the literature as the factorial variational approximation.

In the subsection 3.1.3 derived equation (3.4) showed us that the tightness of the bound is quantified by the KL divergence between the true posterior $Q(\mathbf{x}_{\mathcal{H}})$ and its approximation $p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$. This KL divergence is also known as the *variational free energy* which can be minimized by bringing the approximation $Q(\mathbf{x}_{\mathcal{H}})$ closer to the true posterior $p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$ (tightening the lower bound). Because of the computational tractability of the marginal posterior distributions we will now impose the additional constraint that the distribution $Q(\mathbf{x}_{\mathcal{H}})$ must factorize over some partition given by

$$Q(\mathbf{x}_{\mathcal{H}}) = \prod_{i=1}^{p} Q_i(\mathbf{x}_{\mathcal{H}_i}) \tag{3.5}$$

From this factorized approximation (3.5) we can derive the following update equation for the marginal posterior over each individual partition $\mathbf{x}_{\mathcal{H}_i}$ given by

$$\ln Q_i(\mathbf{x}_{\mathcal{H}_i}) = \langle \ln p(\mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{H}};\boldsymbol{\phi})\rangle_{Q_{k\neq i}} + const_{\mathbf{x}_{\mathcal{H}_i}} \tag{3.6}$$

where $\langle . \rangle_{Q_{k\neq i}}$ denotes expectation with respect to the all distribution Q_k except Q_i. This formulation can be used in the framework of statistical modeling of EM algorithm as we explained in previous subsection (3.1.3). Mathematically speaking, using a superscript (t) to denote the iteration number, starting from initial parameters $\boldsymbol{\phi}^{(0)}$, the update equations for E and M-step are given by

$$\text{E step:} \quad Q_i(\mathbf{x}_{\mathcal{H}_i})^{(t+1)} \leftarrow \underset{Q_i(\mathbf{x}_{\mathcal{H}_i})}{\operatorname{argmax}} \mathcal{F}(Q_{k\neq i}(\mathbf{x}_{\mathcal{H}_{k\neq i}}), \boldsymbol{\phi}^{(t)}) \tag{3.7}$$

$$\text{M step:} \quad \boldsymbol{\phi}^{(t+1)} \leftarrow \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \mathcal{F}(Q(\mathbf{x}_{\mathcal{H}})^{(t+1)}, \boldsymbol{\phi}) \tag{3.8}$$

The update (3.6) to each distribution $Q_i(\mathbf{x}_{\mathcal{H}_i})$ for $1 \leq i \leq p$ guarantees that each update monotonically decreases the KL divergence between the approximated (factored) posterior $Q(\mathbf{x}_{\mathcal{H}})$ and the true posterior $p(\mathbf{x}_{\mathcal{H}}|\mathbf{x}_{\mathcal{D}};\boldsymbol{\phi})$. In the next section (3.2) we will introduce the probabilistic version of backfitting method in the statistical framework of EM algorithm.

3.2 Probabilistic version of backfitting

In this section we will explain in detail how to construct a probabilistic model from backfitting method. The constructed probabilistic backfitting model will be extended in the next section (3.3) in order to get a model in a statistical Bayesian framework. A modification to the standard graphical model for multiple linear regression in figure 3.2 allows us to derive a probabilistic version of backfitting. Our goal is the same: For a given data set $\{(\mathbf{x_n}, y_n)\}_{n=1}^{N}$ determine the most likely values of b_m which linearly combine the basis functions f_m to generate the output y. In the standard graphical model for multiple linear regression, rows of design matrix $\mathbf{X}$ contains the $f_m(\mathbf{x_n})$ of all data points $\{(\mathbf{x_n}, y_n)\}_{n=1}^{N}$.

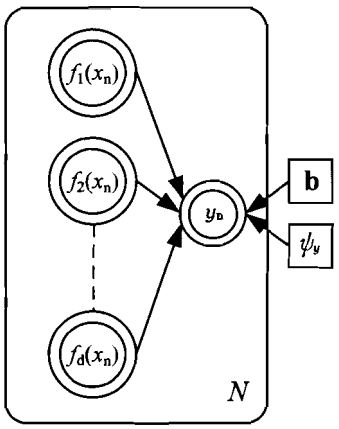


Figure 3.2: Standard graphical model for multiple linear regression.

Consider introduction of random variable z_{nm} as shown in figure 3.3. These random variable z_{nm} decouples the predictor functions in the graphical model, and gives backfitting a probabilistic interpretation. Random variable z_{nm} is analogues to the output of the $g_m(x_{nm})$ function of algorithm 1, and can also be interpreted as a unknown *fake target* for each branch of regression input.

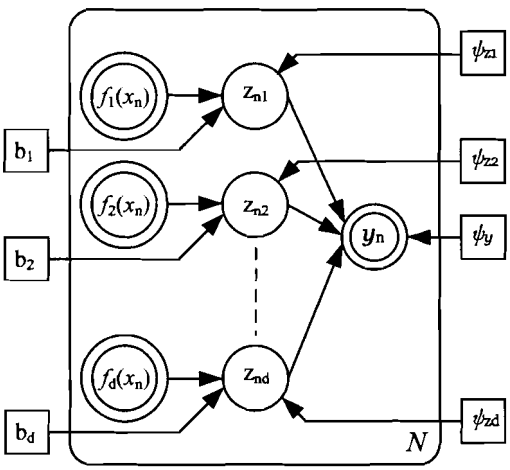


Figure 3.3: Graphical model for Probabilistic Backfitting.

The modified graphical model can be solved using EM (Expectation-Maximization) framework [Dempster et al., 1977] to derive a probabilistic version of backfitting. For the derivation, we assume that z_{nm} and y_n are conditionally normally distributed:

$$y|\mathbf{z}_n \sim \mathcal{N}(1^T\mathbf{z}_n, \psi_y) = \mathcal{N}\Big(\sum_{m=1}^{d} z_{nm}, \psi_y\Big) \tag{3.9}$$

$$z_{nm}|\mathbf{x}_n \sim \mathcal{N}(z_{nm}, \psi_{zm}) = \mathcal{N}(b_m f_m(\mathbf{x}_n), \psi_{zm}) \tag{3.10}$$

The modified graphical model contains now parameters $\phi = \{\{b_m, \psi_{zm}\}_{m=1}^{d}, \psi_y\}$ that can be optimized by using of EM algorithm in the framework of *maximum likelihood* estimation, given the

observed variables $\mathbf{x}_\mathcal{D} = \{(\mathbf{x_n}, y_n)\}_{n=1}^N$ and the unobserved variables $\mathbf{x}_\mathcal{H} = \{\mathbf{z_n}\}_{n=1}^N$. In this thesis we only optimize the model parameters ϕ. We assume that the individual functions $f_m(\mathbf{x}; \theta_m)$ are linear, therefore optimization of the individual functions with respect to the parameters $\boldsymbol{\theta}_m$ is omitted ($f_m(\mathbf{x_n}; \theta_m) = x_{nm}$ for $n = 1, ..., N$). Optimization of model parameters ϕ can be easily formulated as an EM algorithm, which maximizes the *incomplete* log likelihood $\ln p(\mathbf{x}_\mathcal{D}; \phi)$

$$\ln p(\mathbf{x}_\mathcal{D}; \phi) = \ln p(\mathbf{y}|\mathbf{X}; \mathbf{b}, \boldsymbol{\Psi}_\mathbf{z}, \psi_y) = -\frac{N}{2} \ln \psi_y - \frac{1}{2\psi_y} \sum_{n=1}^{N} (y_n - \mathbf{b^T} f(\mathbf{x}_n))^2 + const \tag{3.11}$$

by maximizing the expected *complete* log likelihood ($\ln p(\mathbf{x}_\mathcal{D}, \mathbf{x}_\mathcal{H}; \phi)$), where

$$\begin{aligned} \ln p(\mathbf{x}_\mathcal{D}, \mathbf{x}_\mathcal{H}; \phi) = \ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}; \mathbf{b}, \boldsymbol{\Psi}_\mathbf{z}, \psi_y) &= -\frac{N}{2} \ln \psi_y - \frac{1}{2\psi_y} \sum_{n=1}^{N} (y_n - 1^T \mathbf{z}_n)^2 \\ &- \sum_{m=1}^{d} \Big[\frac{N}{2} \ln \psi_{zm} + \frac{1}{2\psi_{zm}} \sum_{n=1}^{N} (z_{nm} - b_m f_m(\mathbf{x}_n))^2 \Big] + const \end{aligned} \tag{3.12}$$

In the *complete* log likelihood equation 3.12 $\mathbf{Z}$ denotes the N by d matrix of all z_{nm}. The derived EM update equations [D'Souza, 2004] for b_m and the noise variances ψ_y and ψ_{zm} are given by

M – Step :	**E – Step :**
$b_m = \frac{\sum_{n=1}^N \langle z_{nm} \rangle f_m(\mathbf{x}_n)}{\sum_{n=1}^N f_m(\mathbf{x}_n)^2}$	$1^T \boldsymbol{\Sigma}_\mathbf{z} 1 = \Big(\sum_{m=1}^{d} \psi_{zm} \Big) \Big[1 - \frac{1}{s} \Big(\sum_{m=1}^{d} \psi_{zm} \Big) \Big]$
$\psi_y = \frac{1}{N} \sum_{n=1}^{N} (y_n - 1^T \langle \mathbf{z}_n \rangle)^2 + 1^T \boldsymbol{\Sigma}_\mathbf{z} 1$	$\sigma_{zm}^2 = \psi_{zm} \Big(1 - \frac{1}{s} \psi_{zm} \Big)$
$\psi_{zm} = \frac{1}{N} \sum_{n=1}^{N} (\langle z_{nm} \rangle - b_m f_m(\mathbf{x}_n))^2 + \sigma_{zm}^2$	$\langle z_{nm} \rangle = b_m f_m(\mathbf{x}_n) + \frac{1}{s} \psi_{zm} (y_n - \mathbf{b}^T \mathbf{f}(\mathbf{x}_n))$

(3.13)

where we define $s = \psi_y + \sum_{m=1}^{d} \psi_{zm}$ and $\boldsymbol{\Sigma}_\mathbf{z} = \mathrm{Cov}(\mathbf{z}|\mathbf{y}, \mathbf{X})$. It is easy to see that in both the maximization and expectation steps are algorithmically $O(d)$ where d is the number of predictor functions f_m. Substitution of the expression z_{nm} in the maximization equation for b_m result in the following update equation

$$b_m^{(n+1)} = b_m^{(n)} + \frac{\psi_{zm}}{s} \frac{\sum_{n=1}^N \Big(y_n - \sum_{k=1}^d b_k^{(n)} f_k(\mathbf{x}_n) \Big) f_m(\mathbf{x}_n)}{\sum_{n=1}^N f_m(\mathbf{x}_n)^2} \tag{3.14}$$

This update equation (3.14) gives the new value of ${b_m}^{(n+1)}$ for each update cycle of EM. The new $b_m^{(n+1)}$ value is obtained by adding the previous update value of $b_m^{(n)}$ an amount proportional to the correlation between the m th predictor and the residual error. Equation (3.14) enables us to place this algorithm in the context of *backfitting* because the residual error can be interpreted as forming a "*fake target*" for the m th branch of the input. The main question is now how can equation (3.14) be seen in the context of backfitting. For this purpose we assume that backfitting can be viewed as a formal Gauss-Seidel algorithm that becomes exact in the special case of linear models [Hastie and Tibshirani, 1990]. As calculated before (2.18) the linear system $\mathbf{X}^T \mathbf{X} \mathbf{b} = \mathbf{X}^T \mathbf{y}$ can be written in the form of Gauss-Seidel updates for the individual b_m given by

$$b_m = \frac{\sum_{n=1}^{N}\Big(y_n - \sum_{k\neq m}^{d} b_k f_k(\mathbf{x}_n)\Big) f_m(\mathbf{x}_n)}{\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2} \tag{3.15}$$

Note that equation (3.15) only guarantees convergence for very specially structured matrices. The Gauss-Seidel algorithm can be extended by adding a fraction $(1-\omega)$ of b_m. This well-known extension is called *successive relaxation* algorithm. The *relaxation* algorithm is given by

$$b_m^{(n+1)} = (1-\omega) b_m^{(n)} + \omega \frac{\sum_{n=1}^{N}\Big(y_n - \sum_{k\neq m}^{d} b_k^{(n)} f_k(\mathbf{x}_n)\Big) f_m(\mathbf{x}_n)}{\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2} \tag{3.16}$$

which has improved convergence rates for *overrelaxation* $(1 < \omega < 2)$, or improved stability for *underrelaxation* $(0 < \omega < 1)$. For $\omega = 1$ the standard backfitting or Gauss-Seidel equation(3.15) is recovered. When we set $\omega = \psi_{zm}/s$ (see Appendix A.2) we see that the update equation (3.16) become exactly the same update equation in (3.14). This means that we have derived a probabilistic version of backfitting.

The *incomplete* log likelihood in equation (3.11) has only a *global* maximum corresponding to OLS, but could the introduction of hidden variables $\mathbf{Z}$ and the corresponding parameters $\mathbf{\Psi_z}$ in equation(3.11) introduce local maxima. We can examine the convergence property by estimating the parameters $\boldsymbol{\phi} = (\mathbf{b}, \psi_{z1}, ..., \psi_{zd}, \psi_y)$. For this purpose we assume that we have reached a stationary point $\boldsymbol{\phi}^*$ in the EM algorithm, which implies

$$\left.\frac{\partial \langle \ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}; \boldsymbol{\phi})\rangle}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = 0 \tag{3.17}$$

Jensen's inequality have been applied to the *incomplete* log likelihood (3.11) in combination with a arbitrary chosen distribution over the hidden variables $Q(\mathbf{Z})$ that results in following inequality

$$\ln p(\mathbf{y}|\mathbf{X}; \boldsymbol{\phi}) \geq \langle \ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}; \boldsymbol{\phi})\rangle_{Q(\mathbf{Z})} + \mathcal{H}[Q(\mathbf{Z})] = \mathcal{F}(Q, \boldsymbol{\phi}) \tag{3.18}$$

where $\mathcal{H}[Q(\mathbf{Z})]$ denotes entropy of the $Q(\mathbf{Z})$ distribution. $\mathcal{F}(Q, \boldsymbol{\phi})$ is a lower bound on $\ln p(\mathbf{y}|\mathbf{X}; \boldsymbol{\phi})$ and is also known in statistics as free energy function. In the EM algorithm E-step maximizes $\mathcal{F}(Q, \boldsymbol{\phi})$ with respect to Q, and the M step does so with respect to $\boldsymbol{\phi}$ given the statistics gathered from E step. Differentiating $\mathcal{F}(Q, \boldsymbol{\phi})$ with respect to $\boldsymbol{\phi}$ at the stationary point $\boldsymbol{\phi}^*$ gives

$$\left.\frac{\partial \mathcal{F}(Q, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = \left.\frac{\partial \langle \ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}; \boldsymbol{\phi})\rangle}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = 0 \tag{3.19}$$

For the E step the maximum over $Q(\mathbf{Z})$ of the bound (3.18) is obtained by setting $Q(\mathbf{Z})$ to the true posterior distribution $\ln p(\mathbf{Z}|\mathbf{y}, \mathbf{X}; \boldsymbol{\phi}^*)$ at which point the bound becomes an equality given by

$$\ln p(\mathbf{y}|\mathbf{X}; \boldsymbol{\phi}) = \mathcal{F}(Q, \boldsymbol{\phi}) \Rightarrow \left.\frac{\partial \ln p(\mathbf{y}|\mathbf{X}; \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = \left.\frac{\partial \mathcal{F}(Q, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = 0 \tag{3.20}$$

We have reached a maximum in the incomplete likelihood. According to D'Souza [2004], if we assume that *incomplete* log likelihood $\ln p(\mathbf{y}|\mathbf{X}; \boldsymbol{\phi})$ in equation (3.11) has *only* a global maximum for OLS solution, then reaching the stationary point of equation (3.17) means that EM algorithm for probabilistic backfitting corresponds to finding the OLS solution. Therefore, probabilistic backfitting have also a global optimum for the solution of the multiple linear regression problem.

3.3 Bayesian backfitting

As described in previous section (3.2) we have introduced probabilistic version of standard backfitting. Probabilistic backfitting makes it possible to use a Bayesian framework to regularize its OLS solution against overfitting. This purpose can be achieved by placing a prior distribution over the regression coefficients **b**. In this section we will describe how to form a model for Bayesian backfitting from existing (figure3.3) probabilistic backfitting model.

In our graphical model for probabilistic backfitting we can place individual precision variables α_m over each of the regression parameters b_m and z_{nm}. The individual precision variables α_m controls the regression parameter b_m and the hidden variables z_{nm}. Our uncertainty in the value of prior precision is represented by a Gamma distribution over α_m. Figure3.4 shows the probabilistic backfitting model with individual priors.

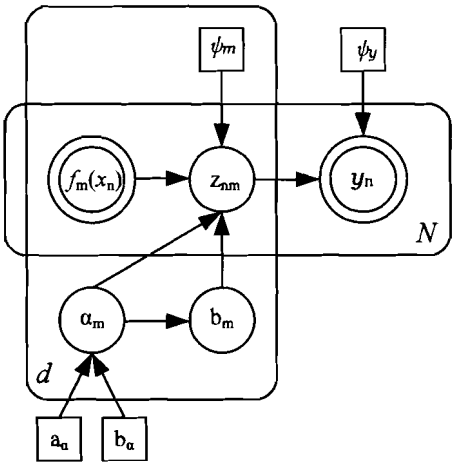


Figure 3.4: Graphical model for Bayesian Backfitting.

The graphical model for Bayesian backfitting can captured by the following set of prior distributions:

$$\begin{aligned} y_n|\mathbf{z_n} &\sim \mathcal{N}(1^T\mathbf{z_n}, \psi_y) \\ z_{nm}|b_m, \alpha_m &\sim \mathcal{N}(b_m f_m(x_n)), \psi_{zm}/\alpha_m) \\ b_m|\alpha_m &\sim \mathcal{N}(0, 1/\alpha_m) \\ \alpha_m &\sim \text{Gamma}(a_{\alpha_m}, b_{\alpha_m}) \end{aligned} \tag{3.21}$$

Placing of priors over each regression coefficient increases the number of hidden (unobserved) variables $\mathbf{x}_\mathcal{H} = \{\mathbf{b}, \boldsymbol{\alpha}, \{\mathbf{z}_n\}_{n=1}^N\}$ in the model(figure3.4). We are interested in obtaining posterior distributions over the variables **b** and $\boldsymbol{\alpha}$. Introducing the new hidden variables **b** and $\boldsymbol{\alpha}$ in the *complete* likelihood function (3.12) results in joint probability distribution over Bayesian backfitting model given by

$$
\begin{aligned}
\ln p(\mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{H}}; \boldsymbol{\phi}) &= \ln p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \boldsymbol{\alpha} | \mathbf{X}; \boldsymbol{\Psi}_{\mathbf{z}}, \psi_y, a_\alpha, b_\alpha) \\
&= -\frac{N}{2} \ln \psi_y - \frac{1}{2\psi_y} \sum_{n=1}^{N} (y_n - 1^T \mathbf{z}_n)^2 \\
&\quad - \sum_{m=1}^{d} \Big[\frac{N}{2} \ln \psi_{zm} + \frac{\alpha_m}{2\psi_{zm}} \sum_{n=1}^{N} (z_{nm} - b_m f_m(\mathbf{x}_n))^2 \Big] \\
&\quad + \sum_{m=1}^{d} \Big[\frac{d}{2} \ln \alpha_m - \frac{\alpha_m}{2} b_m^2 \Big] \\
&\quad + \sum_{m=1}^{d} \Big[(a_\alpha - 1) \ln \alpha_m - b_\alpha \alpha_m \Big] + \text{const}_{\mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{H}}}
\end{aligned}
\tag{3.22}
$$

It is possible to obtain the log joint posterior $\ln p(\mathbf{x}_{\mathcal{H}} | \mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi})$ from this joint probability distribution (3.22) but the intractability of the extraction of marginal probabilities of interest such as $\ln p(\mathbf{b} | \mathbf{x}_{\mathcal{D}}; \boldsymbol{\phi})$ enforce us to use a technic called factorial variational approximation. This technic factorizes over the variables of interest and can be written in the form of $Q(\mathbf{Z}, \mathbf{b}, \alpha) = Q(\mathbf{Z}) Q(\mathbf{b}, \boldsymbol{\alpha})$. We will obtain a joint posterior distribution $Q(\mathbf{b}, \boldsymbol{\alpha})$ that we have to marginalize to get the individual distributions $Q(\mathbf{b})$ and $Q(\boldsymbol{\alpha})$. The marginal distribution over $\mathbf{b}$ is a product of Student-t distributions. The corresponding parameters that belongs to distributions $Q(\mathbf{b})$ and $Q(\boldsymbol{\alpha})$ can be derived by the following iterative update equations [D'Souza, 2004].

$$
\begin{aligned}
Q(\boldsymbol{\alpha}) &= \prod_{m=1}^{d} \text{Gamma}(\alpha_m; \hat{a}_\alpha, \hat{b}_\alpha^{(m)}) \\
Q(\mathbf{b}) &= \prod_{m=1}^{d} t_\nu(b_m; \mu_{b_m}, \sigma_{b_m}^2) \\
\hat{a}_\alpha &= a_\alpha + \frac{N}{2} \\
\hat{b}_\alpha^{(m)} &= b_\alpha + \frac{1}{2\psi_{zm}} \Big[\sum_{n=1}^{N} \langle z_{nm}^2 \rangle - \Big(\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2 + \psi_{zm} \Big)^{-1} \Big(\sum_{n=1}^{N} \langle z_{nm} \rangle f_m(\mathbf{x}_n) \Big)^2 \Big] \\
\nu &= 2\hat{a}_\alpha \\
\mu_{b_m} &= \Big(\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2 + \psi_{zm} \Big)^{-1} \Big(\sum_{n=1}^{N} \langle z_{nm} \rangle f_m(\mathbf{x}_n) \Big) \\
\sigma_{b_m}^2 &= \frac{\hat{b}_\alpha^{(m)} \psi_{zm}}{\hat{a}_\alpha} \Big(\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2 + \psi_{zm} \Big)^{-1}
\end{aligned}
\tag{3.23}
$$

The updates for $Q(\mathbf{Z})$ are identical to the updates derived in previous section with exception that the parameters b_m are replaced with the expectations $\langle b_m \rangle$. Substituting the expressions for $\langle z_{im} \rangle$ in the update equations for the distribution of $Q(\mathbf{b})$ gives us the following update for the regression coefficients.

$$
\langle b_m \rangle^{(n+1)} = \Big(\frac{\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2}{\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2 + \psi_{zm}} \Big) \langle b_m \rangle^{(n)} + \frac{\psi_{zm}}{s\langle \alpha_m \rangle} \frac{\sum_{n=1}^{N} \Big(y_n - \sum_{k=1}^{d} \langle b_k \rangle^{(n)} f_k(\mathbf{x}_n) \Big) f_m(\mathbf{x}_n)}{\Big(\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2 + \psi_{zm} \Big)}
\tag{3.24}
$$

Equation (3.24) demonstrates that if there is no correlation between the current input and residual error, the first term causes the current regression coefficient to decay. The resulting regression solution regularizes over the number of retained inputs in the final regression vector. This can be seen in the framework of automatic relevance detection (ARD). The irrelevant and redundant dimensions will be determined by using of computed distribution parameters of $Q(\mathbf{b})$ and $Q(\alpha)$. Since $\langle \mu_{b_m} \rangle$ is zero mean, very large $\langle \alpha_m \rangle$ suggest that b_m is very close to 0 and has no contribution to the output.

3.4 Conclusions

The constructed Bayesian backfitting method is also known as Variational Bayesian Least Square (VBLS) method because of used approximation method is variational. In the next chapters we will use the designation VBLS instead of Bayesian backfitting. As explained before VBLS is a linear EM based probabilistic estimation method that can predict the marginal posterior distributions over regression coefficients b_m. Each regression coefficient is controlled by a Gamma distributed precision variable α_m which can be used to do automatic feature selection. The maximization and expectation steps are algorithmically $O(d)$ which means that this method is computationally efficient. VBLS method is also applicable in an adaptive model. In the literature there are several method exist that can extend an EM based batch algorithm into a sequential algorithm [Weinstein et al., 1988] [Neal and Hinton, 1998].

In the next chapter(4) we will test the performance of VBLS on in the subsection 1.3.2 mentioned five points by generating artificial regression data with known number of irrelevant and redundant dimensions. After that we will also test the performance of VBLS on real data sets that originate from six subjects.

Chapter 4

Experiments

As described in the previous chapter the Variational Bayesian Least Square (VBLS) model is the most promising one that can possibly handle with five issues mentioned in subsection 1.3.2. The effectiveness of the VBLS algorithm lies in its ability to predict the posterior distributions over each regression coefficient b_m such that the variance of each coefficient is controlled by precision parameter α_m. Because of the ARD mechanism, irrelevant or redundant dimensions will have small mean values b_m and large precision values α_m, so the posterior distributions over b_m will be narrow around zero. This makes possible to do automatic feature selection on regression data. In the first section (4.1) of this chapter we analyzed performance of the VBLS algorithm on five points mentioned in subsection 1.3.2. The performance is tested on artificial regression data sets which are generated from a random process. In the second section (4.2) we analyzed preference data set with VBLS collected from the real time platform.

4.1 Experiments on artificial regression data

In this section we present the results of performance study on VBLS using artificial regression data sets. This section is organized in three subsections. In the first subsection (4.1.1) we explain how artificial regression data with irrelevant and redundant features are generated. The second subsection (4.1.2) describes the method called *t-test* applied in VBLS algorithm for automatic feature selection. In the third subsection (4.1.3) we analyze the performance of VBLS on *irrelevancy, redundancy, computational efficiency* from the experiment results.

4.1.1 Artificial regression data set

The set of experiments was designed to test the performance of VBLS algorithm on data sets with irrelevant and redundant inputs. Each data set consisted of N points of d-dimensional input data $\mathbf{X}$ $[N \times d]$ and 1-dimensional output data $\mathbf{y}$ $[N \times 1]$. Input data $\mathbf{X}$ contain irrelevant (d_{ir}), relevant (d_{rel}), redundant (d_{red}) dimensions and generated as follows:

1. **Input data X**: Generate N normally distributed vectors of dimensionality $d_{ir} + d_{rel} < d$. $[N \times (d_{ir} + d_{rel})]$
2. **Regression vector b**: Generate a d_{ir} dimensional zero vector. $[d_{ir} \times 1]$
3. **Regression vector b**: Generate a d_{rel} dimensional normally distributed vector with zero mean and variance 100. $[d_{rel} \times 1]$
4. **Regression vector b**: Combine d_{ir} dimensional zero vector with d_{rel} dimensional normally distributed vector in order to get $d_{ir} + d_{rel}$ dimensional regression vector $\mathbf{b}$. $[(d_{ir} + d_{rel}) \times 1]$
5. **Output data y**: Generate output data using $d_{ir} + d_{rel}$ dimensional regression vector $\mathbf{b}$. Add Gaussian noise to output data (signal-to-noise SNR=10).$[N \times 1]$

6. **Input data X**: Pad the input data with d_{red} zero dimensions. $(d_{red} = d - (d_{ir} + d_{rel}))$ $[N \times d]$

An independent test set was generated in the same manner, but the output noise was zero in this case. In all experiments, inputs and outputs were scaled to zero mean and unit variance after the data generation.

4.1.2 Feature selection based on t-test

As explained in previous chapter we have placed individual precision variables α_m over each of the regression parameters b_m. This model structure was given by the following set of prior distributions:

$$\begin{aligned} p(\mathbf{b}|\boldsymbol{\alpha}) &\sim \prod_{m=1}^{d} \mathcal{N}(b_m; 0, 1/\alpha_m) \\ p(\boldsymbol{\alpha}) &\sim \prod_{m=1}^{d} \text{Gamma}(\alpha_m; a_{\alpha_m}, b_{\alpha_m}) \end{aligned} \tag{4.1}$$

where a_{α_m} and b_{α_m} were Gamma distribution variables. If we have a Gaussian $\mathcal{N}(x; \mu, 1/\alpha)$ together with a Gamma prior Gamma$(\alpha; a, b)$ and we integrate out the precision we obtain the distribution $t_\nu(x; \mu, \lambda, \nu)$ called Student's t-distribution. The parameter $\lambda = a/b$ is sometimes called the *precision* of the t-distribution, even though it is not in general equal to the inverse of the variance. The parameter $\nu = 2a$ is called the *degrees of freedom*, and its effect is illustrated in figure 4.1.

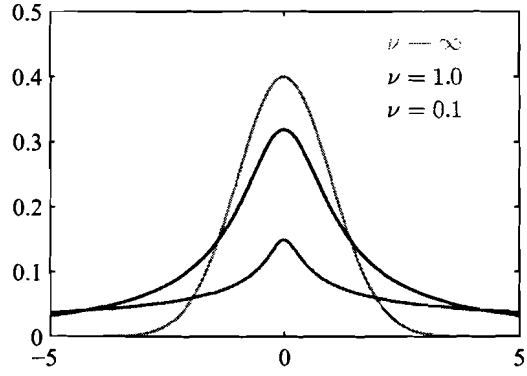


Figure 4.1: *Plot of Student's t-distribution for $\mu = 0$ and $\lambda = 1$ for various values of ν. For the particular case of $\nu \to \infty$ the t-distribution becomes a Gaussian with mean μ and precision λ.*

The given model structure in 4.1 allowed us to derive joint posterior probabilities over $\mathbf{b}, \boldsymbol{\alpha}, \mathbf{Z}$ but the intractability to determine marginal posterior probabilities of interest such as $p(\mathbf{b})$ enforced us to do approximate inference. Here we used a technique called factorial variational approximation in the form $Q(\mathbf{b}, \boldsymbol{\alpha}, \mathbf{Z}) = Q(\mathbf{b}, \boldsymbol{\alpha})Q(\mathbf{Z})$. From this approximation we derived the marginal distributions for regression vector $\mathbf{b}$ by integrating out the precision $\boldsymbol{\alpha}$. The obtained distributions takes the form of product of Student-t distributions and is given by

$$Q(\mathbf{b}) = \prod_{m=1}^{d} t_\nu(b_m; \mu_{b_m}, \sigma^2_{b_m}) \tag{4.2}$$

We also derived in previous chapter the marginal posterior distributions over $\boldsymbol{\alpha}$ integrating out $\mathbf{b}$ and is given by

$$Q(\boldsymbol{\alpha}) = \prod_{m=1}^{d} \text{Gamma}(\alpha_m; \hat{a}_\alpha, \hat{b}_\alpha^{(m)}) \tag{4.3}$$

We know that marginal posterior distribution over each b_m is a Student-t distribution. This property allows us to apply the one sample t-test. As explained before irrelevant and redundant dimensions will have small mean values $\langle \mu_{bm} \rangle$ and large precision values $\langle \alpha_m \rangle = \hat{a}_\alpha / \hat{b}_\alpha^{(m)}$, so the posterior distributions over b_m will be narrow around zero. The significance of a posterior mean value unequal zero can be tested directly via a t-test on the posterior means $\langle \mu_{bm} \rangle$, since the marginal posterior distribution over coefficients b_m is a factorial t-distribution. After applying the two sided t-test in VBLS we are left with a fully automatic feature selection method, where the only remaining hyperparameter $\hat{a}_\alpha$ are the level of the t-test and the convergence criteria for the VBEM loop.

The one sample t-test can either be a one- or two-tailed depending on the nature of the variable under study and the objectives. A one- or two-tailed t-test is determined by whether the total area of δ is placed in one tail or divided equally between the two tails. The one-tailed t-test is performed if the results are interesting only if they turn out in a particular direction. The two-tailed t-test is performed if the results would be interesting in either direction. The choice of a one- or two-tailed t-test effects the hypothesis testing procedure. In this report we will use two tailed t-test because, posterior mean values $\langle \mu_{bm} \rangle$ can possibly be negative or positive. A two-tailed t-test divides δ in half, placing half in the each tail. The null hypothesis in this case is a particular value, and there are two alternative hypotheses, one positive and one negative. The critical value of t, *tcrit*, is written with both a plus and minus sign ($\pm$). For example, the critical value of t when there are ten *degrees of freedom* ($\nu = 10$) and δ is set to 0.05, is $t_{crit} = \pm 2.228$. The sampling distribution model used in a two-tailed t-test is illustrated in figure 4.2.

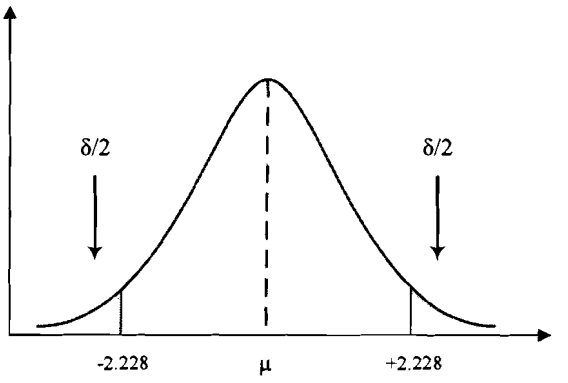


Figure 4.2: *Two sided t-test. The critical value* $t_{crit} = \pm 2.228$ *for* $\nu = 10$ *and* $\delta = 0.05$.

First we have to develop a null hypothesis H_0 which is the logical counterpart to an alternative hypothesis H_1. Depending on the outcome of the t-test, the null hypothesis is either rejected or retained. Rejection of the null hypothesis logically leads to acceptance of the alternative hypothesis. Two-tailed hypotheses take the form $H_0 : \mu = \mu_0$ and $H_1 : \mu \neq \mu_0$. The test statistic is used to evaluate the null hypothesis which then allows evaluation of the alternative hypothesis. The computed test statistic t_{stat} is given by

$$t_{stat} = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}} \tag{4.4}$$

The calculated test statistic has a Students t-distribution with $\nu = n-1$ degrees of freedom, where

$$\begin{aligned} \bar{X} &= \frac{1}{n}\sum_{i=1}^{n} X_i \\ \sigma &= \sqrt{\sum_{i=1}^{n} \frac{(X_i - \bar{X})^2}{n-1}} \end{aligned} \tag{4.5}$$

$\bar{X}$ is the sample mean and σ is the standard deviation. We reject the null hypothesis H_0 when t_{stat} is extreme enough to cause the observed significance p to be less than pre-specified significance level $\delta/2$ ($p < \delta/2$) . We retain the null hypothesis when the observed significance p is greater than the pre-specified significance level $\delta/2$ ($p > \delta/2$). The observed significance is the probability of obtaining a test statistic.

Now we can build in two sided one sample t-test in VBLS algorithm in order to do automatic feature selection. The marginal distribution of all b_m was t-distributed $t_\nu(b_m; \mu_{b_m}, \sigma^2_{b_m})$ with mean $\langle \mu_{bm} \rangle$, variance $\sigma^2_{b_m}$, $2\hat{a}_\alpha$ degrees of freedom and specified significance level of $\delta = 0.05$, which allows a principled way of determining whether a regression coefficient was excluded by means of standard hypothesis testing. Substituting the obtained parameters from VBLS into 4.4 gives

$$t_{crit} = \frac{\langle \mu_{b_m} \rangle - 0}{\sigma_{b_m}/\sqrt{2\hat{a}_\alpha + 1}} \tag{4.6}$$

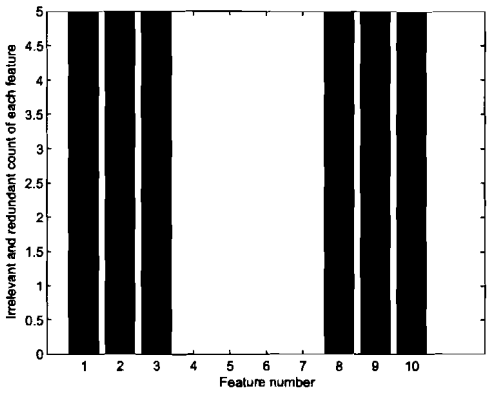
Two-tailed hypotheses take the form $H_0 : \mu = 0$ and $H_1 : \mu \neq 0$. Acceptance of hypotheses H_0 means that corresponding b_m has a zero mean value. The feature that belongs to this coefficient do not contribute to output.

4.1.3 Experiments and results

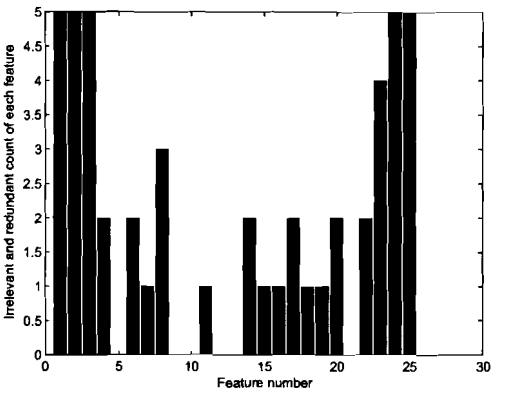
In this subsection we compare the performance of Variational Bayesian Least Square (VBLS) algorithm on training and test data sets with uncorrelated irrelevant and redundant features. For each experiment: *Labeling error*, *Normalized prediction error on test data* and *Computational efficiency* will be determined.

In the experiments for the training of VBLS model we varied the number of samples N as $N = [100, 500, 1000]$ and studied three dimensionalities $d = [10, 25, 50]$. We repeated 5 runs of each feature selection experiment (with each time a new draw of the data). VBLS algorithm was trained for 50000 cycles maximum or when the likelihood was improving less than 1e-3. In the experiments, $d_{ir} = 3$ and $d_{red} = 3$, so the first three features were irrelevant and last three features redundant for predicting output, and all other features were relevant.

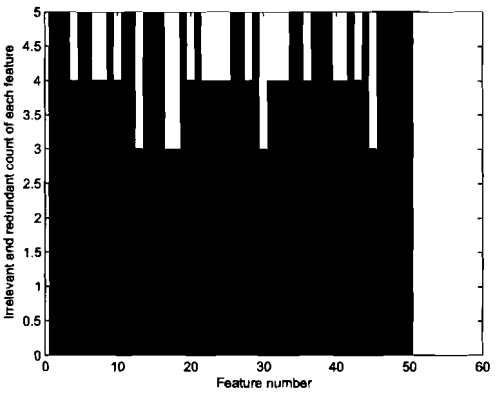
First we presented the irrelevant and redundant counts of each feature as function of feature number for a given sample size and dimension (N, d). Figure 4.3 shows experiments on three artificial regression data sets $(100, 10)$, $(100, 25)$ and $(100, 50)$. As we can see in figure 4.3 the performance of VBLS on three data sets is different. For data set $(100, 10)$ in figure 4.3 (a) we see that at each run the first three features are always detected as irrelevant and last three features as redundant. With increasing number of dimensions the detection of irrelevant and redundant features become inaccurate. This effect can be clearly seen from the training results of data sets $(100, 25)$ and $(100, 50)$ and is shown in figure 4.3(b) and (c). Number of iterations for the convergence of log-likelihood functions also increases with increasing number of dimensions. For the training of data set $(100, 10)$ we need 15371 iterations until convergence is reached. For data sets $(100, 25)$ and $(100, 50)$ number of iterations are 17261 and 23391. All obtained log-likelihood functions of experiments are monotonically increasing functions.



(a) $(N = 100, d = 10)$



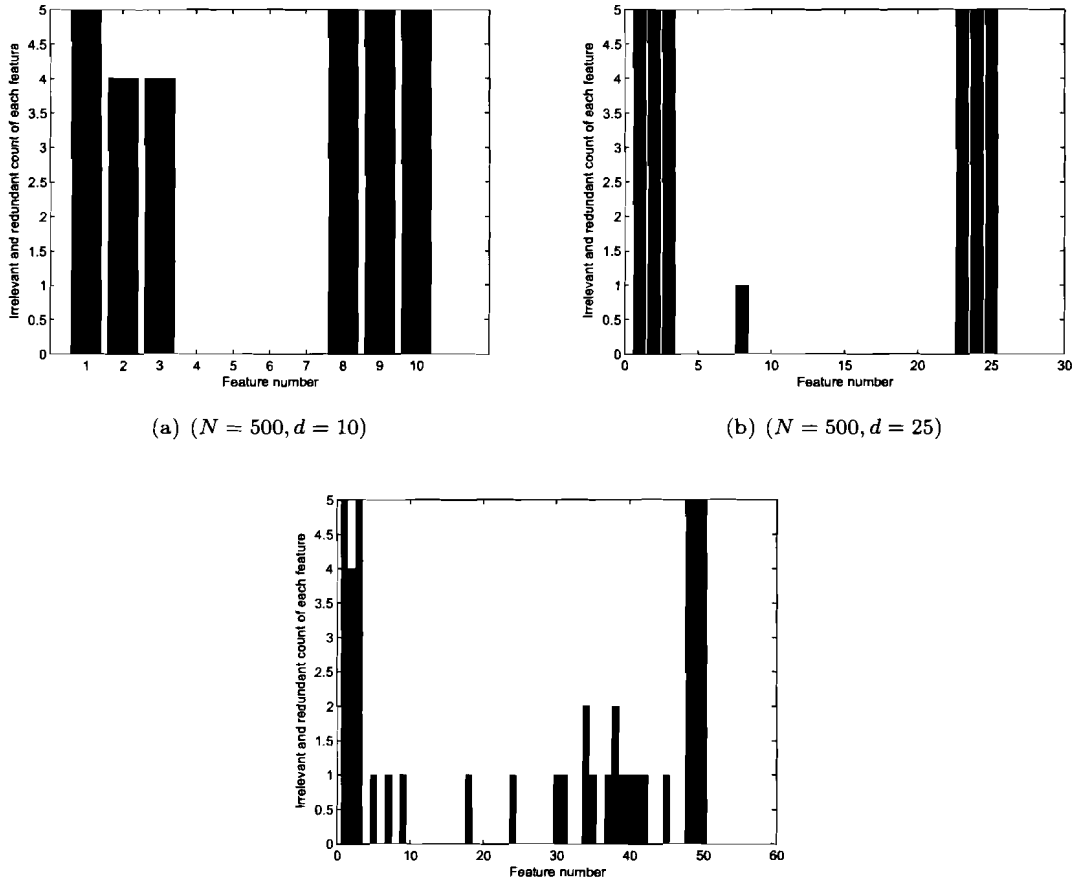
(b) $(N = 100, d = 25)$



(c) $(N = 100, d = 50)$

Figure 4.3: *In the experiments, $d_{ir} = 3$ and $d_{red} = 3$, so the first three features were irrelevant and last three features redundant for predicting output. Each graph shows irrelevant and redundant count of each feature in 5 runs. With increasing number of dimensions the detection of irrelevant and redundant features become inaccurate.*

Now number of samples are set to 500. We have trained the data sets $(500, 10)$, $(500, 25)$ and $(500, 50)$ in order to get better understanding of the effects of increasing number of samples on detection of irrelevant and redundant features. In figure 4.4 we see that detection of irrelevant en redundant features become more accurate. The training result of data set $(500, 10)$ in figure 4.4 (a) is almost the same as in the previous training with data set $(100, 10)$. This means that VBLS performance for $d = 10$ is reasonably good for minimum sample size of 100. Increasing of dimension d influences the performance of VBLS negatively. The training result of data set $(500, 25)$ in figure 4.4 (b) shows that performance of VBLS is much better than the training result of data set $(100, 25)$ in figure 4.3 (b), so we have to train more samples for dimension $d = 25$. We can conclude that we need minimum 500 samples to do accurate feature selection for $d = 25$. The effect of increasing number of samples on feature selection become more clear if we look at the training result of data set $(500, 50)$ in figure 4.4 (c). If we compare this result with previous training with data set $(100, 50)$ in figure 4.3 (c) the performance is better but not good enough. Therefore we increases the number of samples again in order to obtain an accurate feature selection for $d = 50$. For data set $(500, 10)$ we need 47131 iterations, for data set $(500, 25)$ 49191 iterations and finally for data set $(500, 50)$ 50000 iterations.



(a) $(N = 500, d = 10)$

(b) $(N = 500, d = 25)$

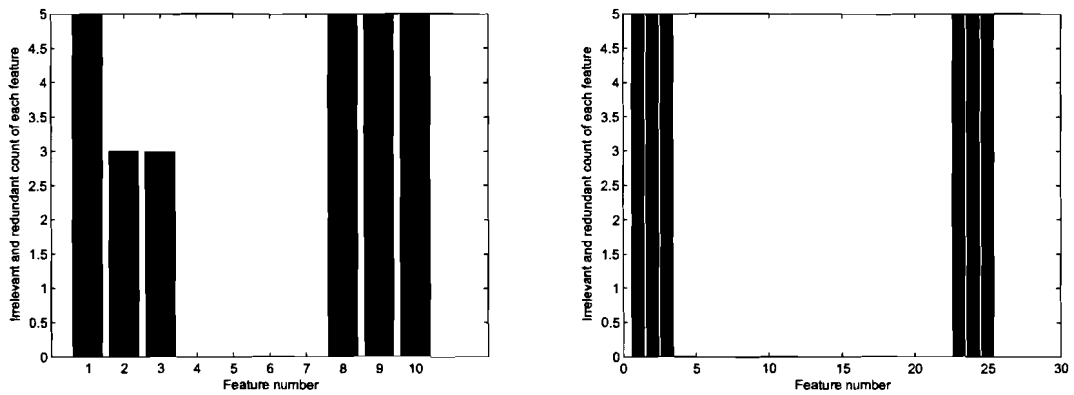
(c) $(N = 500, d = 50)$

Figure 4.4: *Detection of irrelevant and redundant features. VBLS performance for* $d = 10$ *and* $d = 25$ *is reasonably good for sample size 500, but for* $d = 50$ *VBLS do not provide accurate enough detections of irrelevant and redundant features.*

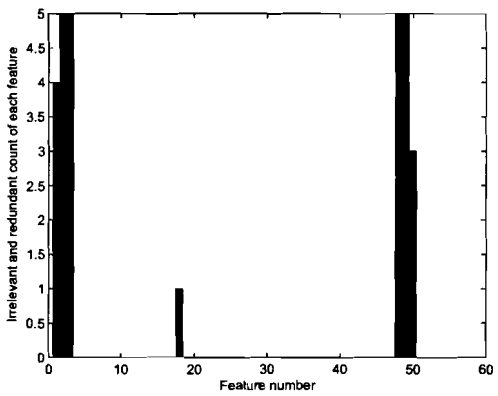
As mentioned in previous experiments for $N = 100$ and $N = 500$ the VBLS algorithm do not provide accurate enough detections of irrelevant and redundant features for dimension $d = 50$. If we set the number of samples to $N = 1000$ and train the data sets $(1000, 10)$, $(1000, 25)$ and $(1000, 50)$ we obtain following results given in figure 4.5. As expected the training results of data sets $(1000, 10)$, $(1000, 25)$ in figure 4.5 (a) and (b) are very accurate. VBLS algorithm also provide a good training result for the data set $(1000, 50)$ in figure 4.5 (c) in comparison with data set $(500, 50)$ in figure 4.4 (c). From this result we can conclude that for data sets with $d = 50$ dimensions we need minimum $N = 1000$ samples to do an accurate feature selection. For all data sets with $N = 1000$ log-likelihood functions convergence after 50000 iterations.

Until now we have trained data sets with dimensions $d = 10$, $d = 25$ and $d = 50$ at 3 different sample sizes $N = [100, 500, 1000]$. From presented 9 experiment results we can conclude that with increasing number of dimensions determination of irrelevant and redundant features become computationally expensive. As explained before we need more samples to do accurate feature selection for higher dimensions. Training process of data sets with large number of samples and high dimensions takes a lot of time. Therefore we have tried to find the minimum required number of samples for a given number of dimension.

For 10-dimensional data set we need minimum 100 samples to do an accurate feature selection. Needed number of samples for 25-dimensional data set is 500 and for 50-dimensional data set 1000. This values are only valid under the condition that SNR is 10. Higher SNR values add more uncertainty in data and makes training process more expensive.



(a) $(N = 1000, d = 10)$ (b) $(N = 1000, d = 25)$



(c) $(N = 1000, d = 50)$

Figure 4.5: *VBLS provide for all dimensions accurate enough detections of irrelevant and redundant features. For data sets with $d = 50$ the minimum required number of samples are 1000.*

Now we will present in a graph the *labeling error* as a function of log sample size N in order to get better understanding of the performance of VBLS on data sets with different dimensions and sample sizes. For *labeling error* we counted the total number of mislabeling of a feature and normalized by the total number of features present in 5 runs. The mean labeling results over 5 runs are shown in figure 4.6. We see that for $d = 10$ the labeling error varies not much with increasing number of samples, the red line $(\cdots)$ is nearly constant. The detection of all irrelevant and redundant features have already been realized by sample size $N = 100$. Therefore increasing of number of samples will not change the labeling error very much. For $d = 25$ $(--)$ irrelevant and redundant features are detected correctly by sample size $N = 500$. Until this point labeling error decreases linearly. After $N = 500$ samples the labeling error is nearly constant. The labeling error for $d = 50$ $(\cdot - \cdot)$ decreases enormously because of the accurate detection of irrelevant and redundant features takes place by higher sample sizes. After $N = 1000$ samples are reached the labeling error stays constant.

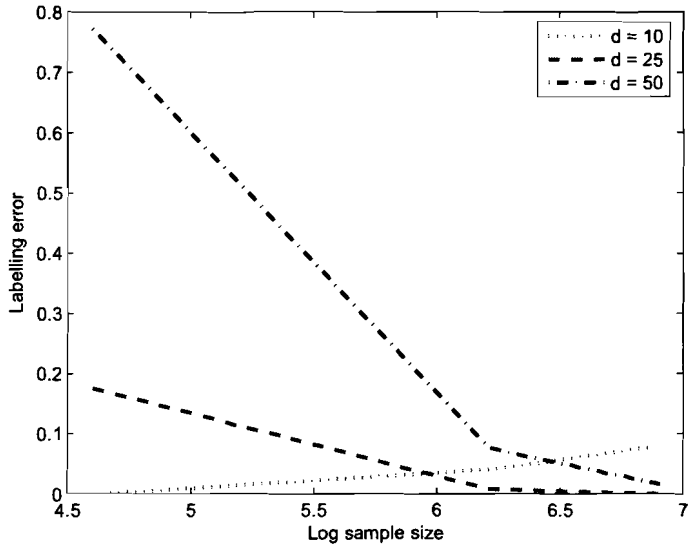


Figure 4.6: *Mean labeling error versus log sample size for* $d = [10, 25, 50]$. *We counted the total number of mislabeling of a feature and normalized by the total number of features present in 5 runs.*

We also calculated the mean prediction error on the test set generated as explained in subsection 4.1.1. As shown in figure 4.7 the prediction error decreases with increasing number of samples N. Data sets with high dimensions have a large prediction error because each dimension contributes a little to the prediction error.

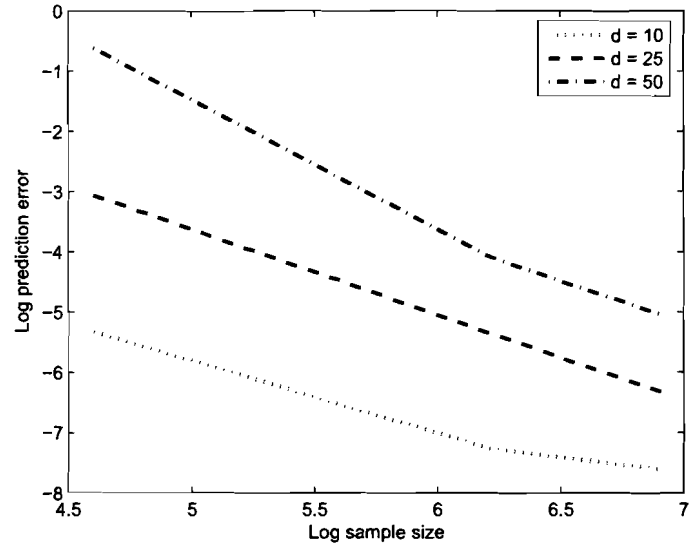


Figure 4.7: *Mean prediction error on the test set versus log sample size for* $d = [10, 25, 50]$. *The mean prediction error is computed on the test sets generated without output noise.*

Each iteration step in the evaluation of VBLS requires many matrix operations. The amount of operations increases with increasing number of dimensions and samples. We estimated the computational cost of each training data set (Matlab tic, toc) that determines the performance of VBLS method on computational efficiency. Running time is linearly dependent on the computational complexity and is shown in figure 4.8 as function of log sample size for $d = [10, 25, 50]$.

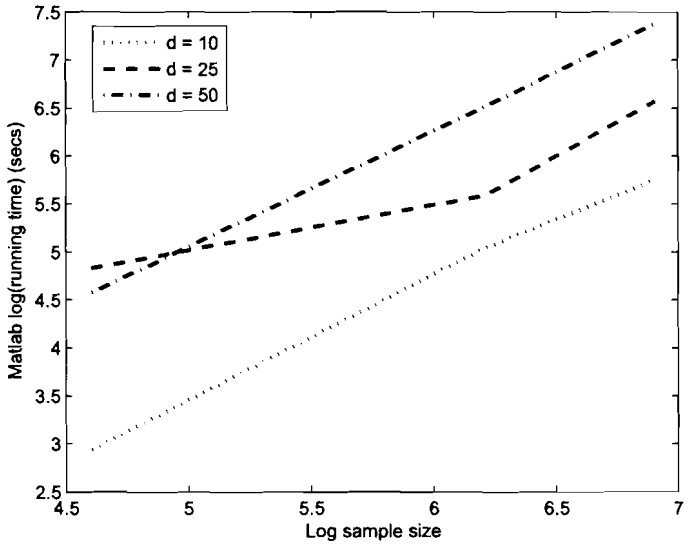


Figure 4.8: *Log mean running time versus log sample size for $d = [10, 25, 50]$. We estimated the computational cost of each training data set (Matlab tic, toc) that determines the performance of VBLS method on computational efficiency.*

The red line $(\cdots)$ belongs to dimension $d = 10$ and is below other lines, namely the green $(--)$ one with $d = 25$ and blue $(- \cdot -)$ one with $d = 50$. For higher dimensions the log running time is increasing enormously with increasing number of samples.

We finished the experiments section on artificial regression data sets. We showed with experiments that VBLS can handle three of the five mentioned points in subsection 1.3.2 namely, irrelevancy, redundancy and computational efficiency.

VBLS method is able to determine the irrelevant and redundant dimensions under certain conditions. It is very important that we offer the required minimum number of samples N as input in VBLS to do an accurate feature selection for a certain number of dimension d. Otherwise the VBLS will not perform the optimal performance on observed data. The computational efficiency decreases with increasing number of dimensions. Large dimensionality means minimum required number of samples is high. The occasional improved accuracy of VBLS comes however at the expense of much larger training times.

For probabilistic embedding we can determine with VBLS posterior distributions over regression coefficients using the inferred parameter posteriors. From inferred posterior distributions we can compute predictive distributions over the output. VBLS method is applicable for sequential processing. There are many methods available in literature to extend an EM based batch algorithm into a sequential algorithm. We are not going to discuss sequential algorithm in detail. For detailed information about EM based sequential algorithms the reader is referred to Weinstein et al. [1988] and Neal and Hinton [1998].

4.2 Experiments on preference data

In this section we apply the VBLS method to real world data sets collected with Real-time simulation platform. This section consists of three subsections. In the first subsection (4.2.1) a short explanation is given about working of Real-time simulation platform (RTProc). In the second subsection (4.2.2) we described the properties of used real world data sets in detail. The last subsection (4.2.3) describes results obtained for VBLS as a regression and feature selection method.

4.2.1 Real-time simulation platform

The Real-time simulation platform (RTProc) is a system that uses a PC for real-time audio processing [Geurts, 2005]. To simplify the development of a new real-time audio processing algorithm, this system is separated in three functional modules,the driver, the algorithm and the host module.

The driver module obtains in real-time, buffer for buffer, the input audio samples from a selected audio device and then converts these samples into a hardware independent data format such as double or float. These buffers are passed to the host module and from there to the algorithm module. The algorithm modifies the input signal by applying an algorithm and the resulting output signal is passed back to the host and then to the driver. The driver converts the output signal back to the hardware specific data format and this signal is then played on he selected audio device. The host module takes care of the communication between the driver module and the algorithm module. Clearly defined Application Programmable Interfaces (API) make this communication possible. The host is integrated into Matlab and can be operated by means of a number of commands in the command window.

The user can select a audio device, after selection this device must return information on its capabilities, sample rate, buffer size, input and output channels and processing format. Finally, for processing the device must be informed to start processing. For a more detailed overview of the working of RTProc system, the reader is referred to Geurts [2005].

4.2.2 Experimental setup

As explained in Chapter1 real-time simulation platform (RTProc) is used to collect real world data. We implemented a hearing aid algorithm on a real-time platform and made one of the processing parameters of the virtual hearing aid on-line modifiable by the user. Six normal hearing subjects were exposed in a lab trial to an acoustic stimulus that consisted of several speech and noise snapshots picked from a database (each snapshot typically in the order of 10 seconds) which were combined in several ratios and appended. This led to one long stream of signal/noise episodes with different types of signals and noise in different ratios. The subjects were asked to listen to this stream several times in a row, and adjust the processing parameter as desired. Each time an adjustment was made, the acoustic input vector and the desired hearing aid processing parameter were stored. At the end of an experiment a set of input-output pairs was obtained from which a regression model could be inferred using off-line training.

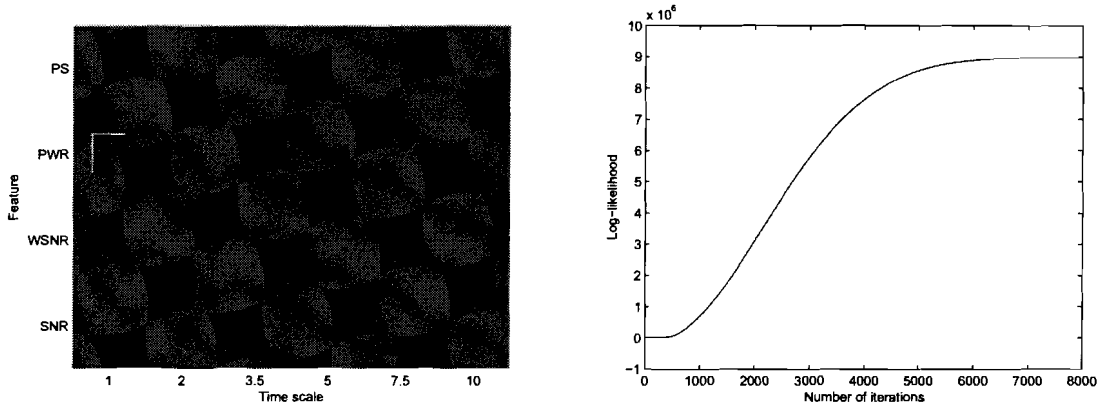
The acoustic input vector $\{\mathbf{x_n}\}_{n=1}^{N}$ contain 19 features for each adjustment moment n (consent moment). Consent moment occurs when the user has stopped with adjustment, at this time user has found satisfying tuning parameters $\{y_n\}_{n=1}^{N}$ (desired output). There are 3 features for 6 different time scales $[1, 2, 3.5, 5, 7.5, 10]$ in seconds. These features are Power (PWR), Weighted Signal to Noise Ratio (WSNR) and Signal to Noise Ratio (SNR). There is also one feature called Power Spectrum(PS) which is the same for all time scales. The desired output y_n is labeled as PM. After N adjustments input-output pairs $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ are used for off-line training with VBLS. The acoustic input vector dimensions with corresponding feature labels and the desired output y_n are listed in table 4.1.

Table 4.1: *Acoustic input vector dimensions x_{nd} (d=1,.,19) and output y_n with corresponding labels.*

Dimension(1-10)	Feature label	Dimension(11-19) and Output y	Feature label
x_{n1}	PS	x_{n11}	WSNR5
x_{n2}	PWR1	x_{n12}	WSNR75
x_{n3}	PWR2	x_{n13}	WSNR10
x_{n4}	PWR35	x_{n14}	SNR1
x_{n5}	PWR5	x_{n15}	SNR2
x_{n6}	PWR75	x_{n16}	SNR35
x_{n7}	PWR10	x_{n17}	SNR5
x_{n8}	WSNR1	x_{n18}	SNR75
x_{n9}	WSNR2	x_{n19}	SNR10
x_{n10}	WSNR35	y_n	PM

4.2.3 Experiments and results

We postulated that two types of features bear information about the user preference. Feature PS was expected to be correlated with feature WSNR and SNR whereas feature PWR was expected to be very different. In a set listening experiments with six normal hearing subjects, features PWR,WSNR and SNR were computed at 6 different time scales. Feature PS was the same for all time scales. The acoustic stream was presented more often (in a looping mode). The number of adjustments for each of the subjects 1 to 6 was $N = [43, 275, 703, 262, 99, 1020]$. This means that we are in the realm of moderate sample size and moderate dimensionality, for which VBLS is accurate. We trained VBLS on the six data sets. For all subjects computed posterior mean values for the variance $1/\langle\alpha_m\rangle$ are shown in a Hinton diagram and the corresponding log likelihood functions. As described in previous chapter Log likelihood function is a monotonically increasing function if the underlying model is linear. Figure 4.9(a) shows Hinton diagram (a) and the log-likelihood function (b) for subject 1.



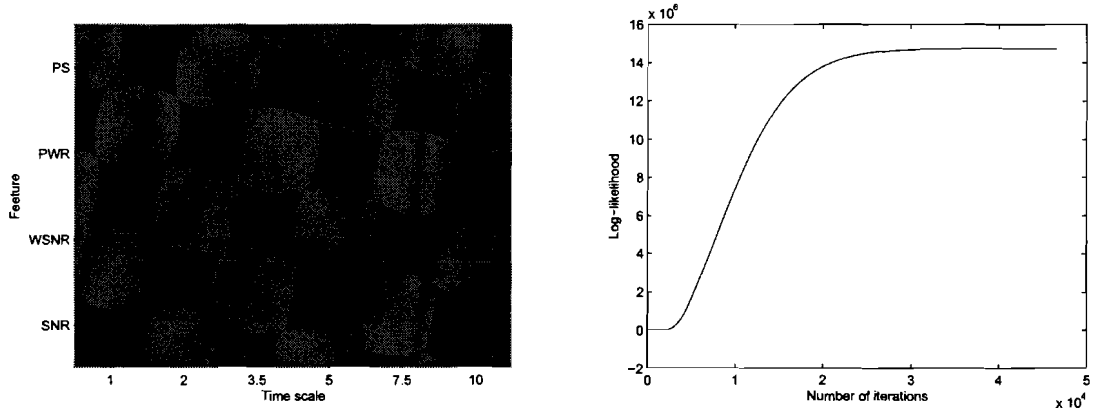
(a) *Hinton diagram*

(b) *Log-likelihood function*

Figure 4.9: *(a) ARD-based selection of hearing aid features. Shown is a Hinton diagram of $1/\langle\alpha_m\rangle$, computed from subject 1 preference data. Box size denotes relevance.(b) The log-likelihood function convergence after 8000 iterations.*

We see that subject 1 adjust only the hearing aid parameter PWR on the shortest time scale. The other features PS, WSNR en SNR for all time scales contribute not to the output.

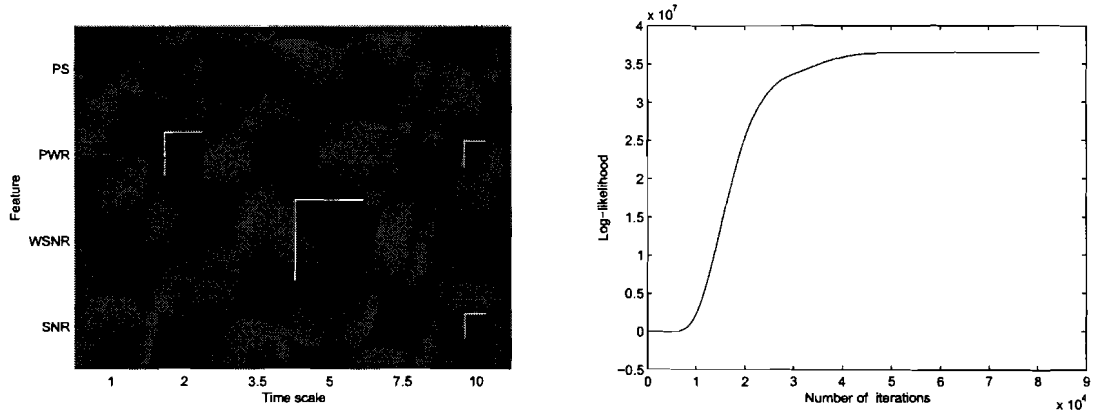
Figure 4.10 (a) shows an empty Hinton diagram for Subject 2. This means there are no relevant features that can explain the user behavior. The computed features for all time scales have variance values $1/\langle\alpha_m\rangle$ almost zero. Subject 2 shows an unreliable model fit. This result can be possibly related to the problem that the underlying model for user behavior is not linear.



(a) *Hinton diagram* (b) *Log-likelihood function*

Figure 4.10: *(a) Hinton diagram computed from subject 2 preference data. All features do not contribute to the output. (b) The log-likelihood function convergence after 40000 iterations.*

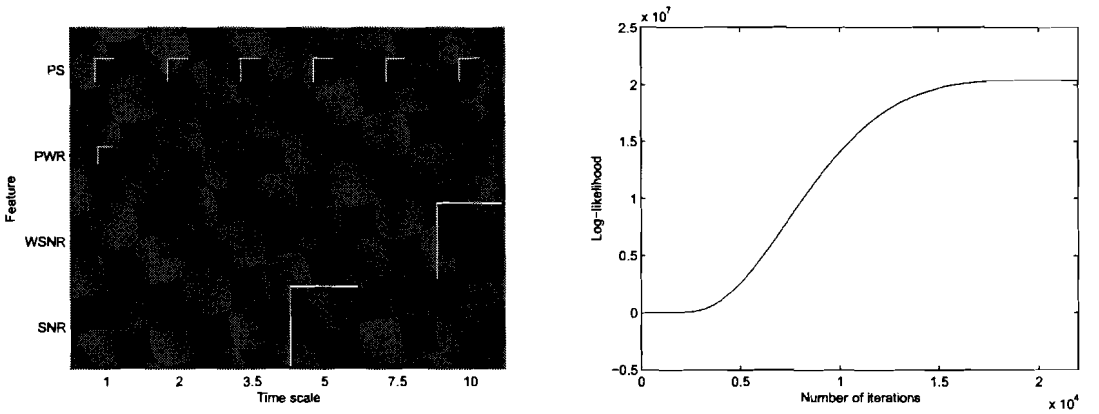
As shown in figure 4.11 (a) subject 3 adjust the hearing aid parameters separately. The PS feature is the only one that not contribute to the output. Other 3 features PWR, WSNR and SNR plays an important role in the user behavior. Features WSNR and SNR contribute to the output in the area with large time scale. Feature PWR has contribution in both small and large time scales.



(a) *Hinton diagram* (b) *Log-likelihood function*

Figure 4.11: *(a) Hinton diagram computed from subject 3 preference data. Feature PS do not contribute to the output. Subject 3 adjust the hearing aid parameters primarily based on features PWR, WSNR and SNR. (b) The log-likelihood function convergence after 70000 iterations.*

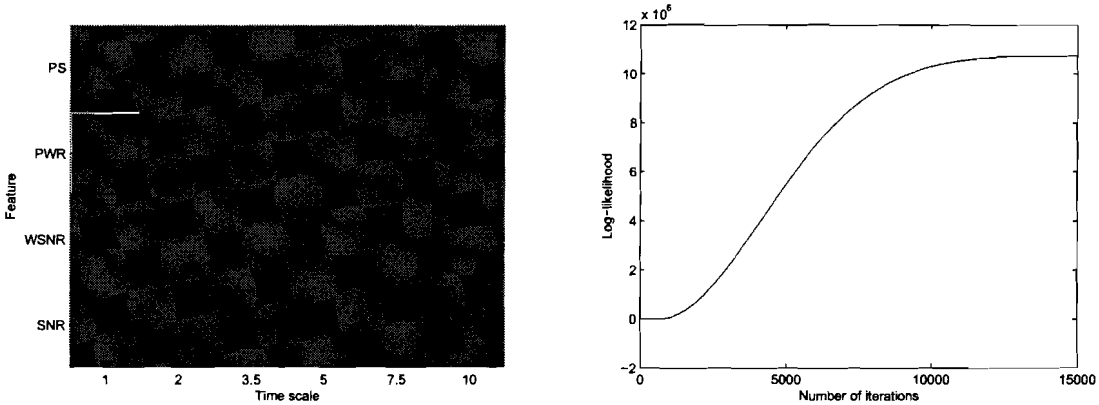
For subject 4 all features contribute to the output. In figure 4.12 (a) features WSNR and SNR have large box sizes in Hinton diagram then the features PS and PWR. This means that WSNR and SNR have more contribution in the user behavior. These two features contribute to the output in the area with large time scale. For feature PS the contribution is the same for all time scales and for PWR contribution is in the short time scale.



(a) *Hinton diagram* (b) *Log-likelihood function*

Figure 4.12: *(a) Hinton diagram computed from subject 4 preference data. All features contribute to the output. (b) The log-likelihood function convergence after 20000 iterations.*

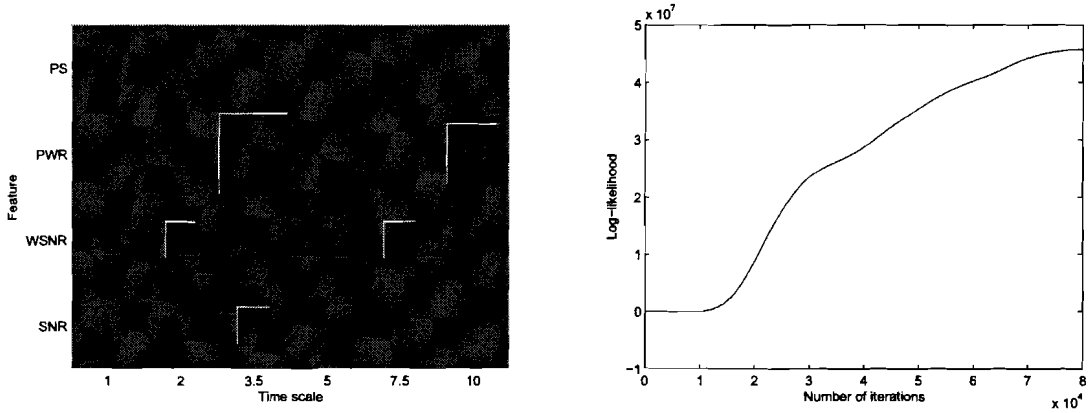
Figure 4.13 (a) shows Hinton diagram for Subject 5. For subject 5 only feature PWR contribute to the output in a short time scale. The other features have posterior mean values for the variance $1/\langle\alpha_m\rangle$ almost zero. The log-likelihood function of preference data set for subject 5 convergence after 15000 iterations.



(a) *Hinton diagram* (b) *Log-likelihood function*

Figure 4.13: *(a) Hinton diagram computed from subject 5 preference data. Only feature PWR in short time scale contribute to the output. Features WSNR and SNR are irrelevant. (b) The log-likelihood function convergence after 15000 iterations.*

As shown in figure 4.14 (a) subject 6 adjust the hearing aid parameters separately. The PS feature is the only one that not contribute to the output. Feature SNR contribute to the output in the area with small time scale. Features PWR and WSNR have contribution in both small and large time scales.



(a) *Hinton diagram*

(b) *Log-likelihood function*

Figure 4.14: *(a) Hinton diagram computed from subject 6 preference data. Only feature PS do not contribute to the output. (b) The log-likelihood function convergence after 80000 iterations.*

4.3 Conclusions

From our artificial regression data experiments (section 4.1), we can conclude that VBLS is a useful method for doing accurate regression and feature selection at the same time. For low dimensionalities VBLS yields useful results if we vary the sample sizes from low to high and computation time is not an issue. When we increase the dimensions required number of samples are increases for doing accurate regression and feature selection. Labeling and prediction errors decreases with higher sample sizes if we fix the dimension at a certain value but required number of iterations increases in order to reach convergence which take more computing time. Beside doing accurate regression and feature selection we can compute with VBLS a predictive distribution using the inferred parameter posteriors, allowing for a prediction with confidence levels.

From our preference data experiments (section 4.2), we noted that 4 out of 6 subjects personalized the hearing aid parameter based on (some of the) features PWR and WSNR, which seem a good choice for inclusion in an on-line learning algorithm. For one of the users, either the sample size was too low, his preference was too noisy or the linearity assumption of the model might not hold. In the future, an experimental setup with a different acoustic stimulus might be considered to possibly diminish the noise in the preference data.

From our experiments, we concluded that VBLS does well at feature selection under the condition that a minimum number of samples is available for training (*Irrelevancy and Redundancy*). Data sets with large sample sizes and dimensions takes longer training time because of required computations increases (*Computational efficiency*). Therefore VBLS is a useful method for doing accurate regression and feature selection when we use it off-line. As with all Bayesian algorithms, as the size of the training data decreases, the quality of the algorithm's predictive performance will degrade. This is due to the fact that if there is very little data to learn from, then the algorithm cannot be expected to learn much from it. One further advantage of the VBLS method over

the other analyzed methods is that one can compute a predictive distribution using the inferred parameter posteriors, allowing for a prediction with confidence levels (*Probabilistic embedding*). We can also extend VBLS algorithm into a sequential algorithm [Weinstein et al., 1988]; [Neal and Hinton, 1998] (*Adaptivity*).

Chapter 5

Conclusions and recommendations

5.1 Conclusions

The goal of this project was to evaluate methods for finding optimal linear models that can control the user preferred parameters in the adaptive hearing aid when the user preferences change. The underlying model that controls the adaptive hearing aid is based on the assumption that user preferred steering signal is linearly dependent on acoustic environmental features. Under this assumption the project focused on analysis of a few properties of a multiple linear regression problem. These properties were related to following difficulties in the project: The vector of acoustic environmental features were likely high dimensional and contain features that not contribute to the output, computational complexity was an issue because of high dimensional behavior of the input. Therefore our goal in this project was to evaluate linear methods in order to get one that can handle with *irrelevancy, redundancy, computational complexity, probabilistic embedding* and *adaptivity.* We analyzed several methods by evaluating these on the afore mentioned points. These analysis leads to a deeper study for one specific algorithm, named Variational Bayesian Least Square (VBLS). The performance of this method was tested through experiments with artificial regression data and real data.

From our artificial regression data experiments, we concluded that in the data range from 500 to 1000 VBLS does well at feature selection under the condition that maximum number of dimensions is not larger then 50. The afore given data range is also the most usual range for off-line hearing aid data experiments. Data sets with larger sample sizes and dimensions takes longer training time because of required computations increases. From our preference data experiments, we noted that 4 out of 6 users personalized the hearing aid parameter based on (some of the) features PWR (Power) and WSNR (Weighted Signal to Noise Ratio), which seem a good choice for inclusion in an on-line learning algorithm. For one of the users, either the sample size was too low, his preference was too noisy or the linearity assumption of the model might not hold.

VBLS is an Bayesian method in a statistical framework. We computed probabilistic distributions over the regression parameters. This provide calculations of the predictive distributions which can be used to represent the uncertainty in the output. This means that it is possible to use decision theory for analysis, and choose in each situation the action, which maximizes the expected utility. VBLS is an EM extended batch algorithm. These means that VBLS is not adaptive and can only be used for off-line model predictions. In this report we used VBLS method for off-line simulations, but after some research we find out that methods exist that can possibly extend an VBLS based batch algorithm into a sequential algorithm. This research has not discussed in this report.

The advantage of VBLS is its numerical robustness in comparison with other methods discussed in this report. This is due to the fact that VBLS requires no matrix inversions. Each EM step

has linear computational complexity in the number of dimensions d which results in an algorithm which scales easily to high dimensional data sets. The VBLS method is a suitable method that can handle with *irrelevancy, redundancy, computational efficiency, probabilistic embedding* and therefore it is widely applicable over a certain range of data sets in off-line usage. The required large training times for high dimensional data sets provides that the VBLS method is not efficient for on-line (adaptive) usage. In table 5.1 we represented the advantages of VBLS method with respect to other analyzed method in chapter 2.

Table 5.1: *Evaluation results of described methods on irrelevancy, redundancy, computational efficiency, probabilistic embedding and adaptivity. The character d is the number of dimensions of the input matrix given that $K \ll d$. (−) can not handle, (+) good, (++) better.*

Criteria	OLS	LMS	RLS	PCR	JFR	VBLS
irrelevancy	−	−	−	+	++	++
redundancy	−	−	−	+	++	++
computational complexity	$O(d^3)$	$O(d)$	$O(d^2)$	$O(Kd^2)$	$O(K^3)$	$O(d)$
probabilistic embedding	−	−	−	−	−	++
adaptivity	−	+	++	−	−	−

5.2 Recommendations

Several assumptions were made to construct the VBLS method at the expense of large training times for high dimensional data sets. Further modifications to the model could enable it to be more accurate and efficient. Examples of potential modifications include:

- Several variants of the EM algorithm exist. To address computational issues [Neal and Hinton, 1998] have theoretically justified incremental and sparse variants, in which only a subset of the hidden variables and/or parameters are updated in each EM cycle.

- Prior to the use of variational approximations, a popular method for approximating intractable distributions has been the Laplace approximation (also known as the Gaussian approximation), which uses a second-order Taylor series expansion at the maximum of the distribution [Bishop, 1998].

From our data experiments, we concluded that VBLS does well at feature selection. Data sets with large sample sizes and dimensions takes longer training time because of required computations increases. Therefore VBLS is a useful method for doing accurate regression and feature selection when we use it off-line. One further advantage of the VBLS method is that one can compute a predictive distribution using the inferred parameter posteriors, allowing for a prediction with confidence levels. As with all Bayesian algorithms, as the size of the training data decreases, the quality of the algorithm's predictive performance will degrade. This is due to the fact that if there is very little data to learn from, then the algorithm cannot be expected to learn much from it.

The VBLS method presented in this report is a suitable method that can handle with *irrelevancy, redundancy, computational efficiency, probabilistic embedding* and therefore it is widely applicable over a certain range of data sets in off-line usage. The required large training times for high dimensional data sets provides that the VBLS method is not efficient for on-line (adaptive) usage. For that reason a further research is necessary to the VBLS in order to improve it on computational efficiency for sequential (on-line) usage Weinstein et al. [1988],Neal and Hinton [1998].

Bibliography

Beal MJ. Variational Algorithms for Approximate Bayesian Inference. Ph.D. thesis, The Gatsby Computational Neurosience Unit, University College London, 2003.

Bishop CM. Bayesian PCA. Proc. NIPS, 1998.

Catania AC. Learning. Prentice Hall, 3rd ed., 1992.

Corwin JT. Identifying the genes of hearing, deafness, and dysequilibrium. The National Academy of Sciences, 1998.

Dempster AP, Laird NM and Rubin DB. Maximum Likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, 1977.

Draper NR and Smith H. Applied Regression Analysis. Wiley, 1981.

D'Souza AA. Towards tractable parameter-free statistical learning. Ph.D. thesis, Univeristy of Southern California, 2004.

Everit BS. An Introduction to Latent Variable Models, Monographs on Statistics and Applied Probability. Chapman and Hall, 1984.

Geurts J. A PC-based real-time simulation platform for evaluating hearing aid algorithms, 2005. Practical training report, Eindhoven university of technology, GNResound.

Ghahramani Z and Beal MJ. Variational inference for Bayesian mixtures of factor analysers. Solla, Leen and Muller, pages 509–514, 2000.

Ghahramani Z and Hinton GE. The EM algorithm for mixtures of factor analyzers, 1997. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto.

Hastie T and Tibshirani R. Generalized Additive Models. Chapman and Hall, 1990.

Haykin S. Introduction to Adaptive Filters. Macmillan Publishing Company, 1985.

Haykin S. Adaptive Filter Theory. Prentice-Hall, 3th ed., 1996.

Hergenhahn BR. An introduction to theories of learning. Prentice Hall, 3rd ed., 1988.

Massey WF. Principal component regression in exploratory statistical research. Journal of the American Statistical Association 60, pages 234–246, 1965.

Mazur JE. Learning and behavior. Prentice Hall, 2nd ed., 1990.

Neal R and Hinton G. A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. Kluwer, 1998.

Shetty KK. A novel algorithm for uplink interference suppression using smart antennas in mobile communications. Master's thesis, Florida State University, 2004.

Ting J, D'Souza A and Schaal S. Bayesian Regression with Input Noise for High Dimensional Data. In International Conference on Machine Learning,. ICML, 2006.

Webster M. Merriam Webster's Collegiate dictionary. Merriam-Webster, 11th ed., 1999.

Weinstein E, Feder M and Oppenheim A. Sequential algorithm for parameter estimation based on the Kullback-Leibler information measure. IEEE Trans. Acoust., Speech, Signal Processing, 36, 477–489, 1988.

Widrow B and Hoff ME. Adaptive switching circuits. IRE WESCON Convention Record, 1960.

Wold H. Soft modeling by latent variables: The nonlinear iterative partial least squares approach. Academic Press, 1975.

Wu CFJ. On the convergence properties of the EM algorithm. The Annals of Statistics 11(1), pages 95–103, 1983.

Ypma A, de Vries B and Geurts J. Robust volume control personalisation from on-line preference feedback. 2006.

Appendix A

Derivations for EM algorithm

Several derivations have been done in the context of EM algorithm This appendix lists the proves of derivations used in chapter 3 of this report.

A.1 Derivation of the EM bound

For a special case lower bound $\mathcal{F}(Q, \phi)$ in equation (3.3) becomes an equality when the distribution $Q(\mathbf{x}_\mathcal{H}) = p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi)$. This can be proven by substituting $p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi)$ for Q in equation (3.3).

$$
\begin{aligned}
\mathcal{F}(Q, \phi) &= \int Q(\mathbf{x}_\mathcal{H}) \ln [\frac{p(\mathbf{x}_\mathcal{D}, \mathbf{x}_\mathcal{H}; \phi)}{Q(\mathbf{x}_\mathcal{H})}] d\mathbf{x}_\mathcal{H} \\
&= \int p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi) \ln [\frac{p(\mathbf{x}_\mathcal{D}, \mathbf{x}_\mathcal{H}; \phi)}{p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi)}] d\mathbf{x}_\mathcal{H} \\
&= \int p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi) \ln [\frac{p(\mathbf{x}_\mathcal{D}; \phi) p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi)}{p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi)}] d\mathbf{x}_\mathcal{H} \qquad \text{(A.1)} \\
&= \int p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi) \ln p(\mathbf{x}_\mathcal{D}; \phi) d\mathbf{x}_\mathcal{H} \\
&= \ln p(\mathbf{x}_\mathcal{D}; \phi) \int p(\mathbf{x}_\mathcal{H}|\mathbf{x}_\mathcal{D}; \phi) d\mathbf{x}_\mathcal{H} \\
&= \ln p(\mathbf{x}_\mathcal{D}; \phi)
\end{aligned}
$$

A.2 From Gauss-Seidel to probabilistic backfitting

We set $\omega = \psi_{zm}/s$ in Gauss-Seidel *succesive relaxation* algorithm (3.16).

$$
\begin{aligned}
b_m^{(n+1)} &= (1 - \frac{\psi_{zm}}{s})b_m^{(n)} + \frac{\psi_{zm}}{s} \frac{\sum_{n=1}^{N} \Big(y_n - \sum_{k \neq m}^{d} b_k^{(n)} f_k(\mathbf{x}_n)\Big) f_m(\mathbf{x}_n)}{\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2} \\
b_m^{(n+1)} &= b_m^{(n)} - \frac{\psi_{zm}}{s} b_m^{(n)} + \frac{\psi_{zm}}{s} \frac{\sum_{n=1}^{N} \Big(y_n - \sum_{k \neq m}^{d} b_k^{(n)} f_k(\mathbf{x}_n)\Big) f_m(\mathbf{x}_n)}{\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2} \\
b_m^{(n+1)} &= b_m^{(n)} + \frac{\psi_{zm}}{s} \frac{\sum_{n=1}^{N} \Big(y_n - \sum_{k=1}^{d} b_k^{(n)} f_k(\mathbf{x}_n)\Big) f_m(\mathbf{x}_n)}{\sum_{n=1}^{N} f_m(\mathbf{x}_n)^2} \qquad \text{(A.2)}
\end{aligned}
$$

The derived update equation (A.2) is exactly the same as in equation (3.14).

Appendix B

Matlab simulation files

B.1 sim_vbls.m

```
0001 % Simulation file for VBLS algorithm
0002
0003
0004 % Insert simulation values: Sample sizes and dimensions
0005 Ns    = [100 500 1000];
0006 Ms    = [10 25 50];
0007 SNRYs = [10];
0008
0009 % Start simulation
0010 for M=Ms
0011     M=M
0012     for N = Ns
0013          N=N
0014          if N>=M,
0015
0016          for SNRY = SNRYs
0017          SNRY=SNRY
0018
0019          % Clear values
0020          clear vbls_nmse TC TS rtimeVBLS w  B vbls_results
0021
0022 Mir     = 3;            % Number of irrelevant dimensions
0023 Mred    = 3;            % Number of redundant dimensions
0024 Mrel    = M-(Mir+Mred); % Number of relevant dimensions
0025
0026 % Number of runs of the algorithm
0027 Runs    = 5;
0028
0029 % Initialize testvector for labeling error
0030 TestVector = Runs*[ones(Mir,1);zeros(Mrel,1); ones(Mred,1)];
0031
0032 % Initialize the buffer variables
0033 L       = Runs;
0034 IrDim   = zeros(M,1);
0035 TS      = zeros(M,1);
0036 B       = zeros(M,1);
```

```
0037
0038 % Algorithm settings
0039 options.noise           = 0.005;        % Output noise variance
0040 options.threshold       = 1e-3;         % Threshold for convergence
0041 options.numIterations    = 50000;        % Number of iterations
0042
0043 for j=1:L,
0044     j
0045
0046     % Data generation function
0047     gen_data;
0048
0049     % Start timer
0050     tic
0051
0052     % Save observed data
0053     Data.Xtrain     = Xtrain;
0054     Data.Ytrain     = Ytrain;
0055     Data.Xtest      = Xtest;
0056     Data.Ytest      = Ytest;
0057
0058     % Save returned VBLS results
0059     vbls_result        = vbls(Xtrain, Ytrain, options);
0060     rtimeVBLS(j)        = toc;  % Stop timer and save
0061
0062     % Automatic feature selection based on t-test
0063     level           = 0.05;
0064     nu              = 2*vbls_result.a_alpha;
0065     Ts              = abs(vbls_result.b_mean) ./ sqrt(vbls_result.b_var);
0066     Tc               = my_tinv(1-level/2, nu);
0067     IrInp            = find(Ts<=Tc);
0068
0069     % Determine total number of irrelevant and redundant dimesions from t-test
0070     IrDim(:)        = IrDim(:) + ismember((1:M)',IrInp);
0071
0072     % Save t-statistcs and coefficients in buffer
0073     TS(:,j)         = Ts;
0074     TC(j)             = Tc;
0075     B(:,j)          = Btrue;
0076
0077     % Set irrelevant and redundant features manually to zero
0078     w(:,j)          = vbls_result.b_mean.*(vbls_result.i_scale.' / vbls_result.o_scale)
0079     w(IrInp,j)      = 0;
0080     vbls_nmse(:,j)     = mean((Ytest - Xtest*w(:,j)).^2) / var(Ytest);
0081
0082     % Storage
0083     vbls_results{j} = vbls_result;
0084
0085 end;
0086
0087 % Compute the classification error
0088 errorTtest       = sum(abs(TestVector - IrDim))/(Runs*M);
0089
0090 % Save the simulation results
```

```
0091 str1 = sprintf('save experiment3_{%d-%d-%f-%d-%d}.mat', N, M, SNRY, Mir, Mred);
0092 eval(str1);
0093
0094 % Plot the results
0095 figure(1);
0096 bar(IrDim);
0097 str2 = sprintf(['Detection of Irrelevant and Redundant features based on
     T-test\nN =',num2str(N),' M=',num2str(M),
     ' Irrel=',num2str(Mir),' Redun=',num2str(Mred)]);
0098 title(str2);
0099 xlabel('Feature number'); ylabel('Irrelevance count of each feature');
0100
0101          end % SNRY
0102          end % N>=M
0103      end % N
0104 end % M
0105
0106 % Simulations finished
```

B.2 gen_data.m

```
0001 % Artificial regression data generation
0002
0003 % Create noisy training dataset with redundant and irrelevant inputs
0004 Xtrain                    = 10*randn(N,M);
0005 Xtrain(:,M-Mred+1:end)     = 0.01*randn(N,Mred);
0006
0007 % Add zero coefficients to create irrelevant dimensions
0008 btemp                     = 10*randn(M-Mir,1);
0009
0010 % Make b values near zero larger
0011 bsgn                      = sign(btemp);
0012 bclip                     = max(abs(btemp), 3);
0013 brel                      = bsgn .* bclip;
0014 Btrue                     = [zeros(Mir,1); brel];
0015
0016 % Generate noisy output from X and Btrue
0017 Ytrain                    = Xtrain*Btrue;
0018 stdY                      = std(Ytrain)*sqrt(1/SNRY);
0019 Ytrain                    = Ytrain + randn(N,1)*stdY;
0020
0021
0022 % Create noiseless test data with the same properties as train data
0023 Xtest                     = 10*randn(N,M);
0024 Xtest(:,M-Mred+1:end)     = 0.01*randn(N,Mred);
0025 Ytest                     = Xtest*Btrue;
```

B.3 vbls.m

```
0001 % Variational Bayesian Least Squares (VBLS) algorithm
0002
0003
```

```
0004 function [result] = vbls(x, y, options)
0005
0006 % Size input matrix
0007 [N,d] = size(x);
0008
0009 % Normalize input matrix (mean=0 & variance=1)
0010 x = x - repmat(mean(x,1), N, 1);
0011 y = y - mean(y);
0012
0013 sX=std(x);
0014
0015 for dd=1:d,
0016  input_scaling(dd) = 1/sX(dd);
0017  x(:,dd) = (x(:,dd)) * input_scaling(dd);
0018 end
0019 output_scaling = 1/std(y);
0020 y=y*output_scaling;
0021
0022 % Initialization of parameters
0023 a_alpha_init = 1e-8;
0024 b_alpha_init = 1e-8*ones(d,1);
0025 a_alpha       = a_alpha_init;
0026 b_alpha       = b_alpha_init;
0027 alpha_mean    = a_alpha ./ b_alpha;
0028 b_mean        = zeros(d,1);
0029 b_var         = ones(d,1)./alpha_mean;
0030 psi_y         = options.noise;
0031 psi_z         = options.noise*ones(d,1);
0032 loglik         = -1.e10;
0033 LL             = [];
0034
0035
0036 % EM Algorithm
0037 for numIter = 1:options.numIterations
0038
0039    if mod(numIter, 10000) == 1
0040        numIter
0041    end
0042
0043    % E-step
0044
0045    % Compute mean and variance of z
0046    s           = psi_y + sum(psi_z./alpha_mean);
0047    z_var       = psi_z./alpha_mean - (psi_z./alpha_mean).^2/s;
0048    z_mean      = x*diag(b_mean) + 1/s*(y - x*b_mean)*(psi_z./alpha_mean)';
0049    z2_mean     = z_mean.^2 + repmat(z_var',N,1);
0050
0051    % Temporary variables
0052    sum_x2_psiz = sum(x.^2,1)' + psi_z;
0053    sum_zx      = sum(z_mean.*x,1)';
0054
0055    % Compute mean and variance of b
0056    b_var       = (psi_z./alpha_mean)./sum_x2_psiz;
0057    b_mean      = sum_zx./sum_x2_psiz;
```

```
0058
0059    % Compute precision variables a_alpha, b_alpha and mean value of alpha
0060    a_alpha     = a_alpha_init + N/2;
0061    b_alpha     = b_alpha_init ...
0062                  + 0.5*(sum(z2_mean,1)' - (sum_zx.^2)./sum_x2_psiz )./psi_z;
0063    alpha_mean  = a_alpha ./ b_alpha;
0064
0065    % M-step
0066
0067    z_mean_1  = sum(psi_z./alpha_mean)/s*y ...
0068                + (1 - 1/s*sum(psi_z./alpha_mean))*x*b_mean;
0069    psi_y     = sum((y - z_mean_1).^2)/N  + sum(z_var);
0070    psi_z     = sum( ((z_mean - x*diag(b_mean)).^2) ...
0071                .* repmat(alpha_mean',N,1),1)'/N ...
0072                + alpha_mean.*z_var + psi_z./sum_x2_psiz .* sum(x.^2,1)'/N;
0073
0074    % Monitor the lower bound of complete log likelihood every 10 iterations
0075    if mod(numIter, 10) == 1
0076       oldloglik = loglik;
0077       loglik = - N/2*log(psi_y) - 0.5/psi_y*sum((y - z_mean_1).^2) ...
0078          - N/2*sum(log(psi_z)) ...
0079          - 0.5*sum(sum((z_mean -x*diag(b_mean)).^2,1)'.*alpha_mean./psi_z) ...
0080          - 0.5*sum(alpha_mean.*(b_mean.^2 + b_var)) ...
0081          - (N-1)/2*sum(log(b_alpha)) - sum(a_alpha) ...
0082          + 0.5*sum(log(z_var)) -sum(log(b_alpha)) ...
0083          + 0.5*sum(alpha_mean.*(b_var+1));
0084    end
0085
0086    % Check to see if the likelihood tolerance has been reached
0087    if (abs(loglik - oldloglik) < options.threshold | ~finite(loglik))
0088      disp(sprintf('Likelihood ratio tolerance reached: (%g)-(%g) < %g',...
0089           loglik, oldloglik, options.threshold))
0090      break;
0091    end
0092
0093    LL(numIter,:) = loglik;
0094 end
0095
0096 % Results
0097 result.alpha_mean = alpha_mean;
0098 result.a_alpha    = a_alpha;
0099 result.b_alpha    = b_alpha;
0100 result.b_mean     = b_mean;
0101 result.b_var      = b_var;
0102 result.z_var      = z_var;
0103 result.z_mean     = z_mean;
0104 result.psi_y      = psi_y;
0105 result.psi_z      = psi_z;
0106 result.numIter    = numIter;
0107 result.LL         = LL;
0108 result.i_scale    = input_scaling;
0109 result.o_scale    = output_scaling;
```