# BAYESIAN MODELS AND MACHINE LEARNING WITH GENE EXPRESSION ANALYSIS APPLICATIONS

by

Ming Liao

Institute of Statistics and Decision Sciences
Duke University

Date: _____

Approved:

_____
Dr. Mike West, Supervisor

_____
Dr. Ed Iversen

_____
Dr. Feng Liang

_____
Dr. Sayan Mukherjee

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Institute of Statistics and Decision Sciences
in the Graduate School of
Duke University

2005

# ABSTRACT

(Statistics)

# BAYESIAN MODELS AND MACHINE LEARNING WITH GENE EXPRESSION ANALYSIS APPLICATIONS

by

Ming Liao

Institute of Statistics and Decision Sciences
Duke University

Date: _____

Approved:

_____
Dr. Mike West, Supervisor

_____
Dr. Ed Iversen

_____
Dr. Feng Liang

_____
Dr. Sayan Mukherjee

An abstract of a dissertation submitted in partial
fulfillment of the requirements for the degree
of Doctor of Philosophy in the
Institute of Statistics and Decision Sciences in the Graduate School of
Duke University

2005

# Abstract

The present thesis is divided into two major parts. The first part focuses on developing model-based estimates for gene expression indices in the Bayesian framework. In the application of oligonucleotide expression array technology, reliable estimation of expression indices is critical for "high-level analysis" such as classification, clustering and regulatory network exploration. A statistical model (Li and Wong, 2001a) has been proposed to develop model-based estimates for gene expression indices and outlier detection. Chapter 1 illustrates an extension of the model in the Bayesian framework. Proper constraints on model parameters, heavy-tail distributions for noise, and mixture priors are introduced with the help of Gibbs sampling. Our model is applied to both artificial probe data and real microarray probe data, with a demonstration that it is more robust and reliable than the original model.

The second part of the thesis concerns a novel Bayesian models for the problem of nonlinear regression for prediction. Recently, kernel methods have been introduced and become an increasingly popular tool for various regression, classification and function estimation problems. They exhibit good generalization performance on many real-life problems and the approach is properly motivated theoretically. After a brief introduction to kernel models and methods in Chapter 2, a new class of Bayesian kernel models is proposed in Chapter 3. First, we derive a novel Bayesian version of radial basis functions (RBFs) by utilizing the Dirichlet process prior on the distribution of location variables. This results in Bayesian kernel models as a special case. To achieve a sparse solution similar to SVMs, we introduce two classes of structured priors for regression parameters: mixture priors with point masses and Student-t priors. Orthogonalized kernel models are introduced to achieve better model mixing and speedup in the computation for problems with large sample sizes $n$. The problem

of inference on kernel parameters is addressed and a new discrete updating algorithm is proposed. For all the models introduced in this section, we develop both MCMC algorithms for fully Bayesian inference and EM algorithms for MAP estimation. Experimental results on some benchmark data sets show that the performance of our Bayesian kernel models is among the best of current state-of-art nonlinear models. Chapter 4 concludes the thesis by summarizing future developments.

# Acknowledgements

Several people have contributed advice and guidance as I have developed the research for this Ph.D. thesis. First of all, I wish to express my appreciation and gratitude to my advisor Professor Mike West for his advice and support. The present work would not have been possible without him. It has been a privilege to learn about statistics and science by working with him. He has given me the necessary independence to grow as a researcher, many insightful ideas, and a great example of enthusiasm for science.

I sincerely thank Professors Ed Iversen, Feng Liang and Sayan Mukherjee for many insightful comments and suggestions that improved this work. In particular, I would like to thank Professors Ed Iversen and Feng Liang for the invaluable help during the whole period of my Ph.D. study; and I would like to thank Professor Sayan Mukherjee for the discussions and help on the kernel methods. I also want to thank Professors Jennifer Pittman and Adrian Dobra for discussions on microarray data analysis.

Besides the dissertation work, I would like to thank Professors Mike West, Alan Gelfand, Dalene Stangl and Ms. Krista Moyle for their understanding and support, especially in the past one year. I also acknowledge the indispensable help I received during my years at ISDS from Pat Johnson and Eric Van Gyzen.

Last but not least I thank my wife Yue and my parents Jufang and Zhenghua for their endless and unconditional support.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Bayesian estimation of gene expression indices

## 1.1 Introduction

With the complete DNA sequences of many genomes already known and the recent release of the complete draft of the human genome (Insua and Muller, 1998; Consortium, 2001; Venter and *et al.*, 2001), it is important to define these genes, discover gene functions and the regulatory mechanisms of gene activations. Microarray technology is a groundbreaking new technology that provides information regarding gene expression on a genome wide scale. By allowing the simultaneous monitoring of the expression level of thousands of genes, microarrays have become ubiquitous tools for the molecular geneticist.

Microarray technology has already seen diverse applications, from understanding gene regulation and interactions (Brown *et al.*, 2000) to pharmaceutical and clinical research. The availability of this technology also poses challenging questions regarding data analysis and experimental design. Microarray experiments produce very large data sets capturing complex biological processes. A systematic and rigorous approach to data analysis is therefore critical for interpreting the information

provided by microarray experiments. This Chapter will focus on the fundamental problems of developing robust statistical estimation of gene expression. We start with a brief introduction to microarray technology and experiments. After an introduction to a model-based analysis of gene expression indexes for oligonucleotide arrays, we propose a Bayesian model to estimate gene expression indexes robustly. Experimental results on both artificial data and real gene data are provided.

## 1.2 Microarray technology

### 1.2.1 Introduction

Microarray technology is based on the fundamental property of hybridization. Hybridization is the process of base pairing two single strands of nucleic acids (a sequence of which is immobilized on a substrate). It provides high sensitivity and specificity of detection as a consequence of exquisite, mutual selectivity between complementary strands of nucleic acids (Southern and Shchepinov, 1999). A microarray consists of a dense patterning of thousands of cDNA strands (cDNA microarrays) or oligonucleotide sequences (synthetic oligonucleotide microarrays) immobilized on a substrate (usually a glass slide or a nylon membrane). Gene expression experiments are performed after isolating mRNA from cells or tissue of interest, and then reverse-transcribing this mRNA to cDNA, which is tagged either with a fluorophore or radioactive P33. The labeled cDNA is then washed over the surface of the microarray, and allowed to incubate for a period of time. The immobilized sequences in the microarray are referred to as "probes", while the cDNA or cRNA representation of cellular mRNA extracted from the cell are referred to as "target". During this incubation phase, hybridization of the test cDNA occurs. For convenience, we say that each probe corresponds to a "gene", irrespective of whether it actually represents a gene of full length, an expressed sequence tag (EST), or DNA from other

sources. Scanning of the microarray, using either confocal microscopy or phosphor imaging techniques, yields quantifiable digital images of the array hybridization experiments. Either fluorescence intensity or the extent of radiolabeling at each spot is proportional to the amount of target hybridized to each probe. Since the concentration of the probe is large relative to that of the target, hybridization occurs at a rate which is proportional to the concentration of the target and to the incubation time (Duggan, 1999). Specific digital image processing procedures take advantage of the highly regular arrangement of the gene spots on the array to extract the intensity value of each spot (Cheung *et al.*, 1999; Eisen, 1999; Carlisle *et al.*, 2000; Chen *et al.*, 1997).

In such experiments, florescence levels measured from substrate regions surrounding the probe spots are non-zero. This so-called "background signal" results from a variety of non-specific chemical and physico-chemical reactions between the labeled biological target sample and the microarray substrate, and represents a form of optical noise. Thus, the measured gene intensity is due to both specific binding of the gene target to the probe and background labeling. Therefore, appropriate analysis of background intensity level is important for proper analysis of microarray data. In experiments using fluorescent dye, local sampling of background intensity is possible around each DNA probe spot, since the signal is sharply delimited (Chen *et al.*, 1997).

In radioactive hybridizations, the smoothness of the transition from maximum signal intensity to background signal intensity makes estimation of local background difficult. However, the design of arrays for radioactive hybridization often includes empty spots (i.e. containing no DNA probe) for estimation of background intensity (Carlisle *et al.*, 2000).

Two classes of microarrays have emerged from two different technological ap-

proaches. High density oligonucleotide arrays (Wodicka *et al.*, 1997; Southern, 1998; Watson *et al.*, 1998; Lipshutz *et al.*, 1999; Hughes *et al.*, 2001) are designed and synthesized based on sequence information alone. cDNA arrays (Shalon and Brown, 1996; Schena *et al.*, 1996; Watson *et al.*, 1998; Duggan, 1999) exploit cDNA libraries and amplification technique (Polymerase Chain Reaction (PCR)). We will now describe the main features of these two approaches.

### 1.2.2 cDNA microarray

cDNA microarray technology was developed at Stanford university and introduced in Schena *et al.* (1995). Production of cDNA microarrays begins with the selection of the probes to be printed on the microarray. In many cases, these are chosen directly from databases including GenBank, dbEST, and UniGene. The corresponding cDNA clones are amplified by a technique, polymerase chain reaction (PCR). cDNA microarrays are then produced by spotting PCR products (of approximately 0.6-2.4 kb) representing specific genes on a glass slide using high-speed robot. Each microarray element is generated by the deposition of a few nanoliters of purified PCR product, typically 100-500 $\mu$g/ml. Printing is carried out by a robot that spots each gene product onto a number of substrates in a serial operation. The set of microarrays produced in the same robot operation define a "lot" of highly similar microarrays. The microarrays are generally divided into grids, all the spots on the same grid are printed using the same print-tip or pin.

Figure 1.1 shows an overview of procedures for preparing cDNA microarrays. First, researchers extract total RNA or mRNA produced from two types of cells, for example, test and control cells. Then, by using a single round of reverse transcription, the mRNA from the two samples is fluorescently labeled with Cy3 (green) and Cy5 (red), and the target mixture is hybridized to the probes on the glass slides. During

the hybridization, if segments of the mRNA representation in the target find their complementary portion among the samples of cDNA on the glass slide, they will bind together. When the hybridization is complete, the glass slide is washed and laser excitement of the glass slide is used to yield a luminous emission that is then measured by a scanning microscope. Fluorescence measurements are made with a microscope that illuminates each spot and measures fluorescence for each dye separately, thus providing a measure of the relative mRNA abundance for each gene in the two cells. The intensity of the green spot measures the relative mRNA abundance of the gene in the cell that had reverse-transcribed mRNA labeled with Cy3, while the intensity of the red spot measures the relative mRNA abundance of the gene in the cell that had reverse-transcribed mRNA labeled with Cy5. Grey spots denote genes that were expressed in neither cell type. These measurements provide information about the relative level of expression of each gene in the two cells. The monochrome images can be pseudocolored to provide a quantitative measure of the relative expression of each gene in the two cells. This measure is adjusted to account for background noise caused by high salt and detergent concentrations during the hybridization or contamination of the target. Figure 1.2 shows one of these images in which spots are colored in red, green, yellow and grey. Each spot corresponds to a gene, and the color of the spot discloses whether the gene is expressed (colored) or not and the relative level of expression in the two targets. Usually a measurement scale is provided to associate each color tone with a ratio between expression level in the two cells (Schena *et al.*, 1995; Brown and Botstein, 1999).

### 1.2.3 Oligonucleotide arrays

In oligonucleotide microarrays, gene probes are short sequences of nucleotides (typically between 20 and 60 nucleotides) specific to a particular gene (Lipshutz *et al.*,

1999). Oligonucleotides can be synthesized in situ or can be pre-synthesized and then deposited on to the substrate. In the in situ synthesis methods, oligonucleotides are built directly on the substrate. Affymetrix (Santa Clara, CA) generates arrays using a photolithographic approach where a mercury lamp is used to shine light through a photolithographic mask on to the surface of the substrate, selectively removing photo-labile deprotecting groups from the oligonucleotide chain in a stepwise fashion to create oligonucleotides. The length of oligonucleotides synthesized by this technique is currently limited to about 25 bases. For gene expression, this can reduce sensitivity and specificity (Lipshutz *et al.*, 1999). This problem is addressed in a microarray design including redundancy and mismatch control probes. Redundancy is achieved through spotting of multiple oligonucleotides of different sequences designed to hybridize to different regions of the same gene.

On the GeneChip platform, each gene is represented by a number of probe pairs ranging from 11 in the new Human Genome U133 set to 16 in the Murine Genome U74v2 set and the Human Genome U95v2. A probe pair consists of a perfect match (PM) probe and a mismatch (MM) probe. Each PM probe is chosen on the basis of uniqueness criteria and proprietary, empirical rules designed to improve the odds that probes will hybridize with high specificity. MM probes are spotted next to each perfect match gene probe. They are identical to their perfect match partner except for a single base difference in central position. They are used as a specificity control, to assess the specificity of the hybridization (Wodicka *et al.*, 1997; Lipshutz *et al.*, 1999). Figure 1.3 shows a illustrative example of PM/MM structure of GeneChip arrays. Each cell of an Affymetrix oligonucleotide microarray consists of millions of samples of a PM or MM probe, and probes that tag the same gene are scattered across the microarray to avoid systematic bias.

To prepare the target, investigators extract total RNA from a cell or tissue. The

mRNA is reverse-transcribed into cDNA, which is made double-stranded and then converted into cRNA using a transcription reaction that fluorescently labels the target. Once hybridization has occurred, the microarray is washed and scanned with a standard laser scanner. The scanner generates an image of the microarray that is gridded to identify the cells that contain each probe and analyzed to extract the signal intensity of each probe cell. Figure 1.4 shows an illustrative example of probe cell, pixel and GeneChip Probe Array. Each probe cell or feature contains millions of copies of a specific oligonucleotide probe. There are over 250,000 different probes complementary to genetic information of interest in a single array.

Oligonucleotide sequences can also be synthesized in situ by a process similar to color ink jet printing. Phosphoramidite (a base synthesis reagent) is delivered robotically to specified locations on the array in successive rounds of synthesis, thereby creating the desired sequence at each array element (Hughes *et al.*, 2001).

## 1.3 Model-based analysis of gene expression indexes for oligonucleotide arrays

In Oligonucleotide microarrays, 16 to 20 probe pairs are used to interrogate each gene, and each probe pair has a Perfect Match (PM) and Mismatch (MM) signal. It is important to find a reliable way to combine the 16 to 20 probe pairs' intensities for a given gene to define a measure of expression index, since the expression index is the starting point for high-level analyses, such as clustering and classification.

We denote the probe intensities for a given gene as

$$PM_{ij} \text{ and } MM_{ij},\ i = 1, ..., n, j = 1, ..., m$$

with $i$ representing the different arrays and $j$ representing the probe pair number. The number of arrays $n$ ranges form 1 to hundreds, and the number of probe pairs

$m$ ranges from 16 to 20 generally.

The original measure of expression index is the average differences of $PM$ and $MM$, as provided by Affymetrix (Lockhart *et al.*, 1996; Wodicka *et al.*, 1997). For each probe set on each array $i$, the average differences $AD_i$ is defined by

$$AD_i = \frac{1}{\#A}\sum_{j\in A}(PM_{ij} - MM_{ij}),$$

with $A$ the subset of probes after deleting those extreme measurements which exceed three standard deviations from the mean. $\#A$ is the number of probe pairs in $A$. There are variants of average differences measures with different ways of removing outliers and different ways of dealing with small values and one of these variants underlies the current MAS 5.0 method commonly used.

Note that another commonly used method, robust multi-array average (RMA) (Irizarry *et al.*, 2003a; Irizarry *et al.*, 2003b), is not a variant of average differences. Instead, RMA is based on a statistical model of background adjusted and log transformed PM values. Thus RMA can be considered as a variant of model-based expression index which will be discussed in the following. In the final Chapter of this thesis, a Bayesian version of RMA is proposed.

Average difference assumes that all probe pairs used have the same effect on the expression index. In reality, there are dramatic variations among $PM - MM$ of different probes in the same array (Li and Wong, 2001a). Analyses of many real probe data sets show that the variation due to probe effects is larger than the variation due to arrays. It is also observed that the probe-specific effects are highly reproducible and predictable as shown in Figure 1.5 (Derived from Li and Wong (2001a)). This suggests that the probe-specific effects is an essential component of estimation of expression index.

Li and Wong (2001a) proposed a model-based expression index. Let $\theta_i$ denote

the expression index for that gene in the $ith$ sample. Li and Wong's (LW) model is defined by

$$\begin{aligned} y_{ij} &= PM_{ij} - MM_{ij} \\ &= \phi_j \theta_i + \varepsilon_{ij}, \end{aligned}$$

where $\phi_j$ is the probe-specific affinities and $\varepsilon_{ij}$s are assumed to be i.i.d. normally distributed errors $\varepsilon_{ij} \sim N(\varepsilon_{ij}|0, \sigma^2)$. The model parameters $\theta_i$, $\phi_j$ and $\sigma^2$ are estimated via MLE, which includes the interactively explicit outlier removal.

The LW model can be considered as a one-component Probabilistic Principal Component Analysis (PPCA) (Tipping and Bishop, 1997) model with spherical noise covariance matrix. Let $\mathbf{y}_i = [y_{i1}, ..., y_{im}]^T$, $\phi = [\phi_1, ..., \phi_m]^T$ and $\varepsilon_i = [\varepsilon_{i1}, ..., \varepsilon_{im}]^T$; then the model can be rewritten as

$$\begin{aligned} \mathbf{y}_i &= \phi \theta_i + \varepsilon_i, \\ \varepsilon_i &\sim N(\varepsilon_i|0, \Psi), \\ \Psi &= \sigma^2 I_{m \times m}. \end{aligned}$$

Though the LW model has been successfully applied in a wide range of applications, there are several limitations:

- The rationale behind the use of difference of PM and MM is that the specific hybridization, represented by the intensity of PM probes, should be stronger than the non-specific hybridization, represented by the intensity of MM probes. Ideally, all $y_{ij}$s are supposed to be greater than zero. However, values of MM can be higher than PM values for various reasons, such as cross-hybridization occurring when the probe sequence has high homology with another unknown sequence. Due to the fact that there are many of negative values of $y_{ij}$ for real world array data, some values of $\phi$ and $\theta$ estimated by LW model are negative. In biological applications, both $\phi$ and $\theta$ are expected to be positive.

It is necessary to impose these constraints on the procedure of model fitting and find an appropriate way to deal with negative $y_{ij}$s.

- In general, the number of arrays, $n$, ranges form 1 to hundreds. When only a small number of arrays are available, the sample sizes could be small compared to the number of parameters to be estimated. In that case, the gene expression index estimated by MLE could be unreliable.

- LW model assumes that the variances of error for different probe pairs are the same. This assumption could be wrong due to the fact that there are dramatic variations among $y_{ij}$s for different probe pairs. It is more realistic to allow probe-specific error variance $\sigma_j^2$ in the model, i.e.,

$$y_{ij} = \phi_j \theta_i + \varepsilon_{ij},$$
$$\varepsilon_{ij} \sim N(\varepsilon_{ij}|0, \sigma_j^2),$$

  which is equivalent to the factor analysis with diagonal covariance matrix:

$$\mathbf{y}_i = \phi \theta_i + \varepsilon_i,$$
$$\varepsilon_i \sim N(\varepsilon_i|0, \Psi),$$
$$\Psi = diag(\sigma_1^2, ..., \sigma_m^2).$$

- The real world array data could be very noisy due to the cross-hybridization, image contamination etc. Though LW method remove outliers iteratively (Li and Wong, 2001a; Li and Wong, 2001b) in the process of model fitting, it is desirable to generate a model to deal with outliers automatically.

In the following, we will introduce a Bayesian model to address the above problems.

## 1.4 Bayesian analysis of gene expression indices

### 1.4.1 Likelihood and priors

To deal with the outliers, we introduce the following model

$$y_{ij} = \phi_j \theta_i + \varepsilon_{ij}, \tag{1.1}$$
$$\varepsilon_{ij} \sim N(\varepsilon_{ij}|0, \sigma_j^2/\lambda_{ij}),$$

where $\lambda_{ij}$ is the individual scale parameter for each data. A small value of $\lambda_{ij}$ corresponds to a noisy datum, which implies that the noisy data play minor roles in model fitting. The priors for $\lambda_{ij}$ is an independent inverse gamma

$$\lambda_{ij} \sim Ga(\lambda_{ij}|\frac{k}{2}, \frac{k}{2}),$$

with shape parameter $k/2$. This prior on $\lambda_{ij}$ means that the implied priors for error term $\varepsilon_{ij}$, on marginalization with respect to the $\lambda_{ij}$, are the independent $T$ priors with $k$ degrees of freedom. To introduce the robustness, $k$ is generally specified as a small integer, say $k = 2$ or 3 (West, 1984).

To impose constraints on $\theta_i$, $\phi_j$, we can simply specify the uniform priors in the range of positive values

$$\phi_j \sim U(\phi_j|0, \infty),$$
$$\theta_i \sim U(\theta_i|0, \infty).$$

To make the model identifiable, we need to impose constraints on $\phi_j$. Here we set the constraint as $\sum_j \phi_j = m$, so that the estimated $\theta_i$ can be comparable to average difference directly since

$$AD_i = (\sum_j y_{ij})/m = \theta_i.$$

The conjugate prior distribution for $\sigma^2$ is an inverse gamma distribution,

$$p(\sigma^{-2}) = Ga\left(\sigma^{-2} \left| \frac{n_0}{2}, \frac{n_0}{2}\sigma_0^2 \right.\right). \tag{1.2}$$

### 1.4.2 Conditional posteriors

Given the above likelihood and priors, we have the following full conditional posteriors:

- The conditional posterior for $\theta_i$ is a truncated normal posterior

$$p(\theta_i|\mathbf{y}_i, \phi, \lambda, \sigma^2) = N(\theta_i|\widehat{\theta}_i, v_i)I(\theta_i > 0),$$

  where

$$\widehat{\theta}_i = (\sum_j \frac{y_{ij}\phi_j\lambda_{ij}}{\sigma_j^2})v_i,$$

$$v_i = 1/(\sum_j \frac{\phi_j^2\lambda_{ij}}{\sigma_j^2}).$$

- The conditional posterior for $\phi_j$ is a truncated normal posterior,

$$p(\phi_j|\mathbf{y}, \theta, \lambda, \sigma^2) = N(\phi_j|\widehat{\phi}_j, s_i)I(\phi_j > 0),$$

  where

$$\widehat{\phi}_j = \frac{(\sum_i y_{ij}\theta_i\lambda_{i,j})}{\sigma_j^2}s_j,$$

$$s_i = \frac{\sigma_j^2}{(\sum_i \theta_i^2\lambda_{i,j})}.$$

- The conditional posterior for $\sigma_j^2$ is an inverse gamma posterior

$$p(\sigma_j^{-2}|\mathbf{y}, \theta, \phi, \lambda) = Ga(\sigma_j^{-2}|\frac{n_0 + n}{2}, \frac{n_0\sigma_0^2 + \sum_i (y_{ij} - \phi_j\theta_i)^2\lambda_{i,j}}{2}).$$

- The conditional posterior for $\lambda_{i,j}$ is a gamma posterior

$$p(\lambda_{i,j}|y_{ij}, \theta_i, \phi_j, \sigma_j^2) = Ga(\lambda_{i,j}|\frac{k+1}{2}, \frac{k\sigma_j^2 + (y_{ij} - \phi_j\theta_i)^2}{2\sigma_j^2}).$$

### 1.4.3 Model fitting via MCMC

Given the above conditional posteriors, it is straightforward to implement the posterior simulation using standard Gibbs sampling, illustrated in detailed below. The initial values are chosen arbitrarily for each of parameters, and then the samples of parameters are drawn from their conditional posterior distributions at each iteration. The components of each MCMC step for our model

$$y_{ij} = \phi_j \theta_i + \varepsilon_{ij},$$
$$\varepsilon_{ij} \sim N(0, \sigma_j^2/\lambda_{ij}),$$

are as follows:

- Given the current value of $\phi, \lambda, \sigma^2$, draw $\theta_i$ from the truncated normal posterior

$$p(\theta_i|\mathbf{y}_i, \phi, \lambda, \sigma^2) = N(\theta_i|\widehat{\theta}_i, v_i) I(\theta_i > 0).$$

- Given the current value of $\theta, \lambda, \sigma^2$, draw $\phi_j$ from the truncated normal posterior

$$p(\phi_j|\mathbf{y}, \theta, \lambda, \sigma^2) = N(\phi_j|\widehat{\phi}_j, s_i) I(\phi_j > 0).$$

- Given the current value of $\phi, \theta, \lambda$, draw $\sigma_j^2$ from the inverse gamma posterior

$$p(\sigma_j^{-2}|\mathbf{y}, \theta, \phi, \lambda) = Ga(\sigma_j^{-2}|\frac{n_0 + n}{2}, \frac{n_0\sigma_0^2 + \sum_i (y_{ij} - \phi_j\theta_i)^2\lambda_{i,j}}{2}).$$

- Given the current value of $\theta_i, \phi_j, \sigma_j^2$, draw $\lambda_{i,j}$ from the gamma posterior

$$p(\lambda_{i,j}|y_{ij}, \theta_i, \phi_j, \sigma_j^2) = Ga(\lambda_{i,j}|\frac{k+1}{2}, \frac{k\sigma_j^2 + (y_{ij} - \phi_j\theta_i)^2}{2\sigma_j^2}).$$

### 1.4.4 Mixture priors with point mass

In real world array data, most of $y_{ij}$s could be negative for a given gene in a given array, which implies that the gene is not expressed in that sample. In that case, we can expect the gene expression index $\theta_i$ should be close to zero. Under the prior specification for $\theta_i$ above, the positively truncated normal posterior does not concentrate around zero very closely, and the posterior mean could still have a large value. In the following, we will specify the mixture priors with point mass at zero for $\theta_i$ to solve this problem.

We assume that the prior for $\theta_i$ is an independent mixture of uniform distribution and a point mass at zero

$$\theta_i \sim (1 - z_i)\delta_0 + z_i U(\theta_i|0, \infty),$$

where $\delta_0$ is the point mass at zero and $z_j \in \{0, 1\}$ is a binary latent variable. In particular, $z_i = 0$ means that the $\theta_j$ would be so small that it could be estimated as zero. Otherwise $z_i = 1$ means that the $\theta_j$ would be estimated as a positive value as above.

We now calculate the conditional posterior distribution for $z_i$. Let

$$P1 = p(D|z_i = 1, \phi_j, \theta_i, \sigma_j^2, \lambda_{ij}) = \prod_j N(y_{ij}|\phi_j\theta_i, \frac{\sigma_j^2}{\lambda_{ij}}),$$

$$P2 = p(D|z_i = 0, \phi_j, \theta_i, \sigma_j^2, \lambda_{ij}) = \prod_j N(y_{ij}|0, \frac{\sigma_j^2}{\lambda_{ij}}),$$

and $\pi = p(z_i = 1)$. Given the uniform prior for $\pi$, the conditional posterior for $\pi$ is a beta posterior

$$p(\pi|\mathbf{z}) \propto \pi^{\sum_i z_i}(1 - \pi)^{n - \sum_i z_i},\ 0 < \pi < 1.$$

Let

$$\pi_i^* = \frac{\pi P_1}{\pi P_1 + (1 - \pi)P_2}.$$

Then the conditional posterior for $z_i$ is Bernoulli

$$z_i \sim Ber(z_i|\pi_i^*).$$

The above MCMC procedure can be modified as

- Given the current value of $\phi$, $\lambda$, $\sigma^2$ and $\theta_i$, draw $z_i$ from the Bernoulli posterior

$$z_i \sim Ber(z_i|\pi_i^*)$$

  - For $z_i = 1$, sample $\theta_i$ from the truncated normal posterior

$$p(\theta_i|\mathbf{y}_i, \phi, \lambda, \sigma^2) = N(\theta_i|\widehat{\theta}_i, v_i)I(\theta_i > 0).$$

  - Otherwise set $\theta_i = 0$

- Sample $\phi_j, \sigma_j^2, \lambda_{ij}$ in the same way as before.

## 1.5 Experimental results

In this section, we apply our Bayesian model to three data sets. The first data set is an artificial array data set with known expression index $\theta_i$, probe-specific affinities $\phi_j$ and error variance $\sigma_j^2$. It is used to explore if the Bayesian model can give the correct estimation of expression index. The second data set is almost the same as data set 1 except that several outliers are included. It is used to show how the Bayesian model accommodate outliers. The third data set is the real array data set from Duke breast cancer studies. Besides the results of our Bayesian model, the average difference and the results of LW model are also provided for comparisons.

### 1.5.1 Artificial array data set 1

This data set is an artificial array data set with known expression index $\theta_i$, probe-specific affinities $\phi_j$ and error variance $\sigma_j^2$. Suppose there are 11 samples of one given

gene. The gene expression indices in 11 samples are $\theta$ =[100, 125, 150, 175, 200, 225, 250, 275, 300, 0, −100]$^T$. The $10th$ sample is not expressed and the $11th$ sample is a "bad" array because $\theta_{11}$ is negative. Suppose there are 6 probe pairs for the given gene. The probe-specific affinities are $\phi$ =[1, 1, 1.5, 2.5, 0, 0]$^T$ and the error variances are $\sigma^2 = [1, 9, 1, 1, 1, 1]^T$. Thus the $5th$ and $6th$ probe pairs are "bad" probe pairs that are supposed to play small roles in estimating the gene expression index, and the $2rd$ probe pair has more noise than other probe pairs.

The artificial array data $y_{ij}$s are generated by the following model

$$y_{ij} = \phi_j \theta_i + \varepsilon_{ij},$$
$$\varepsilon_{ij} \sim N(\varepsilon_{ij}|0, \sigma_j^2).$$

The bar plots of $y_{ij}$s in 4 arrays (1,5,8,11) are plotted in the first 4 rows of Figure 1.6.

We fit our Bayesian model via MCMC. The MCMC was run for 5,000 iteration after a burn-in of 1,000. The posterior means of $\phi$ and $\sigma^2$ are shown in the last two rows of Figure 1.6. Figure 1.7 shows the fitted value $\widehat{y}_{ij} = \overline{\phi}_j \overline{\theta}_i$ for 4 arrays (1,5,8,11), where the $\overline{\phi}_j$ and $\overline{\theta}_i$ are posterior means. The $x$ axis represents the probe pairs, and the $y$ axis represents the value $\widehat{y}_{ij}$s. The blue line shows the original data $y_{ij}$s, while the red line shows the fitted value $\widehat{y}_{ij}$s. Figure 1.8 shows the similar plot for the results of LW model. As we can see, both LW model and our Bayesian model can fit the data with positive value (array 1,5,8) very well. While LW mode also tries to fit the data with negative value (array 11), our Bayesian model gives the fitted value close to zero since the negative $y_{ij}$s are inferred to be outliers. The first 10 rows of Table 1.1 report the gene expression index estimated by average difference, LW model and our Bayesian model. Note that the $\phi$ in LW model is scaled under the constraint $\sum \phi_j = m$, while the original constraint is $\sum \phi_j^2 = m$, thus the scaled $\theta$ is comparable to average differences and the Bayesian model. It shows that both

LW model and Bayesian model give estimates that are close to the true values of first 9 samples. While LW model gives negative expression values for the $10th$ sample and $11th$ sample, the Bayesian model provides positive values that are very close to zero. The last 6 rows of Table 1.1 report the probe-specific affinities estimated by LW model and the Bayesian model, which shows that both models give estimates that are close to the true values.

### 1.5.2 Artificial array data set 2

This data set is an artificial array data set that is generated in the same way as the artificial array data set 1, except for more intentionally added outliers. The signs of two data values in each of three previously "good" arrays 1,5,8 are changed to introduce the outliers. Table 1.2 shows the data $y_{ij}$s in which the outliers are highlighted, and the bar plot of $y_{ij}$s in 4 arrays (1,5,8,11) are plotted in the first 4 rows of Figure 1.9.

We fit our Bayesian model via MCMC. The MCMC was run for 5,000 iteration after a burn-in of 1,000. The posterior means of $\lambda_{ij}$s are listed in Table 1.3, in which the value corresponding to the outliers are highlighted. We can clearly see that the $\lambda_{ij}$s corresponding to the outliers are extremely small, while the $\lambda_{ij}$s corresponding to the ordinary data are close to 1. This indicates that our Bayesian model can identify those outliers automatically, and those outliers have little effect in estimating the gene expression index.

The posterior mean of $\phi$ and $\sigma^2$ are shown in the last two rows of Figure 1.9. Figure 1.10 shows the fitted value $\widehat{y}_{ij} = \overline{\phi}_j \overline{\theta}_i$ for 4 arrays (1,5,8,11), where the $\overline{\phi}_j$ and $\overline{\theta}_i$ are posterior mean. The $x$ axis represents the probe pairs, and the $y$ axis represents the value $\widehat{y}_{ij}$s. The Blue line shows the original data $y_{ij}$s, while the red line shows the fitted value $\widehat{y}_{ij}$s. Figure 1.11 shows a similar plot for the results of LW model.

As we can see, our Bayesian model can fit the non-outliers very well and recover the value of outliers, while the LW model fails to fit the data. The first 10 rows of Table 1.4 report the gene expression index estimated by average difference, LW model and our Bayesian model. The samples (1,5,8) with the outliers are highlighted. This shows that both average difference and LW model fail to give proper expression index estimation, even in the samples (2,3,4,6,7,9) in which there are not outliers, because they are seriously affected by outliers. Instead, the Bayesian model gives the estimates that are close to the true value for all 11 samples with or without outliers. The last 6 rows of Table 1.4 report the probe-specific affinities estimated by LW model and the Bayesian model. This shows that estimates of LW model are seriously affected by outliers, while the Bayesian model is robust to the outliers.

### 1.5.3 ER gene

This data set is from the probe data of ER gene in Duke's breast cancer study. The number of samples is 49 and the number of probe pairs is 20. The bar plots of $y_{ij}$s in 4 random arrays (2, 15, 17, 18) are plotted in the first 4 rows of Figure 1.12.

We fit our Bayesian model via MCMC. The MCMC was run for 5,000 iteration after a burn-in of 1,000. The posterior means of $\phi$ and $\sigma^2$ are shown in the last two rows of Figure 1.12. The Figure 1.13 shows the fitted value $\widehat{y}_{ij} = \overline{\phi}_j\overline{\theta}_i$ for 4 arrays , where the $\overline{\phi}_j$ and $\overline{\theta}_i$ are posterior means. The $x$ axis represents the probe pairs, and the $y$ axis represents the value $\widehat{y}_{ij}$s. The Blue line shows the original data $y_{ij}$s, while the red line shows the fitted value $\widehat{y}_{ij}$s. Figure 1.14 shows a similar plot for the results of LW model. As we can see, both LW mode and our Bayesian model give similar results for the data with positive value (array 2, 15). While LW mode also try to fit the data with negative value (array 17, 18), our Bayesian model gives the fitted value close to zero since the negative $y_{ij}$s are suppose to be outliers. Since the

true gene expression index is unknown, the comparison of estimated gene expression index is not provided.

| Parameter | True value | AD | LW | Bayesian |
|---|---|---|---|---|
| $\theta_1$ | 100 | 88.6245 | 100.2981 | 100.63 |
| $\theta_2$ | 125 | 112.4154 | 126.1608 | 126.27 |
| $\theta_3$ | 150 | 130.3252 | 149.3395 | 149.22 |
| $\theta_4$ | 175 | 154.2671 | 175.6591 | 176.23 |
| $\theta_5$ | 200 | 177.9248 | 200.9446 | 200.96 |
| $\theta_6$ | 225 | 196.9750 | 225.0479 | 226.4 |
| $\theta_7$ | 250 | 220.2333 | 250.7780 | 251.05 |
| $\theta_8$ | 275 | 236.7399 | 273.6411 | 275.93 |
| $\theta_9$ | 300 | 265.6442 | 301.2850 | 300.67 |
| $\theta_{10}$ | 0 | -0.4543 | -1.5188 | 0.0097826 |
| $\theta_{11}$ | -100 | -86.9064 | -99.6128 | 0.62201 |
| $\phi_1$ | 1 | | 0.9989 | 0.9968 |
| $\phi_2$ | 1 | | 1.0067 | 1.0162 |
| $\phi_3$ | 1.5 | | 1.4987 | 1.4954 |
| $\phi_4$ | 2.5 | | 2.4960 | 2.4885 |
| $\phi_5$ | 0 | | -0.0016 | 0.0011885 |
| $\phi_6$ | 0 | | 0.0012 | 0.0019383 |

**Table 1.1**: Comparison of AD, LW model and the Bayesian model for artificial array data set 1: First 10 rows reports the gene expression index estimated by average difference, LW model and our Bayesian model. Both LW model and Bayesian model give estimates that are close to the true value for first 9 samples. While LW model gives negative expression values for the $10th$ sample and $11th$ sample, the Bayesian model provides the positive values that are very close to zero. The last 6 rows report the probe-specific affinities, which shows that both models give estimates that are close to the true values.

| | PP 1 | PP 2 | PP 3 | PP 4 | PP 5 | PP 6 |
|---|---|---|---|---|---|---|
| Array 1 | 100.3 | **-103** | 150.7 | **-249** | 0.888 | 0.172 |
| Array 2 | 125.8 | 133.7 | 188.8 | 312.4 | 1.317 | 0.228 |
| Array 3 | 148.6 | 148.7 | 224.5 | 373.3 | -0.42 | -0.62 |
| Array 4 | 175.7 | 178.1 | 263.4 | 437.8 | -0.15 | -0.09 |
| Array 5 | **-200** | 210.5 | **-301** | 498.6 | 0.121 | -0.22 |
| Array 6 | 225.6 | 222.3 | 338.6 | 562.3 | -1.47 | 1.395 |
| Array 7 | 250.4 | 255.1 | 374.3 | 625.8 | -2.24 | 1.098 |
| Array 8 | 275 | **-260** | **-411** | 687.7 | 0.221 | -1.01 |
| Array 9 | 299.5 | 312.6 | 449.5 | 750 | 0.799 | 0.897 |
| Array 10 | 0.138 | -14.6 | -1.65 | 0.429 | -0.74 | 0.565 |
| Array 11 | -101 | -95.9 | -149 | -250 | -1.01 | -0.35 |

**Table 1.2**: Table of artificial array data set 2: the data set is generated in the same way as the artificial array data set 1. The signs of two data value in each of three previously "good" arrays 1,5,8 are changed to introduce the outliers. The rows correspond to the arrays, and the column correspond to the probe pairs (PP). The outliers are highlighted in **bold** font.

| | PP 1 | PP 2 | PP 3 | PP 4 | PP 5 | PP 6 |
|---|---|---|---|---|---|---|
| Array 1 | 1.135 | **0.007** | 1.371 | **0** | 0.717 | 1.45 |
| Array 2 | 0.898 | 1.219 | 1.268 | 0.734 | 0.473 | 1.354 |
| Array 3 | 0.676 | 1.259 | 1.095 | 0.936 | 0.936 | 0.439 |
| Array 4 | 0.985 | 1.372 | 1.363 | 0.97 | 1.206 | 0.973 |
| Array 5 | **0** | 1.127 | **0** | 0.996 | 1.424 | 0.715 |
| Array 6 | 0.995 | 0.958 | 1.206 | 0.94 | 0.27 | 0.397 |
| Array 7 | 1.122 | 1.371 | 0.848 | 1.151 | 0.141 | 0.675 |
| Array 8 | 1.107 | **0.001** | **0** | 1.115 | 1.372 | 0.202 |
| Array 9 | 0.926 | 1.143 | 1.247 | 0.944 | 1.037 | 0.988 |
| Array 10 | 1.339 | 0.607 | 0.603 | 1.338 | 0.768 | 0.715 |
| Array 11 | 0 | 0.028 | 2E-04 | 0 | 0.565 | 0.992 |

**Table 1.3**: Table of the posterior means of $\lambda_{ij}$s: the value corresponding to the outliers are highlighted in **bold** font. We can clearly see that the $\lambda_{ij}$s corresponding to the outliers are extremely small, while the $\lambda_{ij}$s corresponding to the ordinary data are close to 1.

| Parameter | True value | AD | LW | Bayesian |
|---|---|---|---|---|
| $\theta_1$ | 100 | **-0.3026** | **-48.8568** | **100.7963** |
| $\theta_2$ | 125 | 1112.4154 | 111.4512 | 126.1475 |
| $\theta_3$ | 150 | 130.3252 | 132.1931 | 149.9907 |
| $\theta_4$ | 175 | 154.2671 | 155.4170 | 176.0978 |
| $\theta_5$ | 200 | **2.5233** | **98.0822** | **200.3695** |
| $\theta_6$ | 225 | 196.9750 | 199.2218 | 226.2214 |
| $\theta_7$ | 250 | 220.2333 | 221.9818 | 251.5120 |
| $\theta_8$ | 275 | **3.4361** | **146.9864** | **276.3694** |
| $\theta_9$ | 300 | 265.6442 | 266.5048 | 301.4039 |
| $\theta_{10}$ | 0 | -0.4543 | -0.9590 | 0.0213 |
| $\theta_{11}$ | -100 | -86.9064 | -88.2880 | 1.3277 |
| $\phi_1$ | 1 | | 1.0886 | 1.0163 |
| $\phi_2$ | 1 | | 0.8271 | 1.0162 |
| $\phi_3$ | 1.5 | | 0.9439 | 1.4942 |
| $\phi_4$ | 2.5 | | 3.1409 | 2.4887 |
| $\phi_5$ | 0 | | -0.0022 | 0.00065386 |
| $\phi_6$ | 0 | | 0.0017 | 0.0011624 |

**Table 1.4**: Comparison of AD, LW model and the Bayesian model for artificial array data set 2: First 10 rows report the gene expression index estimated by average difference, LW model and our Bayesian model. The samples (1,5,8) with the outliers are highlighted in **bold** font. The last 6 rows report the probe-specific affinities, which shows that both models give estimates that are close to the true value. This shows that estimates of LW model are seriously affected by outliers, while the Bayesian model is robust to the outliers.

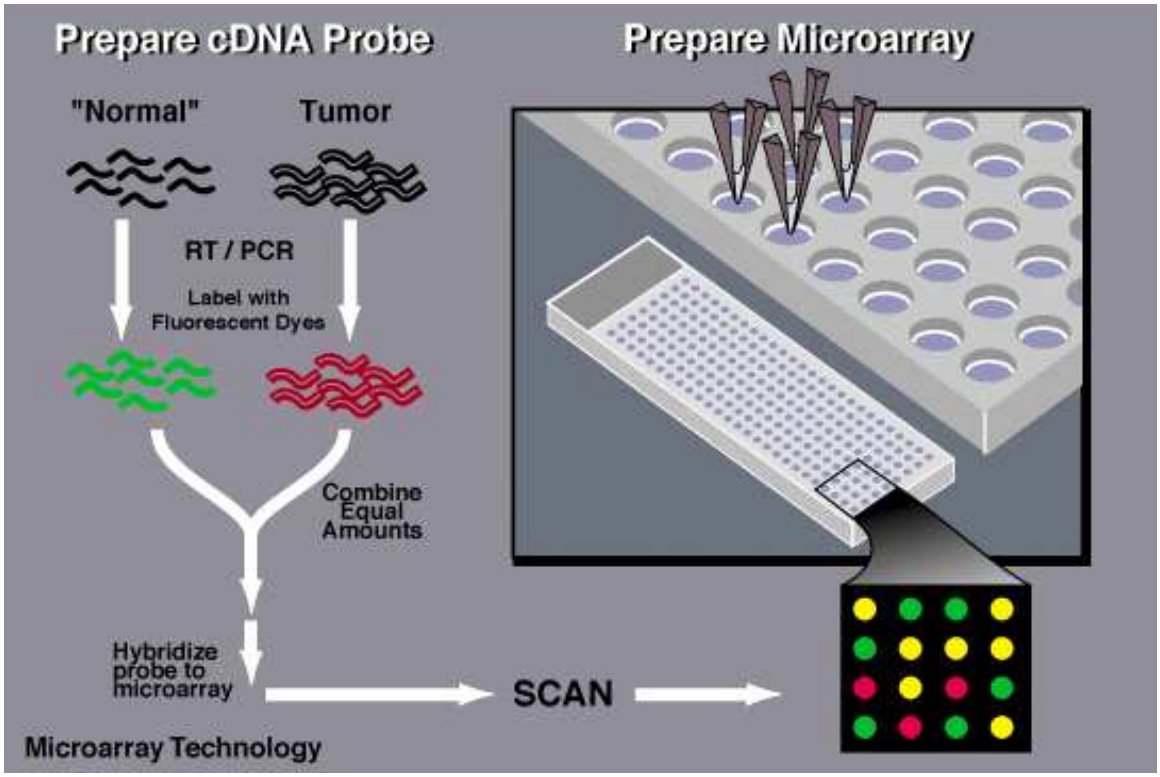


**Figure 1.1**: An overview of procedures for preparing cDNA microarrays: Selected probes are amplified by PCR and the PCR product is printed to a glass slide using a high-speed robot. The targets are labelled representations of cellular mRNA obtained by reverse transcriptions of total RNA extracted from the test and reference cells, and the pooled target is allowed to hybridize with the cDNA spotted on the slides. Once the hybridization is completed, the slides are washed and scanned with a scanning laser microscope able to measure the brightness of each fluorescent spot; brightness reveals how much of a specific DNA fragment is present in the target

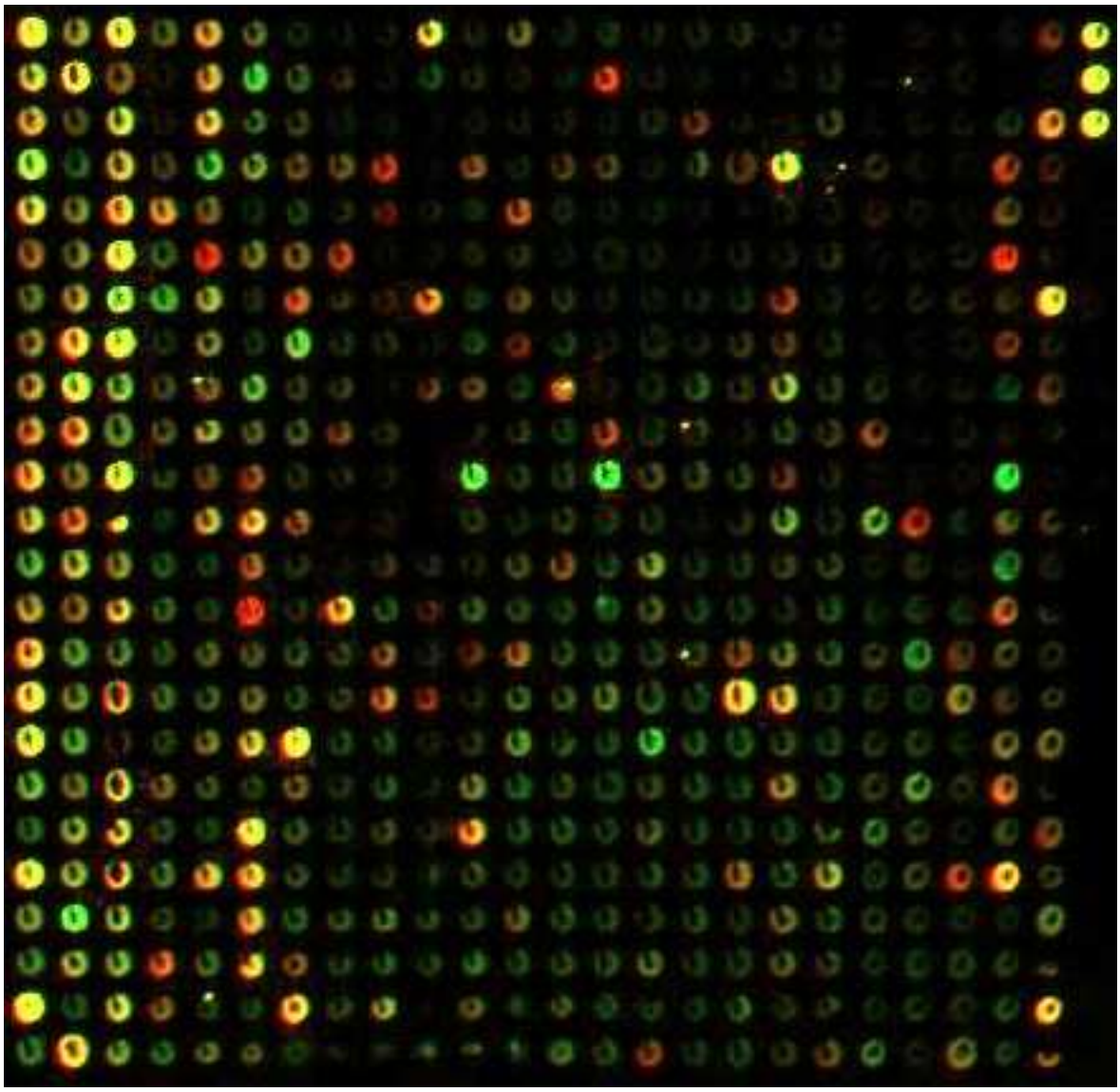

**Figure 1.2**: A scanned image produced from a cDNA microarray experiment: Each spot represents a gene. Grey spots denote genes that were expressed in neither type of cell; colored spots identify genes that were expressed in one of the two cells or both. The color of the spot discloses the relative expression of the gene in the two cells.

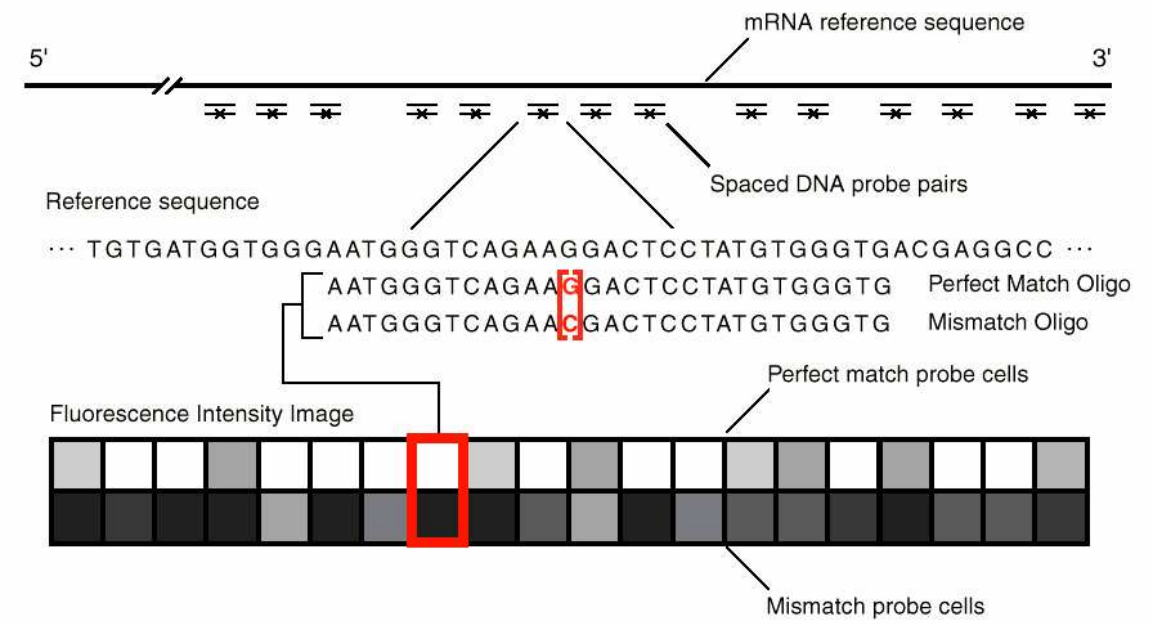


**Figure 1.3**: An oligonucleotide microarray associates a gene with a set of probe pairs: Each probe pair consists of a perfect match probe (PM) and a mismatch probe (MM). Each PM probe is 25 bases long and is paired with the MM probe, in which the central base of the oligonucleotide is inverted. After hybridization of the target to the probes, the microarray is read with a laser scanner to produce an image and the intensity of the MM probes is used to correct the intensity of the PM probes.

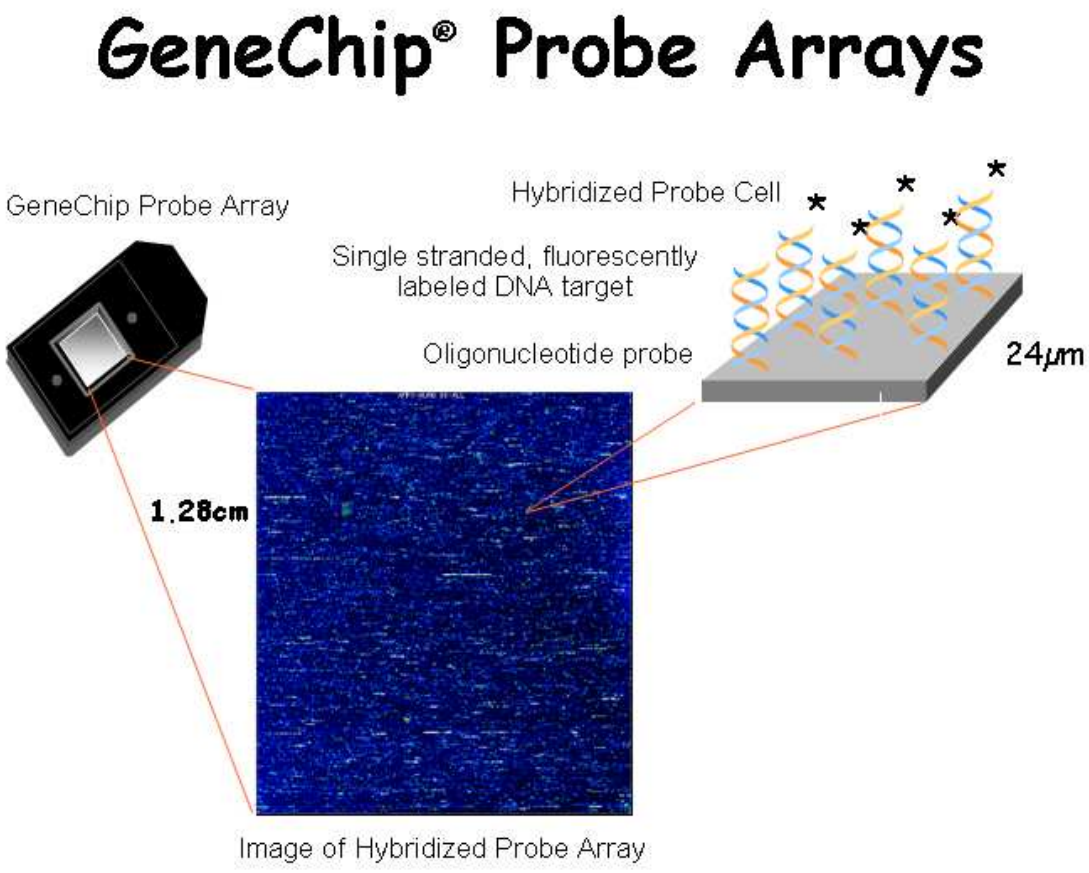


**Figure 1.4**: An illustrative example of probe cell, pixel and GeneChip Probe Array: Image is composed of probe cells, which contain probes that appear as pixels. Each probe cell contains millions of copies of a specific oligonucleotide probe. There are over 250,000 different probes complementary to genetic information of interest in a single array.

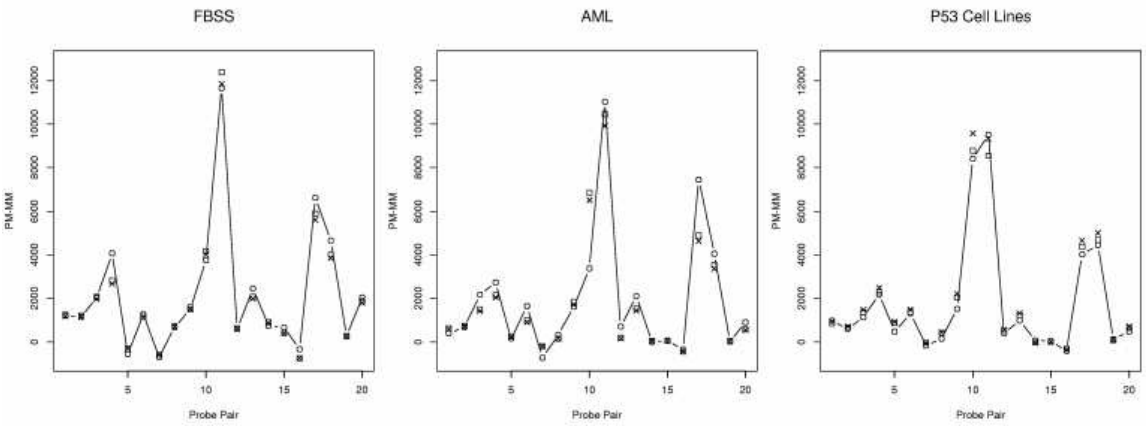


**Figure 1.5**: The probe-specific effects: The probe-specific effects in Oligonucleotide microarrays are highly reproducible and predictable: an example derived from Li and Wong (2001a)

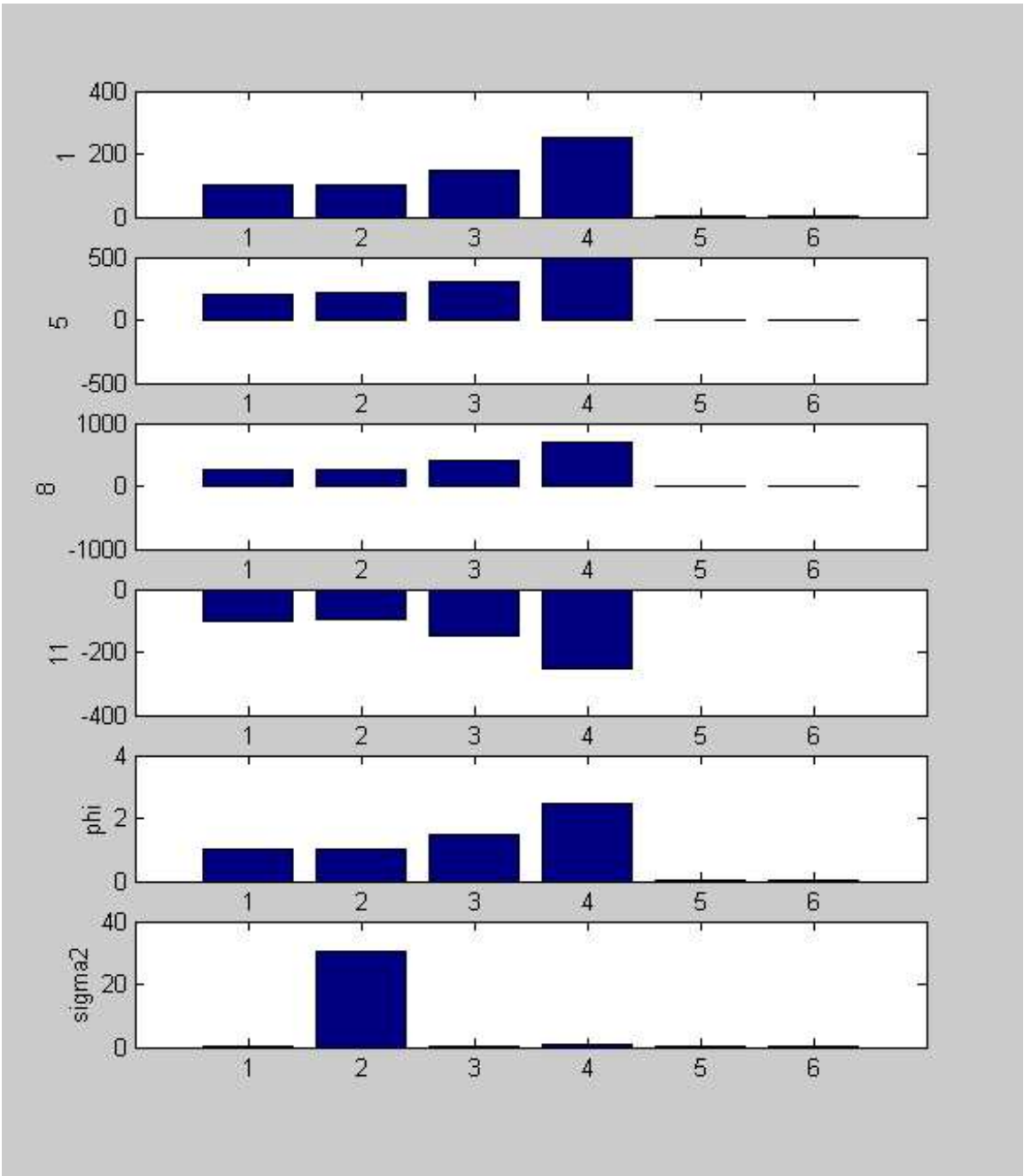


**Figure 1.6**: Artificial array data set 1: The first 4 rows are the bar plots of $y_{ij}$s in 4 arrays (1,5,8,11). The last two rows are the posterior means of $\phi$ and $\sigma^2$

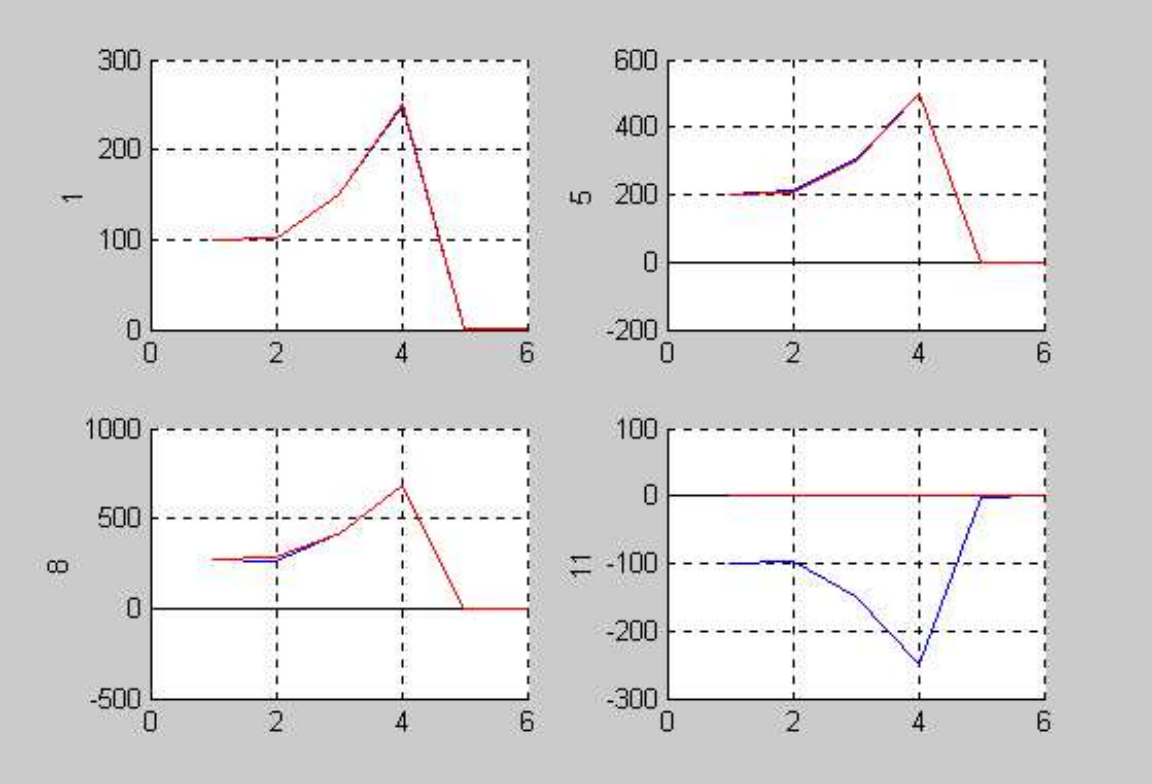


**Figure 1.7**: The fitted values of $\widehat{y}_{ij}$s of the Bayesian model for artificial array data set 1: The $x$ axis represents the probe pairs, and the $y$ axis represents the values of $\widehat{y}_{ij}$s. The blue line shows the original data $y_{ij}$s, while the red line shows the fitted values of $\widehat{y}_{ij}$s.

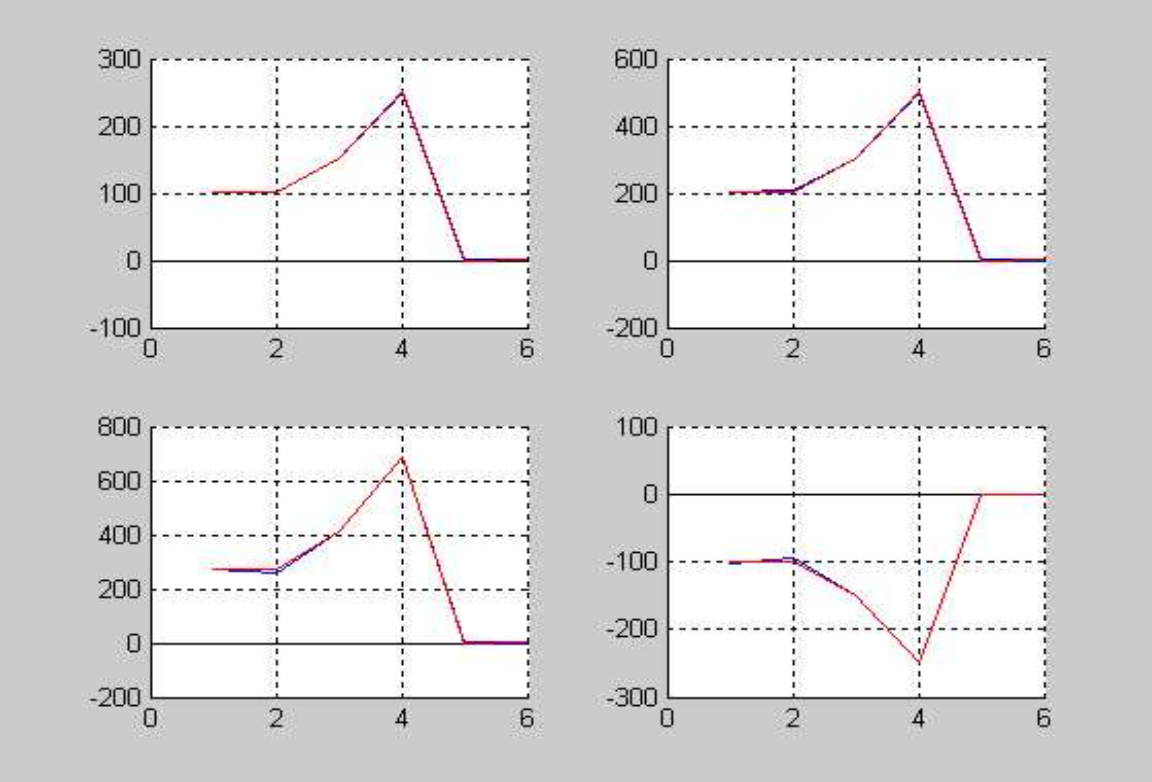


**Figure 1.8**: The fitted values of $\widehat{y}_{ij}$s of LW model for artificial array data set 1: The $x$ axis represents the probe pairs, and the $y$ axis represents the values of $\widehat{y}_{ij}$s. The blue line shows the original data $y_{ij}$s, while the red line shows the fitted values of $\widehat{y}_{ij}$s.

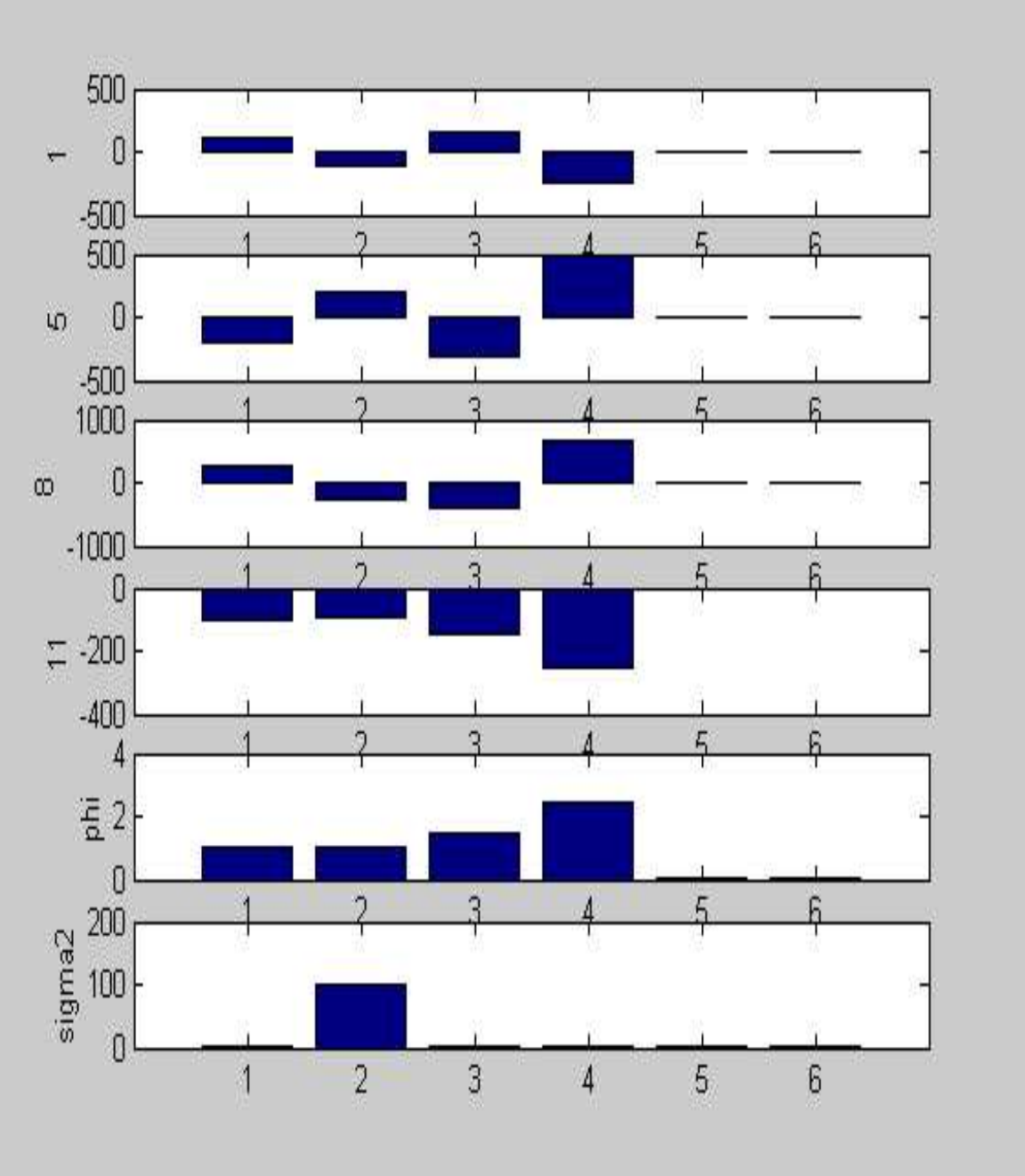


**Figure 1.9**: Artificial array data set 2: The first 4 rows are the bar plots of $y_{ij}$s in 4 arrays (1,5,8,11). The last two rows are the posterior means of $\phi$ and $\sigma^2$

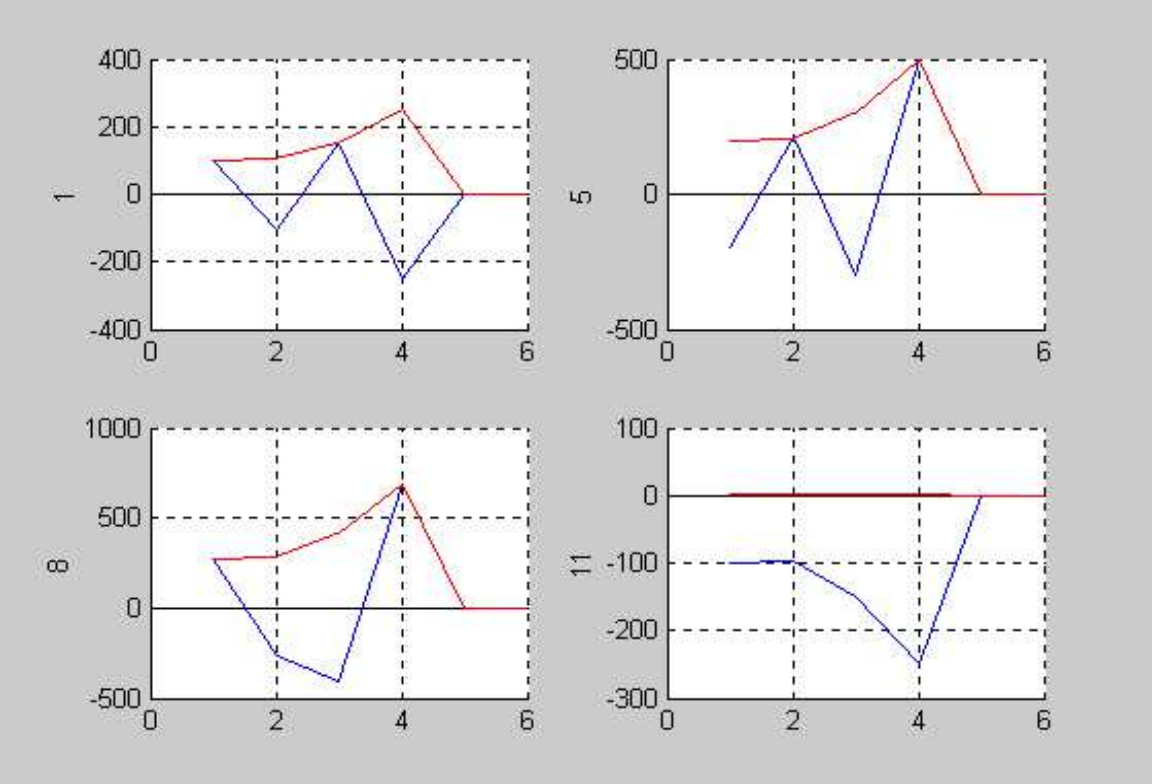


**Figure 1.10**: The fitted values of $\widehat{y}_{ij}$s of the Bayesian model for artificial array data set 2: The $x$ axis represents the probe pairs, and the $y$ axis represents the values of $\widehat{y}_{ij}$s. The blue line shows the original data $y_{ij}$s, while the red line shows the fitted values of $\widehat{y}_{ij}$s.

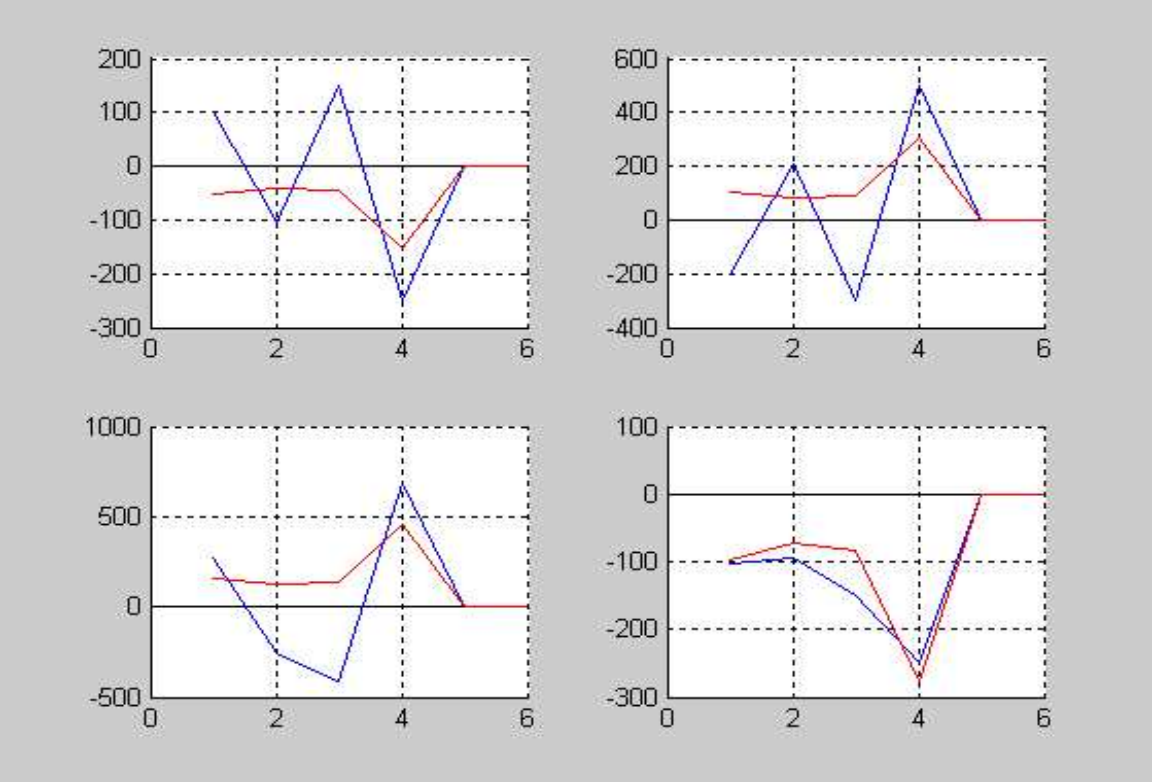


**Figure 1.11**: The fitted values of $\widehat{y}_{ij}$s of LW model for artificial array data set 2: The $x$ axis represents the probe pairs, and the $y$ axis represents the values of $\widehat{y}_{ij}$s. The blue line shows the original data $y_{ij}$s, while the red line shows the fitted values of $\widehat{y}_{ij}$s.

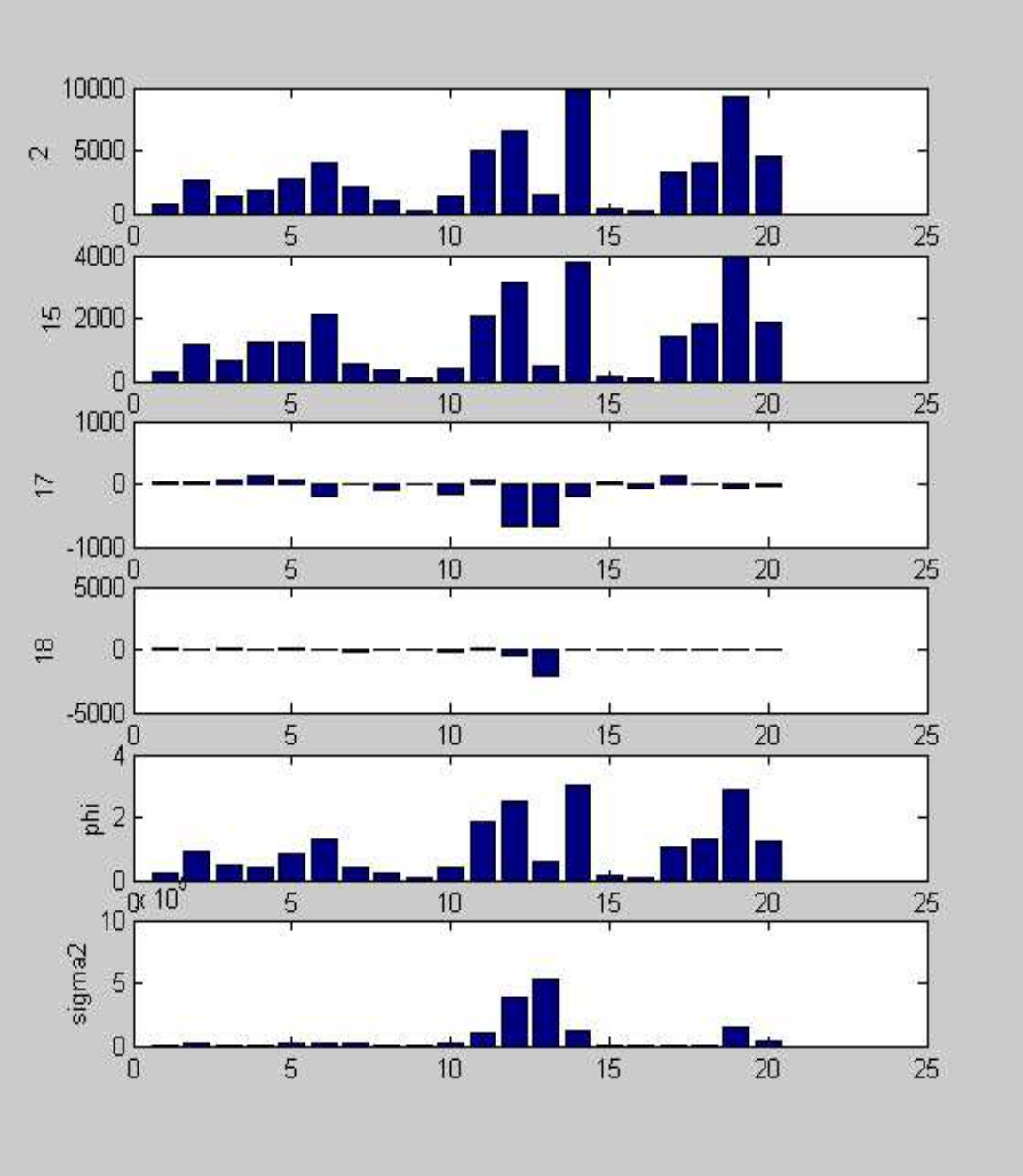


**Figure 1.12**: ER gene: The first 4 rows are the bar plots of $y_{ij}$s in 4 arrays (2, 15, 17, 18). The last two rows are the posterior means of $\phi$ and $\sigma^2$

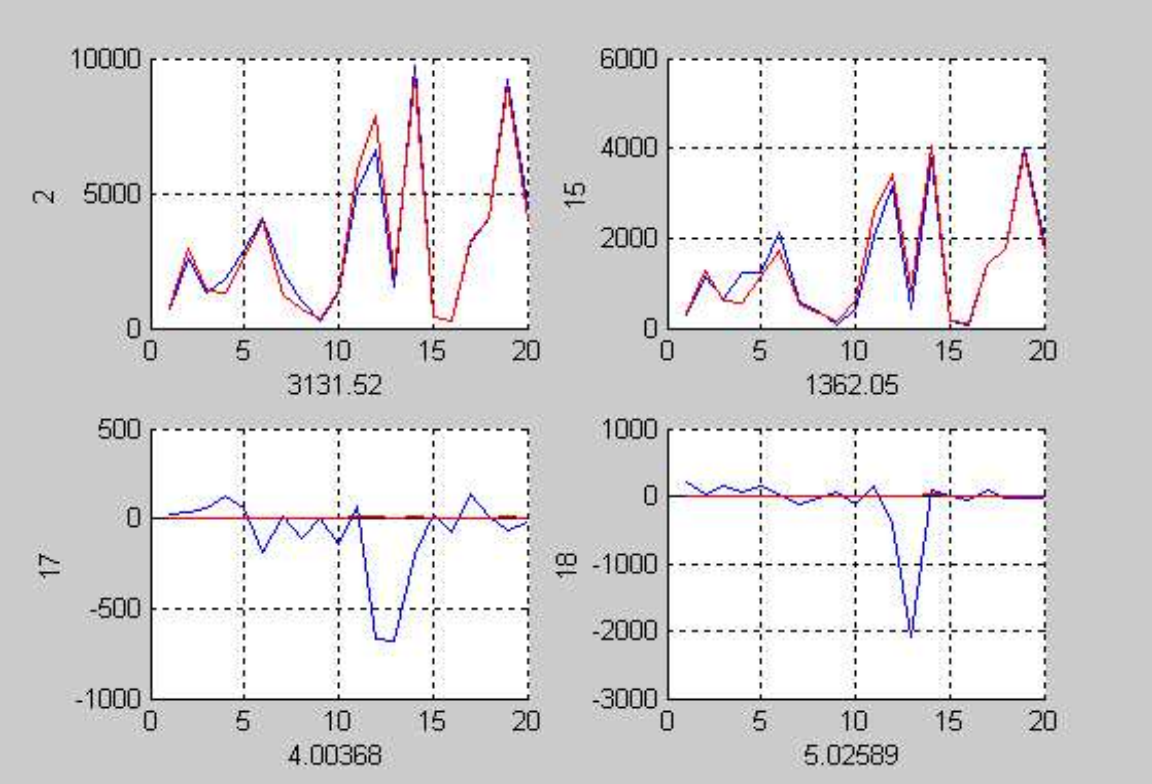


**Figure 1.13**: The fitted values of $\widehat{y}_{ij}$s of the Bayesian model for ER gene: The $x$ axis represents the probe pairs, and the $y$ axis represents the values of $\widehat{y}_{ij}$s. The blue line shows the original data $y_{ij}$s, while the red line shows the fitted values of $\widehat{y}_{ij}$s.

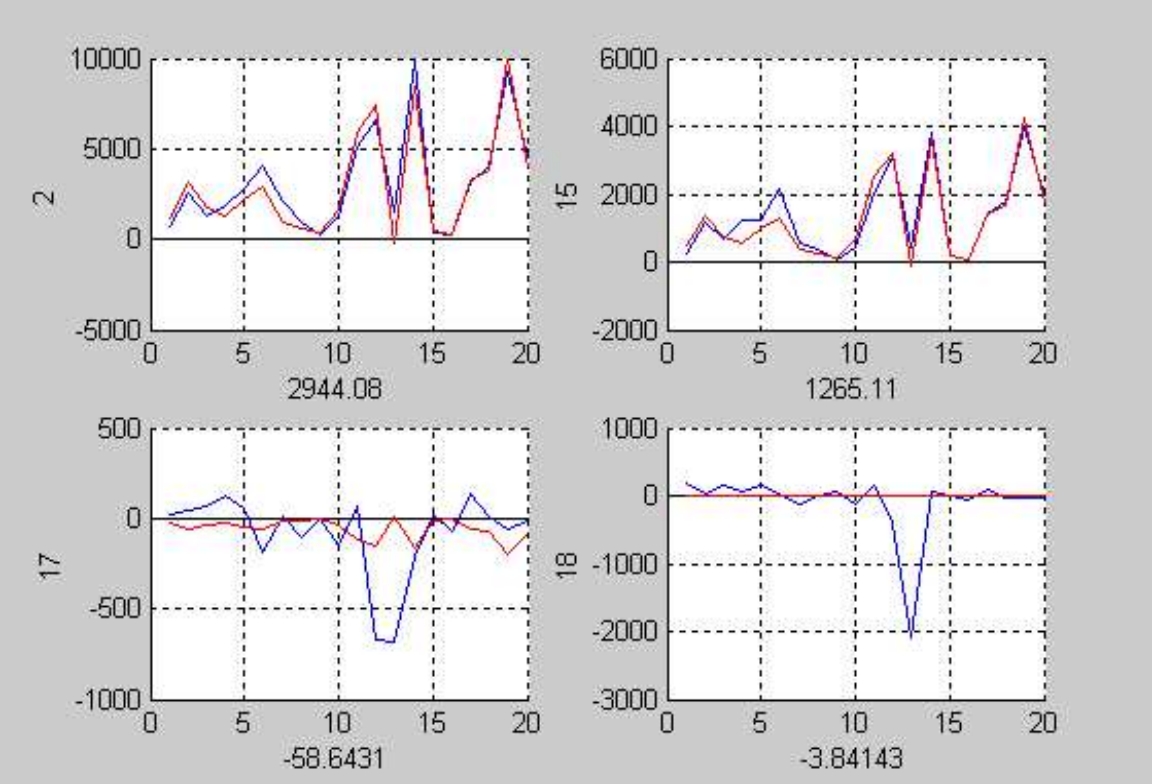


**Figure 1.14**: The fitted values of $\widehat{y}_{ij}$s of LW model for ER gene: The $x$ axis represents the probe pairs, and the $y$ axis represents the values of $\widehat{y}_{ij}$s. The blue line shows the original data $y_{ij}$s, while the red line shows the fitted values of $\widehat{y}_{ij}$s.

# Chapter 2

# Introduction to kernel models

## 2.1 Introduction

Kernel models (Schölkopf *et al.*, 1999; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) have recently been introduced and become increasingly popular tools for various regression, classification and function estimation problems. They exhibit good generalization performance on many real-life data sets, and the approaches are properly motivated theoretically. The most popular kernel models are support vector machines(SVMs) (Vapnik, 1995; Vapnik, 1998; Schölkopf, 1997) that are introduced in the field of machine learning, and the Gaussian processes (GPs) (Rasmussen, 1996; Williams and Barber, 1998; Williams, 1998) that are introduced in the field of Bayesian statistics. This Chapter includes a brief introduction to both support vector machines and Gaussian processes. Their strength and weakness are also discussed.

## 2.2 Basis function and kernel extension

In standard supervised learning problems, we are given a set of training data $D = (y_1, \mathbf{x}_1, y_2, \mathbf{x}_2, ..., y_n, \mathbf{x}_n)$, where the output variable $y_i$ might be a continuous variable

in the setting of regression analysis, or a categorical variable in the setting of classification. Usually, it is assumed that the training data are an independently and identically distributed sample from an unknown probability distribution $P(X, Y)$. We wish to learn a model of dependency of the output variables on the input variables, so that we can make accurate predictions of $y$ given a new input $\mathbf{x}$. Typically, our prediction is based on a function $f(\mathbf{x})$ defined over the input space. In the setting of linear regression, we assume the prediction function $f(\mathbf{x})$ is a linear function of $\mathbf{x}$, which can be written as

$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \epsilon_i, \\ f(\mathbf{x}_i) &= \mathbf{x}_i{}^T \beta, \\ \epsilon_i &\sim N(0, \sigma^2), \end{aligned}$$

where $\beta$ is a $p \times 1$ vector of regression coefficient parameters.

To extend linear models to nonlinear models, we can try to transform the data from input space $R^p$ into some high-dimensional feature space $H$ via a nonlinear basis function $\phi : R^p \rightarrow H$, and then construct the linear model in $H$

$$f(\mathbf{x}_i) = \phi(\mathbf{x}_i)^T \beta,$$

where $\beta$ is the regression coefficient vector in high-dimensional feature space $H$.

However, direct computation in such high or infinite feature space is often unrealistic. For instance, a $d$-degree polynomial transformation has $(p+d-1)!/(d!(p-1)!)$ different monomials. In the application of character recognition, a $16 \times 16$ pixel input images and a 5-degree polynomial transformation yield a dimensionality of $10^{10}$.

This problem can be overcome by the idea of kernel extension. The basic idea is to construct a model in where all the computation in feature space $H$ can be done in the form of dot product $(\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j))$ without explicit nonlinear mapping $\phi(\cdot)$. In

some cases, the dot products can be evaluated by a simple kernel function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j).$$

For example, the polynomial kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$$

can be shown to correspond to a map $\phi$ into the space spanned by all products of exactly $d$ dimension of $R^p$ (Poggio, 1975; Boser *et al.*, 1992; Burges, 1998). For $d = 2$ and $\mathbf{x} \in R^2$,e.g., we have

$$\begin{aligned}(\mathbf{x}_i^T \mathbf{x}_j)^2 &= (x_{1i}^2, x_{2i}^2, \sqrt{2} x_{1i} x_{1i})(x_{1j}^2, x_{2j}^2, \sqrt{2} x_{1j} x_{1j})^T \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j),\end{aligned}$$

defining

$$\phi(\mathbf{x}_i) = (x_{1i}^2, x_{2i}^2, \sqrt{2} x_{1i} x_{1i})^T.$$

By using

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d$$

with $c > 0$, we can take into account all products of order up to $d$.

More generally, Mercer's theorem (Mercer, 1909; Aizerman *et al.*, 1964; Boser *et al.*, 1992) in the area of functional analysis shows that kernels $k$s of positive integral operators give rise to implicit nonlinear mapping maps $\phi$s.

There are two major classes of models that take advantages of the idea of kernel extension: support vector machines(SVMs) and the Gaussian processes (GPs). SVMs come from the the field of machine learning, especially statistical learning theory, while GPs come from the field of Bayesian statistics. In the following sections, a brief introduction to both models is given.

## 2.3 Support vector machines

### 2.3.1 Linear classifier

To introduce the SVMs, we will begin by a simplest binary classification problem. From the perspective of statistical learning theory, the motivation for considering binary classifier SVMs comes from theoretical bounds on the generalizations error (Vapnik, 1995; Vapnik, 1998). Though we do not give a detailed introduction to the statistical learning theory here, we note that it has two important features. Firstly, the upper bound on the generalization error does not depend on the dimension of the space. Secondly, the error bound is minimised by maximising the margin, i.e., the minimal distance between the hyperplane separating the two classes and the closest data points to the hyperplane.

Let us consider a binary classification problem with a set of training data $D = (y_1, \mathbf{x}_1, y_2, \mathbf{x}_2, ..., y_n, \mathbf{x}_n)$, where the output variable $y_i$ is a binary variable $y_i \in \{\pm 1\}$. A simple linear discriminate function is:

$$y_i = sign(f(\mathbf{x}_i)), \tag{2.1}$$
$$f(\mathbf{x}_i) = \mathbf{x}_i{}^T\beta,$$

If the training data set is separable then the data will be correctly classified $y_i\mathbf{x}_i{}^T\beta > 0$ for all $i$. Since this relation is invariant under a positive rescaling of the argument inside the *sign* function, we can define a canonical hyperplane such that $y_i\mathbf{x}_i{}^T\beta = 1$ for the closest points on one side, and $y_i\mathbf{x}_i{}^T\beta = -1$ for the closest points on the other side. So we can rescale the parameter and place the constraints such that

$$y_i\mathbf{x}_i{}^T\beta \geq 1,$$

then the SVMs try to find the classifiers with maximum margin, which turns out to

be the following optimization problem:

$$\begin{array}{ll} \min & \frac{1}{2}\|\beta\|^2. \\ s.t. & y_i \mathbf{x}_i{}^T \beta \geq 1. \end{array} \tag{2.2}$$

This can be reduced to the minimization of primal Lagrangian objective function

$$L = \frac{1}{2}\beta^T\beta - \sum_{i=1}^{n} \alpha_i (y_i \mathbf{x}_i{}^T \beta - 1), \tag{2.3}$$

where the $\alpha_i \geq 0$ are Lagrange multipliers. Taking the derivative with respect to $\beta$ and equating to zero, i.e., $\partial L / \partial \beta = 0$, we can express $\beta$ in terms of dual variables,

$$\beta = \sum_j y_j \alpha_j \mathbf{x}_j,$$

which can be put back in the primal Lagrangian (2.3), it then becomes the Wolfe dual Lagrangian (Bertsekas, 1995)

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j, \tag{2.4}$$

which must be maximized with respect to the $\alpha_i$ subject to the constraint:

$$\alpha_i > 0, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

Then the prediction function $f(\mathbf{x})$ becomes

$$f(\mathbf{x}) = \sum_j y_j \alpha_j \mathbf{x}^T \mathbf{x}_j. \tag{2.5}$$

### 2.3.2 Nonlinear classifier via kernel extension

From the Wolfe dual Lagrangian (2.4) and prediction function (2.5), we notice that the data point $\mathbf{x}_i$ only appears inside an inner product which makes it possible to

use the kernel "trick" discussed in Section 2. Specifically, a SVM with nonlinear prediction function

$$f(\mathbf{x}_i) = \phi(\mathbf{x}_i)^T\beta,$$

is equivalent to the maximization of Wolfe dual Lagrangian

$$W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_i).$$

Let

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j),$$

which is the inner product in the higher dimensional Hilbert space. With a suitable choice of kernel, the data can become separable in feature space despite being non-separable in the original input space. Feasible kernels implicitly describing this mapping must satisfy Mercer's conditions described in more detail in Vapnik (1995) and Vapnik (1998). The class of mathematical objects which can be used as kernels is very general and includes a wide range of functions. Some commonly used kernel functions include the polynomial kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T\mathbf{x}_j)^d, \tag{2.6}$$

and Gaussian kernel

$$k(\mathbf{x}_j, \mathbf{x}_i) = \exp\left(-\rho^2\sum_{h=1}^{p}(x_{hi} - x_{hj})^2\right). \tag{2.7}$$

For the given choice of kernel, the learning task therefore involves maximization of

$$W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

with respect to the $\alpha_i$ subject to the constraints

$$\alpha_i > 0, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

The prediction function $f(\mathbf{x})$ becomes

$$f(\mathbf{x}) = \sum_j y_j \alpha_j k(\mathbf{x}_i, \mathbf{x}_j).$$

### 2.3.3 Soft margin

Most real life data sets contain noise, and two classes are not linear separable. An SVM can overfit noise leading to poor generalization. SVMs aim to deal with noise using soft margins, i.e., adopting a positive slack variable $\xi_i > 0$ such that

$$y_i {\mathbf{x}_i}^T \beta \geq 1 - \xi_i.$$

Then the learning procedure is to find the classifiers with maximum margin, which turns out to be the following optimization problem

$$\begin{array}{ll} \min & \frac{1}{2}\|\beta\|^2 + C\sum \xi_i, \\ s.t. & y_i {\mathbf{x}_i}^T \beta \geq 1 - \xi_i, \\ & \xi_i \geq 0, \end{array} \tag{2.8}$$

where $C$ is a constant. It can then be reformulated to the Wolfe dual in a way similar to that for hard margin.

### 2.3.4 Related models

Given the linear prediction function $f(\mathbf{x}_i) = {\mathbf{x}_i}^T \beta$, consider optimization problem (2.8) in the SVMs with soft margin. It is easy to rewrite the optimization problem as

$$\min_{\beta} \frac{1}{2C} \beta^T \beta + \sum_i [1 - y_i f(\mathbf{x}_i)]_+, \tag{2.9}$$

where the subscript "+" means the truncated positive function, i.e.,

$$[\mathbf{x}]_+ = \max(\mathbf{x}, 0).$$

We now view the SVMs in the Bayesian framework. Suppose $\beta$ has the prior

$$\beta \sim N(\beta|0, diag(C)),$$

and define the following likelihood function that is proportional to

$$p(\mathbf{y}|X, \beta) \propto \exp(-\sum_i [1 - y_i f(\mathbf{x}_i)]_+). \tag{2.10}$$

Then the optimization problem (2.9) is equivalent to finding the maximum a posteriori (MAP) estimation of $\beta$, i.e., minimizing an objective function

$$M(\beta) = -\log[p(\beta)p(\mathbf{y}|X,\beta)]. \tag{2.11}$$

While (2.10) is not differentiable, we can use the other more commonly used likelihood functions and transfer the optimization problem to the Wolfe dual Lagrangian form of optimization and prediction function as in SVMs, and then apply the idea of kernel extension to the nonlinear classifiers. For example, the normal likelihood function

$$p(\mathbf{y}|X, \beta) \propto N(\mathbf{y}|X^T\beta, \sigma^2)$$

results in least squares SVMs introduced in Suykens and Vandewalle (1999), and the logistic likelihood function

$$p(\mathbf{y}|X, \beta) \propto (1 + \exp(-\mathbf{y}X^T\beta))^{-1}$$

results in the logistic kernel regression (Hastie *et al.*, 2001). With other likelihood functions and priors for $\beta$, we can develop other nonlinear classifiers in a similar way.

Alternatively, we can define a loss function

$$L(\mathbf{y}, f) = -\log p(\mathbf{y}|X,\beta).$$

In SVMs, the loss function is

$$L(\mathbf{y}, f) = \sum_i [1 - y_i f(\mathbf{x}_i)]_+.$$

We can consider the term $\frac{1}{2C}\beta^T\beta$ as a regulation term, so that the objective function (2.9) in SVMs is the sum of a loss function and a regulation term. Different loss functions and regulations will result in different models.

A more general model arises from the regularized function estimation problems in the reproducing kernel Hilbert space (RKHS) (Aronszajn, 1950; Wahba, 1990; Wahba, 1999). Let $H_k$ be the RKHS generated by kernel $K$. Then the regularized function estimation can be solved by minimizing

$$\frac{1}{n}\sum_{j=1}^{n} L(y_i, f_i) + \lambda \left\|\mathbf{f}\right\|_{H_k}.$$

The representer theorem in Wahba (1990) tell us that the solution has the form

$$f(\mathbf{x}) = b + \sum_{j=1}^{n} w_j k(\mathbf{x}, \mathbf{x}_j),$$

where $b$, $\alpha_i$ are coefficients and $k(\mathbf{x}_j, \mathbf{x}) = \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}) \rangle$ is the kernel function between $\mathbf{x}_j$ and $\mathbf{x}$.

## 2.4 Gaussian processes

### 2.4.1 Bayesian linear regression

Let us consider a regression problem with a set of training data $D = (y_1, \mathbf{x}_1, y_2, \mathbf{x}_2, ..., y_n, \mathbf{x}_n)$, where the output variable $y_i$ is a continuous variable. A simple normal linear regres-

sion model is

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{2.12}$$
$$f(\mathbf{x}_i) = \mathbf{x}_i{}^T\beta,$$
$$\epsilon_i \sim N(\epsilon_i|0, \sigma^2).$$

Let the regression coefficients $\beta$ have the Gaussian prior

$$\beta \sim N(\beta|0, \Sigma_\beta).$$

Suppose $\sigma^2$ is known. Then the posterior distribution for $\beta$ is

$$p(\beta|\mathbf{y}, X, \Sigma_\beta, \sigma^2) = N(\beta|\widehat{\beta}, V_\beta),$$

where

$$\widehat{\beta} = \frac{1}{\sigma^2} V_\beta X\mathbf{y}, \tag{2.13}$$

and

$$V_\beta^{-1} = \Sigma_\beta^{-1} + \frac{1}{\sigma^2} XX^T.$$

To make a prediction $y^*$ for a new cases $\mathbf{x}^*$, we can derive the prediction distribution

$$\begin{aligned} P(y^*|\mathbf{y}) &= \int P(y^*, \beta|\mathbf{y}) d\beta \\ &= \int P(y^*|\beta, \mathbf{y}) P(\beta|\mathbf{y}) d\beta \\ &= \frac{1}{P(\mathbf{y})} \int P(y^*|\beta) P(\beta) P(\mathbf{y}|\beta) d\mathbf{y}, \end{aligned}$$

which is analytically tractable because everything inside the integration is Gaussians. The mean of $y^*$ is

$$E(y^*) = \frac{1}{\sigma^2} \mathbf{x}^{*T} V_\beta X\mathbf{y}.$$

### 2.4.2 Gaussian processes

In Bayesian linear regression, the uncertainty is described through a probability distribution over the regression coefficients $\beta$. It is also possible to deal directly with uncertainty with respect to the function values at the points in which we are interested. This is the stochastic process or function space view of the model. A stochastic process $\mathbf{f}(\mathbf{x})$ is a collection of random variables indexed by $\mathbf{x}$. A general stochastic process is specified by giving the probability distributions of any finite subset $(\mathbf{f}(\mathbf{x}_1), \mathbf{f}(\mathbf{x}_2), ...\mathbf{f}(\mathbf{x}_n))$ in a consistent way. Gaussian processes are a subset of stochastic processes that can be specified by giving only the mean vector and covariance matrix for any finite subset of points.

We now consider the linear regression problem in the view of Gaussian processes. Suppose $\beta$ has the prior

$$\beta \sim N(\beta|0, \Sigma_\beta).$$

Then $\mathbf{f} = X^T\beta$ is just a linear combination of Gaussian variables in function space, which is also a Gaussian processes, since

$$\mathbf{f} \sim N(\mathbf{f}|0, X^T\Sigma_\beta X).$$

Here $C = X^T\Sigma_\beta X$ be the covariance matrix. We can generalize beyond linear regression with other covariance matrices $C$ to define the GPs as

$$\mathbf{f} \sim N(\mathbf{f}|0, C).$$

In the setting of non-linear regression with Gaussian noise, $y_i = f(\mathbf{x}_i) + \epsilon_i$, $\mathbf{y}$ is also an Gaussian processes as

$$\mathbf{y} \sim N(\mathbf{y}|0, C + \sigma^2 I_{n\times n}).$$

To make a prediction $y^*$ for a new case $\mathbf{x}^*$, we need to calculate the joint distribution of $(y_1, y_2, ..., y_n, y^*)$, then get the conditional distribution $p(y^*|\mathbf{y})$. $\mathbf{y}_+ =$

$(y_1, y_2, ..., y_n, y^*)$ has joint Gaussian distribution with mean 0 and covariance matrix $K_+$. Let the $(n+1) \times (n+1)$ matrix $K_+$ be partitioned into a $n \times n$ matrix $K$, a $n \times 1$ vector $\mathbf{k}$ and a scalar $k^*$

$$K_+ = \begin{pmatrix} K & \mathbf{k} \\ \mathbf{k}^T & k^* \end{pmatrix}.$$

Derived from the properties of multivariate Gaussian distributions, the conditional distribution $p(y^*|\mathbf{y})$ is

$$p(y^*|\mathbf{y}) = N(y^*|\mathbf{k}^T K^{-1}\mathbf{y}, k^* - \mathbf{k}^T K^{-1}\mathbf{k}.)$$

In the setting of normal regression, we have

$$E(y^*) = k(\mathbf{x}^*, X)^T (C + \sigma^2 I_{n\times n})^{-1}\mathbf{y}$$

and the variance is

$$var(y^*) = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, X)^T (K + \sigma^2 I_n)^{-1} k(\mathbf{x}_*, X).$$

Now, we are ready to apply the idea of kernel extension in SVM to Bayesian regression. Note that a zero-mean GPs is solely based on its covariance functions $C_{ij} = c(\mathbf{x}_i, \mathbf{x}_j)$. In the above linear regression as a special example,

$$c(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \Sigma_\beta \mathbf{x}_j.$$

Formally the covariance function can be any function that will generate a non-negative definite covariance matrix for any set of points $(\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n)$. Just like the kernel functions in SVMs, the covariance functions in GPs can be considered as the dot product $(\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j))$

$$c(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \Sigma_\beta \phi(\mathbf{x}_j)$$

where $\phi(\mathbf{x})$ is the basis function in high-dimensional, even infinite dimensional feature space $F$. Then GPs can be viewed as the Bayesian linear regression on the basis functions

$$f(\mathbf{x}_i) = \phi(\mathbf{x}_i)^T \beta,$$

where $\beta$ is the regression coefficient vector in high-dimensional feature space $F$.

If the covariance functions $c(\mathbf{x}_i, \mathbf{x}_j)$ is known or fixed, it is straightforward to make inference and prediction based on the training data. In practice, covariance functions are generally chosen from a parametric family of covariance functions with a parameter vector $\theta$, and then we need to also make inference on $\theta$. For example, a commonly used covariance function is

$$c(\mathbf{x}_j, \mathbf{x}_i) = \exp\left(-\sum_{h=1}^{p} \rho_h^2 (x_{hi} - x_{hj})^2\right), \tag{2.14}$$

where $\rho_h^2$ is an individual scale parameter for each dimension $h = 1, 2, ..., p$. This class of covariance functions is closely related to the principle of automatic relevance determination (MacKay, 1994; MacKay, 1995; Neal and Hinton, 1995). This class of covariance functions is seldom used in SVMs because it is very hard to make inference on such complex covariance functions in the frequentist framework. In contrast, it is relatively straightforward to make a fully Bayesian inference on kernel parameters in GPs with the help of MCMC. Williams and Rasmussen (1996), Rasmussen (1996) and Neal (1998) have used MCMC, especially Hybrid Monte Carlo methods (Duane *et al.*, 1987), to obtain samples of $\theta$. We will apply MCMC to a new class of Bayesian kernel models in the next Chapter.

## 2.5 Summary

In this Chapter, we give a brief introduction to the nonlinear kernel models that are based on the idea of kernel extensions. There are two major classes of kernel models:

support vector machines (SVMs) and the Gaussian processes (GPs). SVMs are very popular recently in the field of machine learning and data mining. GPs have a long tradition in the field of Bayesian statistics, and have been rediscovered recently for the purpose of predictive modelling.

Though SVMs and related models have been successfully applied to a wide range of applications, they do have a number of significant disadvantages. One of major disadvantages is that SVMs are not probabilistic models, so they can only provide point estimates instead of proper predictive distributions. Another disadvantage is that it is difficult for SVMs to get reliable estimates of model parameters, such as the parameters in the kernel functions. Thus only simple kernel functions with few tunable parameters are used in SVMs generally, which greatly affects performance in some real world applications.

Compared to SVM, GPs has have a number of advantages (Williams and Barber, 1998): it is a valid probabilistic model; it is easy to be integrated into other probabilistic models, such as factor analysis, mixture models etc.; it is relatively easy to build a flexible hierarchical model, especially for the adapting parameters inside the kernel function, and inference on these parameters is straightforward in the Bayesian framework.

However, SVM does have one advantage the GPs: the solution is sparse, so it is possible to develop fast algorithms for large data sets. Instead, GPs have to invert a $n \times n$ covariance matrix, which makes it infeasible for large data set. In the next Chapter, we propose a new class of Bayesian kernel models, which provides sparse solutions besides keeping all the benefits of Bayesian learning.

# Chapter 3

# Bayesian kernel models

## 3.1 Introduction and notation

In standard supervised learning problems, we are given the training data that includes input $X$ and continuous output $\mathbf{y}$ in the setting of regression analysis, or categorical output $\mathbf{z}$ in the setting of classification analysis. Our goal is to find a prediction function $f(\mathbf{x})$ so that we can make accurate predictions of $\mathbf{y}$ or $\mathbf{z}$ given a new input $\mathbf{x}$. Throughout this chapter, we denote $X$ as the $p \times n$ design matrix $X_{p\times n}$; $\mathbf{x}_i$ as the $p$-dimensional vector of $i^{th}$ sample, i.e., the $i^{th}$ column of $X$; $X_j$ as the $j^{th}$ predictor, i.e., the $j^{th}$ row of $X$; and $x_{ij}$ as the $i^{th}$ and $j^{th}$ entry of $X$.

### 3.1.1 Support vector machines and related models

In Chapter 2, we discussed the support vector machines and related kernel models. In the setting of binary classification, SVMs make prediction for a new case $\mathbf{x}$ based on the function:

$$\begin{aligned} f(\mathbf{x}) &= \phi(\mathbf{x})^T\beta \\ &= b + \sum_{j=1}^{n} z_j a_j k(\mathbf{x}_j, \mathbf{x}). \end{aligned}$$

where $z_j$ are the binary responses, $b$, $\alpha_i$ are model parameters and $k(\mathbf{x}_j, \mathbf{x}) = \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}) \rangle$ is the kernel function between $\mathbf{x}_j$ and $\mathbf{x}$. Under the framework of maximum margin classifiers, SVMs try to find the solution by minimizing an error function on the training set while simultaneously maximizing the margin between two classes.

A more general model can be derived by the regularized function estimation in the reproducing kernel Hilbert space (RKHS) (Aronszajn, 1950; Wahba, 1990; Wahba, 1999). Let $H_k$ be the RKHS generated by kernel $K$ and $g(\cdot)$ is a cost function. Then the regularized function estimation can be solved by minimizing

$$\frac{1}{n}\sum_{i=1}^{n} g(z_i, f(\mathbf{x}_i)) + \lambda \left\| f \right\|_{H_k}, \tag{3.1}$$

and, by the representor theorem in Wahba (1990), the solution has the form

$$f(\mathbf{x}) = b + \sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}). \tag{3.2}$$

By using the kernel function $k(\mathbf{x}_j, \mathbf{x})$, we can perform all necessary computation in the form of dot product $\langle \phi(\mathbf{x}_j), \phi(\mathbf{x}) \rangle$ without the explicit nonlinear mapping $\phi : R^p \to F$, where $F$ could have a very high, or even infinite, dimensionality.

Though SVMs and related kernel models have been successfully applied to a wide range of applications, they do have a number of significant disadvantages. One of major disadvantages is that SVMs are not probabilistic models, so can only provide the point estimate of 'hard' predictions, i.e., 0 or 1. Another related disadvantage is that it is difficult for SVMs to get the reliable estimate of the model parameters, such as the parameters in the kernel functions. Thus only simple kernel functions with a few tunable parameters are used in SVMs generally. A common choice of kernel function in SVMs and other kernel methods is the Gaussian kernel

$$k(\mathbf{x}_j, \mathbf{x}_i) = \exp\left(-\rho^2 \sum_{h=1}^{p} (x_{hi} - x_{hj})^2\right), \tag{3.3}$$

where $\rho^2$ is the global scale parameter that is generally pre-selected or chosen by cross-validation. Though this kernel function works well for many practical problems, the performance of corresponding kernel models deteriorates severely when many predictor variables are irrelevant. This deterioration is due to the fact that each predictor variable is given the same weight to contribute to the kernel function no matter whether it is relevant or not.

To illustrate this problem, we give a nonlinear binary classification example that will be used throughout this chapter. We draw 60 samples independently from a mixture of 4 bivariate Gaussians

$$\mathbf{x}_i \sim \sum_{j=1}^{4} \pi_j N(\mathbf{x}_i | u_j, \Sigma_j),$$

where

$$\pi_1 = \pi_2 = \pi_3 = \pi_4 = 0.25,$$
$$\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = diag([1, 1]),$$
$$u_1 = [1, 0]', u_2 = [-1, 0]', u_3 = [0, 1]', u_4 = [0, -1]'.$$

We set the response $z_i = 0$ for component 1 and component 2 and $z_i = 1$ for component 3 and component 4. Thus, the class conditional distribution is

$$p(\mathbf{x}_i | t_i = 0) = \sum_{j=1}^{2} \pi_j N(\mathbf{x}_i | u_j, \Sigma_j),$$

$$p(\mathbf{x}_i | t_i = 1) = \sum_{j=3}^{4} \pi_j N(\mathbf{x}_i | u_j, \Sigma_j).$$

These 60 samples of $(\mathbf{x}_i, z_i)$ are considered as the training data and another 100 independent samples from the above mixture of Gaussians are used as test data to evaluate the predictive performance. The visualization of this XOR training and test data is shown in Figure 3.1 and Figure 3.2, where blue '+' represents class 0

and red 'o' represents class 1. It is a relatively easy problem for SVMs with the kernel function (3.3) taking $\rho^2 = 1$. The error rate for test data is 5%. We now complicate the classification problem by adding 8-dimensional independent Gaussian noise, $x_{3i}, ..., x_{10i}$, to the training data, and the data are divided into two classes only by values of the first two dimension $x_{1i}, x_{2i}$. Then the performance of SVMs, whose error rate increase to 45%, deteriorates severely because of the fact that the kernel function (3.3) is calculated using all 10 dimensional data with the same weight.

A more reasonable kernel function for the problem above is

$$k(\mathbf{x}_j, \mathbf{x}_i) = \exp\left(-\sum_{h=1}^{p} \rho_h^2 (x_{hi} - x_{hj})^2\right), \tag{3.4}$$

where $\rho_h^2$ is an individual scale parameters for each dimension $h = 1, 2, ..., p$. This class of kernel functions is closely related to the principle of automatic relevance determination (MacKay, 1994; MacKay, 1995; Neal and Hinton, 1995). If $\rho_h^2$ is close to zero, then the corresponding predictor variable $X_h$ will have little effect on the kernel function. Ideally, the scale parameters $\rho_3^2, \rho_4^2, ..., \rho_{10}^2$ should be adjusted to be close to zeros in the above simulation example.

The kernel function (3.4) is seldom used in SVMs and other relative models because it is very hard to make inference of such complex kernel function in the frequentist framework. In contrast, it is relatively in principal straightforward to make a fully Bayesian inference of above kernel parameters with the help of MCMC, which will be shown in section 3.6.

In this chapter, we propose a proper probabilistic kernel model in the Bayesian framework. Compared to the SVMs, our model can provide more reliable prediction via full posterior distributions. With the help of MCMC, we can make fully Bayesian inference on model parameters, even under complex model forms and kernel functions. Our model is also highly flexible and may be extended and applied in various

situations, such as regression, binary classification, multi-class classification, robust regression etc.

### 3.1.2 Comparison to relevance vector machine

We notice that our model is closely related to the recently proposed relevance vector machines (RVM) introduced in Tipping (2001) and Bishop and Tipping (2003). RVM adopts similar prediction functions in SVMs and fits the model in the Bayesian framework. Though RVM has been shown to have better performance than SVMs, there are some potential problems in both theory and practice.

- The equation (3.2) comes from the solution of equation (3.1) in a frequentist framework. RVM simply extracts the form of the equation (3.2) and applies the Bayesian computation directly. For example, RVM has the following model form for a normal regression model:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{3.5}$$

$$f(\mathbf{x}_i) = \sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}_i),$$

$$\epsilon_i \sim N(0, \sigma^2),$$

  and the model parameters are estimated in the Bayesian framework. However, such direct adoption of the equation (3.2) is not appropriate. In fact, it is not a proper model from a Bayesian perspective, because the model (3.5) depends on the sample sizes of training data. In contrast, our model is based on a proper model form in the Bayesian framework.

- RVM is based on the type-II maximum likelihood estimation with a series of approximations. First, the marginal posteriors for hyperpriors are approximated by the delta-function at its mode. Second, in the classification models, the

posteriors for prediction are approximated by the second order approximation, i.e., the Laplace approximation. Both approximations could be very inaccurate in practice. In contrast, we make fully Bayesian inference with the help of MCMC, which can provide more reliable estimation.

- RVM doesn't address the problem of estimation of kernel parameters in complex kernel functions such as (3.4). In contrast, we can make fully Bayesian inference of kernel parameters with the help of MCMC.

### 3.1.3 Organization of the chapter

The remainder of this chapter is organized as follows. Section 2 gives a brief introduction to radial basis functions (RBFs) that is the theoretic foundation of our model. In Section 3, we derive the Bayesian version of RBFs by utilizing the Dirichlet process priors on the distribution of location variables, which results in Bayesian kernel models as a special case. To achieve sparse solutions, we introduce two classes of structured priors for regression parameters $\mathbf{w}$ in Section 4: mixture priors with zero point masses, and Student-t priors, which result in our Bayesian kernel models. In Section 5, we introduce the orthogonalized kernel model to achieve better model mixing and speed-up the computation for problems with large sample sizes $n$. This model utilizes the reduced-rank singular value decomposition (SVD) to approximate the kernel matrix, which can be interpreted as a kernel principle component regression. We also address the problem of selecting the number of SVD factors via Bayes factors in this section. In Section 6, we focus on Bayesian inference for kernel parameters and propose a new empirical updating algorithm. For all the models introduced in this chapter, we develop both MCMC algorithms for fully Bayesian inference and EM algorithms for MAP estimation.

## 3.2 Radial basis functions

We are interested in the regression of a response variable $\mathbf{y}$ on a set of predictor variables $X$, given the training data set $D = (y_1, \mathbf{x}_1, y_2, \mathbf{x}_2, ..., y_n, \mathbf{x}_n)$. In a linear regression model, we assume $f(X) = E(\mathbf{y}|X)$ is a linear function of $X$. For example, a normal linear regression model can be presented as:

$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \epsilon_i, \\ f(\mathbf{x}_i) &= \mathbf{x}_i{}^T\beta, \\ \epsilon_i &\sim N(0, \sigma^2), \end{aligned} \tag{3.6}$$

where $\beta$ is a $p \times 1$ vector of regression coefficient parameters.

To extend the linear model to nonlinear cases, one simple idea is applying basis expansions, i.e., mapping the data from input space $R^p$ into some high-dimensional feature space $F$ via a nonlinear basis function $\phi : R^p \rightarrow F$, then performing linear algorithm in $F$, i.e., $f(\mathbf{x}_i)$ has the following form

$$\begin{aligned} f(\mathbf{x}_i) &= \phi(\mathbf{x}_i)^T\beta \\ &= \sum_{j=1}^{m} \beta_j \phi_j(\mathbf{x}_i) \end{aligned} \tag{3.7}$$

where $\beta$ is the regression coefficient vector in high-dimensional feature space $F$.

There are a lot of choices of basis functions, which correspond to a broad range of models, such as polynomials model, splines, multi-layer perceptrons (MLPs) (Ripley, 1996), multivariate adaptive regression splines (MARS) (Friedman, 1991), generalized additive models (Hastie and Tibshirani, 1990), etc.

Some of the families of basis functions are defined locally so that a simple model can be fitted in a region local to the target point $\mathbf{x}_*$, which is the basic idea of local regression. One special kind of such models is radial basis functions (RBFs) (Lowe, 1995), which combine the idea of nonlinear mapping via basis expansions and

localization via a weighting kernel. RBFs implement the idea by treating the basis function $\phi_j(\mathbf{x}_*)$ as a radial basis function $\phi_j(||\mathbf{x}_* - \mathbf{u}_j||)$, which is parameterized by a location (center/knot) vector $\mathbf{u}_j$ located in $p$-dimensional predictor variables space. In RBFs, the basis function is only a function of distance of target point $\mathbf{x}_*$ to their basis location $\mathbf{u}_j$. The model can be written as

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{3.8}$$

$$f(\mathbf{x}_i) = \sum_{j=1}^{m} w_j \phi(||\mathbf{x}_i - \mathbf{u}_j||).$$

Depending on our a priori assumption on the smoothness of nonlinear mapping, we can choose many different forms of basis functions (Girosi *et al.*, 1995). Some common choices include:

- Linear

$$\phi(d) = d$$

- Cubic

$$\phi(d) = d^3$$

- Thin-plate spline

$$\phi(d) = d^2 \log d$$

- Multiquadric

$$\phi(d) = (d^2 + c^2)^{1/2}$$

- Gaussian

$$\phi(d) = \exp(-cd^2)$$

A basis function is a function of only the distance of target point $\mathbf{x}_*$ to the basis location $\mathbf{u}_j : ||\mathbf{x}_* - \mathbf{u}_j||$, which is not necessarily a good metric for predictive purposes

when there are many irrelevant variables in high dimensional input space, just as we have shown in the previous section. It is straightforward to extend the basis functions $\phi(||\mathbf{x}_i - \mathbf{u}_j||)$ to more flexible kernel function $k(\mathbf{x}_i, \mathbf{u}_j)$, thus the RBFs becomes:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{3.9}$$

$$f(\mathbf{x}_i) = \sum_{j=1}^{m} w_j k(\mathbf{x}_i, \mathbf{u}_j).$$

Similar to the selection of the number of hidden neurons in a neural network, an essential part of RBFs modelling is the selection of $m$, the number of locations and the estimation of the locations $\mathbf{u}_j$. The frequentist methods include:

- Unsupervised learning. The centers are calculated by clustering method (Moody and Darken, 1989), such as agglomerative clustering, $k-$means, self organizing map (Kohonen, 1995) etc.

- Penalized likelihood. A penalty term is added to the likelihood function for functional regularization. Classical examples include Akaike information criterion (AIC) (Akaike, 1974), Bayesian information criterion (BIC) (Schwarz, 1978), and minimum description length (MDL) (Rissanen, 1987).

- Predictive assessment. The data is split into the training set and validation set, then the model is chosen to balance the bias and variance of prediction. Classical methods include bootstrap, jackknife (Chernick *et al.*, 1985; Efron, 1982), and cross-validation (Stone, 1974).

- Growing and pruning. The number of locations is set by growing and pruning algorithm. Examples include upstart algorithm (Frean, 1990), cascade correlation (Fahlman and Lebiere, 1990), optimal brain damage (Le Cun *et al.*, 1990) and the resource allocating network (Platt, 1991).

## 3.3 Bayesian RBFs

Recently, there has been considerable interest in applying the Bayesian framework and MCMC methods to neural network and radial basis functions. Neal and Hinton (1995) proves that certain classes of priors for neural networks, in which the number of hidden neurons tends to infinity, converge to Gaussian processes. Insua and Muller (1998) and Holmes and Mallick (1998) use the growing and pruning algorithm to select the number of hidden neurons from a Bayesian perspective. By applying the reversible jump MCMC (Green, 1995; Richardson and Green, 1997) to MLP and RBFs, they compute the joint posterior distribution of the model parameters and the number of hidden neurons.

Here we also address the issue about the number of locations in RBFs from a Bayesian perspective. The innovations here include use of a Dirichlet Process (DP) prior (Ferguson, 1983) on the distribution of the location variables without trying to select their number explicitly. It will be shown that our Bayesian definition, interpretation and rationale of RBFs is very close to the support vector machine and smoothing splines from the frequentist perspective.

### 3.3.1 A Bayesian perspective

From the Bayesian perspective, we can consider $\mathbf{u}$ as a random variable which has the unknown distribution $F(\cdot)$, and $\mathbf{x}_1, ..., \mathbf{x}_n$ are $i.i.d$ samples from $F$. Then the RBFs (3.8) can be represented as

$$f(\mathbf{x}) = \int_{-\infty}^{+\infty} w(\mathbf{u}) k(\mathbf{x}, \mathbf{u}) dF(\mathbf{u}), \tag{3.10}$$

or

$$f(\mathbf{x}) = \sum_{i=1}^{+\infty} w(\mathbf{u}_i) k(\mathbf{x}, \mathbf{u}_i) p(\mathbf{u}_i),$$

which can be considered as an infinite RBF network.

First, we will analyze the model (3.10) in the form of Gaussian processes.

Suppose $w(\mathbf{x})$ has a Gaussian process prior $w \sim N(0, C)$, where $C$ is the covariance matrix with the elements

$$\begin{aligned} c_{i,j} &= cov(w(\mathbf{x}_i), w(\mathbf{x}_j)) \\ &= c(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

In (3.10), the latent function $f(\mathbf{x})$ is also a Gaussian process because it is a linear function of a Gaussian process. The mean of $f(\mathbf{x})$ is zero, i.e.,

$$E(f(\mathbf{x})) = 0,$$

and the covariances are

$$\begin{aligned} Cov(f(\mathbf{x}_i), f(\mathbf{x}_j)) &= E(\int_{-\infty}^{+\infty} w(\mathbf{u})k(\mathbf{x}_i, \mathbf{u})dF(f))E(\int_{-\infty}^{+\infty} w(v)k(\mathbf{x}_j, v)dF(v)) \\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} E(w(\mathbf{u}))E(w(v))k(\mathbf{x}_i, \mathbf{u})w(v)k(\mathbf{x}_{j,}v)dF(v)dF(\mathbf{u}) \\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} c(\mathbf{u}, v)k(\mathbf{x}_i, \mathbf{u})k(\mathbf{x}_j, v)dF(v)dF(\mathbf{u}). \end{aligned} \tag{3.11}$$

As a simple case, we assume that $w(\mathbf{x})$ is a white noise process, the kernel function $k$ is stationary, and $\mathbf{u}$ is uniformly distributed over a region $U$. Thus we have

$$f(x) = \int_U w(\mathbf{u})k(\mathbf{x} - \mathbf{u})du. \tag{3.12}$$

The function $f(\mathbf{x})$ can be considered as a convolution of a random process with a smooth kernel $k$. It is easy to prove that $f(x)$ is also a stationary Gaussian process with the covariance function

$$cov(f(\mathbf{x}_i), f(\mathbf{x}_j)) = \int_U k(\mathbf{x}_i - \mathbf{u})k(\mathbf{x}_j - \mathbf{u})du.$$

A related model is illustrated by Higdon (2000) in detail. Instead of modelling $f(\mathbf{x})$ as a Gaussian process directly, we can construct a new Gaussian process $f(\mathbf{x})$ over a general spatial or temporal region $U$ by convolving another Gaussian process, with a smoothing kernel $k(s)$, in which the covariance function $cov(w(\mathbf{x}_i), w(\mathbf{x}_j))$ can be simple and it is not necessarily a strictly positive function.

This idea is very useful for the situations where data do not provide enough information to estimate the covariance function $cov(f(\mathbf{x}_i), f(\mathbf{x}_j))$ of a standard Gaussian process, but enough information to draw inferences on the simpler covariance function $cov(w(\mathbf{x}_i), w(\mathbf{x}_j))$. Such situations exist in many inverse problems. A good example is fluid flow problems (Lee *et al.*, 2002), in which a Markov random field is used to model $w(\mathbf{u})$.

As an approximation to (3.12),

$$f(\mathbf{x}) = \sum_{j=1}^{m} k(\mathbf{x} - \mathbf{u}_j) w(\mathbf{u}_j)$$

is still a continuous process while the underlying process is discrete. Such an approximation on a finite set of basis points can save much computing time since a close approximation only needs a relatively small number of basis points $\mathbf{u}_j$ in some situations.

In the following, we run some simple one-dimensional simulations to show what $f(\mathbf{x})$ looks like under various assumptions on $k$ and $w$.

By setting $w(\mathbf{u})$ as a white noise process and $F$ as $U(-1, 1)$, we draw 1,000 samples of $w$ and $\mathbf{u}$ to calculate the function $f(x)$. Figure 3.3 show the three samples of function $f(\mathbf{x})$ by using the Gaussian kernel, i.e.,

$$k(\mathbf{x}, \mathbf{u}) = \exp\left(-\rho^2(\mathbf{x} - \mathbf{u})^2\right),$$

where the kernel parameter $\rho$ has four different values: $0.1, 1, 4, 10$.

As discussed, with a stationary smoothing kernel $k$ , $f(\mathbf{x})$ is just a stationary Gaussian process.

Though such a nonparametric Gaussian process model has many appealing features, it still suffers from the huge computational cost for large data sets as earlier discussion. So, we now consider $f(\mathbf{x})$ as a deterministic function of $\mathbf{x}$ given all of the model parameters. To derive the form of $f(\mathbf{x})$, we have to specify $F(\cdot)$. To make $f(\mathbf{x})$ based on a finite set of knots, we should specify $F(\cdot)$ as a discrete distribution. There are many possible choices of discrete distribution, and different choices will result in different models. The family of Dirichlet processes provide a one commonly used framework for uncertain discrete distributions. In the following, we explore the use of the Dirichlet process to model $F$ in (3.10) to develop a more accessible analysis.

### 3.3.2 New models based on the Dirichlet process prior

In this section, we introduce a novel kernel model by using Dirichlet process priors. Dirichlet processes were first introduced in the area of Bayesian non-parametric modelling. The early papers include Ferguson (1973), Ferguson (1974) and Antoniak (1974). The theories and applications of Dirichlet process were further enriched in Ferguson (1983), introducing the field of Dirichlet process mixture models. With the advent of Markov chain Monte Carlo methods in 1990s, the computation of Dirichlet process mixture models and Bayesian non-parametric cluster models were developed in the papers of Escobar (1994), West *et al.* (1994), Escobar and West (1995) and Green and Richardson (1998). In the following, we give a brief introduction to Dirichlet process priors and basic properties.

A Dirichlet process prior for $F$, denoted by $F \sim DP(F|F_0, \alpha)$, is determined by two parameters: $F_0$ is the "baseline" prior and $\alpha$ is the positive scalar precision parameter. $F_0$ defines the "location" of Dirichlet process prior. The precision pa-

rameter determines the concentration of prior for $F$ around the prior "guess" $F_0$, and therefore measures the "strength of belief" in $F_0$. For large values of $\alpha$, a sample of $F$ is likely to be close to $F_0$. For small values of $\alpha$, a sample of $F$ is likely to put most of its probability mass on a few atoms. Denoting $\mathbf{x}_1, ..., \mathbf{x}_n$ as $i.i.d$ draws from $F$, Ferguson (1983) discovered the following result:

$$\begin{aligned}
\mathbf{x}_1 &\sim F_0(\mathbf{x}), \\
\mathbf{x}_2|\mathbf{x}_1 &\sim \frac{\alpha}{\alpha+1}F_0(\mathbf{x}) + \frac{1}{\alpha+1}\delta_{\mathbf{x}_1}(\mathbf{x}_2), \\
\mathbf{x}_3|\mathbf{x}_2, \mathbf{x}_1 &\sim \frac{\alpha}{\alpha+2}F_0(\mathbf{x}) + \frac{1}{\alpha+2}\delta_{\mathbf{x}_1}(\mathbf{x}_3) + \frac{1}{\alpha+2}\delta_{\mathbf{x}_2}(\mathbf{x}_3), \\
&.... \\
\mathbf{x}_n|\mathbf{x}_1, ..., \mathbf{x}_{n-1} &\sim \frac{\alpha}{\alpha+n-1}F_0(\mathbf{x}) + \frac{1}{\alpha+n-1}\sum_{j=1}^{n-1}\delta_{\mathbf{x}_j}(\mathbf{x}_n),
\end{aligned}$$

where $\delta_{\mathbf{x}_j}(\mathbf{x}_n)$ denotes a unit point mass at $\mathbf{x}_n = \mathbf{x}_j$. The joint distribution of $X = (\mathbf{x}_1, ..., \mathbf{x}_n)$ is the product of the above conditional components, namely

$$\prod_{i=1}^{n}\frac{\alpha F_0(\mathbf{x}_i) + \sum_{j=1}^{i-1}\delta_{\mathbf{x}_j}(\mathbf{x}_i)}{\alpha+i-1},$$

and the full conditional distribution of $(\mathbf{x}_i|\mathbf{x}_1, ..., \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ...\mathbf{x}_n)$, for $i = 1, ..., n$, is

$$\mathbf{x}_i|\mathbf{x}_1, ..., \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ...\mathbf{x}_n \sim \frac{\alpha}{\alpha+n-1}F_0(\mathbf{x}_i) + \frac{1}{\alpha+n-1}\sum_{j\neq i}\delta_{\mathbf{x}_j}(\mathbf{x}_i).$$

Given that $\mathbf{x}_1, ..., \mathbf{x}_n$ are $i.i.d$ draws from $F$ , we know that a new sample $\mathbf{u}$ has the distribution

$$\mathbf{u}|\mathbf{x}_1, ..., \mathbf{x}_n \sim \frac{\alpha}{\alpha+n}F_0(\mathbf{u}) + \frac{1}{\alpha+n}\sum_{j=1}^{n}\delta_{\mathbf{x}_j}(\mathbf{u}). \tag{3.13}$$

Thus, given existing samples, the next sample represents a new, distinct value drawn from $F_0$ with probability $\alpha/(\alpha+n)$, and is otherwise drawn uniformly from among

the first $n$ values $\mathbf{x}_1, ..., \mathbf{x}_n$. Based on the properties of DPs, the expectation of a function $g$ on a new sample $\mathbf{u}$ is

$$E(g(\mathbf{u})|\mathbf{x}_1, ..., \mathbf{x}_n) = \frac{\alpha}{\alpha + n} E_{F_0}(g(\mathbf{u})) + \frac{1}{\alpha + n} \sum_{j=1}^{n} g(\mathbf{x}_j),$$

where

$$E_{F_0}[g(\mathbf{u})] = \int_{-\infty}^{+\infty} g(\mathbf{u}) dF_0(\mathbf{u}).$$

In some cases, those $n$ values will reduce to some $k_n < n$ distinct values $\mathbf{x}_1^*, ..., \mathbf{x}_{k_n}^*$ with positive probability. Suppose there are $n_j$ occurrences of $\mathbf{x}_j^*$ and let $I_j$ be the index set of those occurrences; thus $\mathbf{x}_i = \mathbf{x}_j^*$ for $i \in I_j$ and $j = 1, ..., k$, with $n_1 + n_2 + ... + n_{k_n} = n$. Correspondingly, (3.13) reduces to the mixture of fewer components (West, 1992; Escobar and West, 1995):

$$\mathbf{u}|\mathbf{x}_1, ..., \mathbf{x}_n \sim \frac{\alpha}{\alpha + n} F_0(\mathbf{u}) + \frac{1}{\alpha + n} \sum_{j=1}^{k_n} n_j \delta_{\mathbf{x}_j^*}(\mathbf{u}). \tag{3.14}$$

Antoniak (1974) gives the prior for $k_n$ induced by the above DPs model. The prior distribution for $k_n$ may be written as

$$k_n|\alpha, n \sim c_n(k) n! \alpha^{k_n} \frac{\Gamma(\alpha)}{\Gamma(\alpha + n)}, \qquad (k_n = 1, 2, ..., n),$$

where $c_n(k) = P(k_n|\alpha = 1, n)$, not involving $\alpha$. West (1992) gives the asymptotic distribution of $k_n$. For $k_n = o(\log(n))$, $k_n - 1$ has a Possion distribution. In particular, for $n$ moderately large, the expectation of $k_n$ is about $\alpha \ln(1 + n/\alpha)$, which is an increasing function of $\alpha$.

In the following, we show some simple one-dimensional simulations of $f(\mathbf{x})$ based on the Dirichlet process prior.

By setting $w(\mathbf{u})$ as a white noise process and the baseline prior $F_0$ as $U(-1,1)$, we draw 1,000 samples of $\mathbf{u}$ according to the properties of DPs:

$$\begin{aligned}
\mathbf{u}_1 &\sim F_0(\mathbf{u}),\\
\mathbf{u}_2|\mathbf{u}_1 &\sim \frac{\alpha}{\alpha+1}F_0(\mathbf{u}) + \frac{1}{\alpha+1}\delta_{\mathbf{u}_1}(\mathbf{u}_2),\\
\mathbf{u}_3|\mathbf{u}_1,\mathbf{u}_2 &\sim \frac{\alpha}{\alpha+2}F_0(\mathbf{u}) + \frac{1}{\alpha+2}\delta_{\mathbf{u}_1}(\mathbf{u}_3) + \frac{1}{\alpha+2}\delta_{\mathbf{u}_2}(\mathbf{u}_3),\\
&....\\
\mathbf{u}_n|\mathbf{u}_1,...,\mathbf{u}_{n-1} &\sim \frac{\alpha}{\alpha+n-1}F_0(\mathbf{u}) + \frac{1}{\alpha+n-1}\sum_{j=1}^{n-1}\delta_{\mathbf{u}_j}(\mathbf{u}_n),
\end{aligned}$$

Then we calculate the function $f(x)$ by using the Gaussian kernel, i.e.,

$$k(x,u) = \exp\left(-\rho^2(x-u)^2\right).$$

Figure 3.4 shows the three samples of $f(x)$ when $\alpha = 1$ and $\rho = 0.1, 1, 4, 10$ respectively. As we can see, due to the Dirichlet process prior, the function $f(x)$ is no longer stationary, and it is clearer with large value of $\rho$. To show the effect of different scalar precision parameters $\alpha$ on $f(x)$, we draw three samples of $f(x)$ when kernel parameter $\rho$ is fixed but $\alpha = 0, 1, 10, 1,000$ respectively. Figure 3.5 shows such simulations when the kernel parameter $\rho = 10$. As we can see, Dirichlet process prior introduces the non-stationarity. But as we increase the $\alpha$, we have more confidence on our baseline prior $F_0$, and $f(x)$ looks more like a stationary Gaussian process.

To better illustrate the effects of different scalar precision parameters $\alpha$, we assume that the first three samples of $\mathbf{u}$ are $\mathbf{u}_1 = -0.5, \mathbf{u}_2 = 0, \mathbf{u}_3 = 0.5$. Then we draw the rest of the samples in the same way above. Figure 3.6 shows the different functions $f(x)$ by different scalar precision parameters $\alpha = 0, 1, 10, 1,000$, with kernel parameter $\rho = 10$. As we can see, the function $f(x)$ has a high concentration on initial distinct values for small $\alpha$. But as we increase the $\alpha$, $\mathbf{u}$ has more distinct values

drawn from $F_0$. Thus the baseline prior $F_0$ has more smoothing effects on $f(x)$. It is consistent with the fact that the asymptotic mean of $k_n$ is an increasing function of $\alpha$ (West 1992). The larger $\alpha$ implies larger number of distinct values.

Besides the Gaussian kernel, we can also use other commonly used kernel functions, e.g., the sum of Gaussian kernel and linear kernel:

$$k(x, \mathbf{u}) = \exp\left(-\rho^2(x - \mathbf{u})^2\right) + xu.$$

Figure 3.7 shows the different functions $f(x)$ by different scalar precision parameters $\alpha = 0, 1, 10, 1000$ and kernel parameter $\rho$ equals 10.

### 3.3.3 A novel class of Bayesian kernel models

We now derive a novel class of Bayesian kernel models based on the Dirichlet process priors. We first consider the posterior predictive distribution of $y$ given a new input $\mathbf{x}$ and a set of observations $\mathbf{x}_1, ..., \mathbf{x}_n$ in the setting of normal regression. Supposing that $w(\mathbf{u})$ is a deterministic function of $\mathbf{u}$ and all of model parameters are fixed (we will discuss inference on $w$ and other parameters later), then the only uncertainty comes from $F$. Given $F$, we write our regression model as

$$P(y|\mathbf{x},F) = N(y|f(\mathbf{x}), \sigma^2),$$

where

$$f(\mathbf{x}) = \int_{-\infty}^{+\infty} w(\mathbf{u})k(\mathbf{x}, \mathbf{u})dF(\mathbf{u}).$$

Note that $P(y|\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x},F) = P(y|\mathbf{x},F)$ since $y$ is conditionally independent of $\mathbf{x}_1, ..., \mathbf{x}_n$ given $F$. Assuming the new input $\mathbf{x}$ is also a sampled from $F$, the posterior predictive distribution is

$$P(y|\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x}) = \int P(y|\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x},F)P(F|\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x})dF \tag{3.15}$$

$$= \int P(y|\mathbf{x},F)P(F|\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x})dF$$

Thus the predictive distribution is a very complex mixture of normals; the relevant density function $P(y|\mathbf{x},F)$ mixed with respect to the posterior uncertainty over $F$.

A fully Bayesian analysis requires the calculation of above posterior predictive distribution (3.15) and the inference of DP parameter $\alpha$ which is fully illustrated in West (1992). However it is impossible to calculate equation (3.15). To simplify our calculation, we use a "plug-in" approximation for large sample size $n$.

For a large $n$, we have two useful properties:

- The posterior density $P(F|\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x})$ is very concentrated on $E(F|\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x})$. Thus we can have the approximation

$$\begin{aligned} P(y|\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x}) &= \int P(y|\mathbf{x},F)P(F|\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x})dF \\ &\approx P(y|\mathbf{x},E(F|\mathbf{x_1},...,\mathbf{x_n},\mathbf{x})). \end{aligned}$$

- Let $\mathbf{x}_{n+1}=\mathbf{x}$. As $n$ increases, $E(F|\mathbf{x_1},...,\mathbf{x_n},\mathbf{x})$ approximates to the empirical CDF of the $\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x}_{n+1}$, which is denoted as $F_{n+1}$. Namely, given that $\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x}_{n+1}$, we know that a new sample $\mathbf{u}$ has the distribution

$$\mathbf{u}|\mathbf{x}_1,...,\mathbf{x}_{n+1} \sim \frac{1}{n+1}\sum_{j=1}^{n+1}\delta_{\mathbf{x}_j}(\mathbf{u}),$$

  and the expectation of a function $g$ on a new sample $\mathbf{u}$ is

$$E(g(\mathbf{u})|\mathbf{x}_1,...,\mathbf{x}_{n+1}) = \frac{1}{n+1}\sum_{j=1}^{n+1}g(\mathbf{x}_j).$$

In summary, we have the approximation

$$\begin{aligned} P(y|\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x}) &= \int P(y|\mathbf{x},F)P(F|\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x})dF \\ &\approx P(y|\mathbf{x},E(F|\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x})) \\ &\approx P(y|\mathbf{x},F_{n+1}), \end{aligned}$$

where

$$P(y|\mathbf{x},F_{n+1}) = N(y|f_{n+1}(\mathbf{x}),\sigma^2),$$

and

$$f_{n+1}(\mathbf{x}) = \int_{-\infty}^{+\infty} w(\mathbf{u})k(\mathbf{x},\mathbf{u})dF_{n+1}(\mathbf{u}) \tag{3.16}$$

$$= \frac{1}{n+1}\sum_{j=1}^{n+1} w(\mathbf{x}_j)k(\mathbf{x}_j,\mathbf{x}).$$

Thus the prediction function in our Bayesian kernel model becomes $f_{n+1}(\mathbf{x})$ which is an approximation of infinite RBF network (3.10).

When the $n+1$ values $\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x}_{n+1}$ reduce to some $k_n < n+1$ distinct values $\mathbf{x}_1^*,...,\mathbf{x}_{k_{n+1}}^*$, $f_{n+1}(\mathbf{x})$ (3.16) can be rewritten as

$$f_{n+1}(\mathbf{x}) = \frac{1}{n+1}\sum_{j=1}^{k_{n+1}} n_j w(\mathbf{x}_j^*)k(\mathbf{x}_j^*,\mathbf{x}).$$

Then $f_{n+1}(\mathbf{x})$ is reduced on $k_n$ locations.

Note that we assume the new input $\mathbf{x}$ is also a sampled from $F$ and is known at the time of modeling. This can be considered as a learning problem with both labeled and unlabeled data, which will be discussed in more detail later. However, in the setting of classical supervised learning problems, the new input $\mathbf{x}$ is somehow assumed to be unknown at the time of modeling. In this case, the posterior distribution of $F$ does not depend on $\mathbf{x}$. Thus the posterior predictive distribution of $y$ is modified as

$$P(y|\mathbf{x}_1,...,\mathbf{x}_n,\mathbf{x}) = \int P(y|\mathbf{x},F)P(F|\mathbf{x}_1,...,\mathbf{x}_n)dF$$

$$\approx P(y|\mathbf{x},F_n),$$

where

$$P(y|\mathbf{x},F_n) = N(y|f_n(\mathbf{x}),\sigma^2),$$

and

$$f_n(\mathbf{x}) = \int_{-\infty}^{+\infty} w(\mathbf{u})k(\mathbf{x},\mathbf{u})dF_n(\mathbf{u}) \tag{3.17}$$

$$= \frac{1}{n}\sum_{j=1}^{n} w(\mathbf{x}_j)k(\mathbf{x}_j,\mathbf{x}).$$

In the following, we will stick to the form (3.17) in the setting of classical supervised learning problems. This can be considered as a special kind of RBFs, in which locations are just each training data points. Moreover, this has the exact form of the representor theorem discussed in the previous chapter. However, there is a fundamental difference between the representor theorem and our Bayesian kernel model. The form of the representor theorem comes from the solution of regularized function estimation in the Reproducing Kernel Hilbert Space (RKHS). Our parametric kernel model comes from the proper Bayesian modelling of RBFs as a special case of Dirichlet process priors. One key implication is that it is therefore a coherent model for all $n$, and provides immediate access to Bayesian learning.

### 3.3.4 Likelihood

In the setting of normal regression, we have the kernel regression model from (3.17) in the following form:

$$y_i = f_n(\mathbf{x}_i) + \epsilon_i,$$

$$f_n(\mathbf{x}_i) = \sum_{j=1}^{n} w_j k(\mathbf{x}_j,\mathbf{x}_i),$$

$$\epsilon_i \sim N(0,\sigma^2),$$

where $w_j = w(\mathbf{x}_j)/n$. Note that the kernel weight $w_j$ depends on $n$, and hence priors must reflect this dependency.

By using the notation of classical linear regression model, we can consider the kernel matrix $K$ as a $n \times n$ design matrix, each column of which corresponds to a $n$-dimensional vector $\mathbf{k}_i$. We can rewrite our model as

$$y_i = f_n(\mathbf{x}_i) + \epsilon_i,$$
$$f_n(\mathbf{x}_i) = \mathbf{k}_i^T \mathbf{w},$$
$$\epsilon_i \sim N(0, \sigma^2),$$

which is simply a normal linear regression model of $\mathbf{y}$ on a set of predictor variables $K$.

Given the observed value of $\mathbf{y}$, the likelihood function for $\mathbf{w}$ and $\sigma^2$ is

$$p(\mathbf{y}|K, \sigma^2, \mathbf{w}) \propto \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{w}-\widehat{\mathbf{w}})^T K K^T (\mathbf{w}-\widehat{\mathbf{w}})\right\} \tag{3.18}$$

where $\widehat{\mathbf{w}}$ is the least-squares vector

$$\widehat{\mathbf{w}} = (KK^T)^{-1} K\mathbf{y}.$$

In the following, we will introduce two classes of structured priors for regression parameters $\mathbf{w}$. Again, it is important to note the dependence of $\mathbf{w}$ on $n$, though that is not explicitly reflected in the notation.

### 3.3.5 Mixture priors with point mass

One of key features of SVMs is the sparsity of its solution, i.e., only few of non-zero elements in $\mathbf{w}$. Let $s \subset \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, and the prediction function as

$$f_n(\mathbf{x}) = \sum_{j \in s} w_j k(\mathbf{x}_j, \mathbf{x}). \tag{3.19}$$

Suppose the sizes of $s$ is $m$ , a much smaller number than $n$; thus the sparse model depends only a small subset of kernel functions and corresponding training data.

In SVMs, the sparsity is achieved by the truncation property of its loss function $(1-zf)_+$. In the setting of standard regression analysis, the sparsity can be achieved using subset selection algorithms. For example, Zhu and Hastie (2001) adopt a greedy forward subset selection algorithm for kernel logistic regression, which is called Import Vector Machine.

In the Bayesian framework, we can put an hierarchical prior on the regression coefficients $w_j$ for the 'automatic' subset selection to achieve the sparsity. In this subsection, we assume that the prior for $w_j$ is an independent mixture of normal distribution and a point mass at zero, namely

$$w_j|b_j \sim b_j\delta_0 + (1-b_j)N(w_j|0, \sigma_w^2/n^2),$$

where $\delta_0$ is the point mass at zero, $\sigma_w^2$ is a large value that results in a diffuse prior for $w_j$ and $b_j \in \{0,1\}$ is the binary latent variable. In particular, $b_i = 1$ means that the $w_j$ would be so small that it could be estimated as zero and the corresponding $\mathbf{x}_j$ is not included in the model. Otherwise $b_i = 0$ means that the $w_j$ would be estimated as a nonzero value and and the corresponding $\mathbf{x}_j$ is included. To reflect our a priori belief that most of $w_j$ are 0, i.e., most of $b_j$ are 1, we can specify that $\pi_j = p(b_j = 1)$ has a prior heavily concentrated near one, such as $\pi_j \sim Beta(9,1)$. Note that the prior for $w_j$ depends on $n$ because of the fact that $w_j = w(\mathbf{x}_j)/n$, and thus the canonical hyperparameters to specify, or estimate, are $\sigma_w^2$ and $\pi_j$.

We now derive the conditional posterior distribution for $b_i$. Let

$$\begin{aligned} y_i^{(j)} &= y_i - \sum_{l \neq j} w_l k_{li} \\ &= w_j k_{ji} + \epsilon_i, \end{aligned}$$

and we can calculate

$$r = \frac{p(b_j = 1|D, w_{-j}, b_{-j})}{p(b_j = 0|D, w_{-j}, b_{-j})} \tag{3.20}$$

$$= \frac{\sigma_w \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i^{(j)}\right)^2\right\} p(b_j = 1).}{\sqrt{v_w}\exp\left\{-\frac{1}{2}\left[\sum_{i=1}^{n}\frac{\left(y_i^{(j)}\right)^2}{\sigma^2} - \left(\frac{\sum_{i=1}^{n} y_i^{(j)} k_{ji}}{\sigma^2}\right)^2 v_w\right]\right\} p(b_j = 0)},$$

where

$$\frac{1}{v_w} = \frac{\sum_{i=1}^{n} k_{ji}^2}{\sigma^2} + \frac{n^2}{\sigma_w^2}.$$

Then we have the conditional posterior distribution for $b_i$ as

$$b_j|D, w_{-j}, b_{-j} \sim Bern(\frac{r}{1+r}),$$

which can be plugged into the standard Gibbs sampling procedure to create a sequence of $b_i, w_i$ as follows: First, the samples of $b_j$ are drawn from above Bernoulli distribution. If $b_j = 1$, then $w_j$ is set as zero. If $b_j = 0$, then we draw the samples of $w_j$ according to the following conditional posterior distribution:

$$p(w_j|D, w_{-j}, b_j = 0) = N(w_j|((\sum_{i=1}^{n} y_i^{(j)} k_{ji})n^2/\sigma^2)v_w, v_w). \tag{3.21}$$

### 3.3.6 Student-t priors

As an alternative to the above mixture priors with point mass, a Student-t prior distribution can be considered as a continuous version of mixture distribution. In practice, we apply the following hierarchical prior (West, 1984; West, 2003) on the

regression coefficients $w_j$ :

$$p(\mathbf{w}|X,T)=\prod_{i=1}^{n} N(w_i|0,\frac{\sigma^2}{n^2}\tau_i^2), \tag{3.22}$$

$$= N(\mathbf{w}|\mathbf{0},\frac{\sigma^2}{n^2}T),$$

where $\tau_i^2$ are the prior scale parameters with $T = diag(\tau_1^2, \tau_2^2, ..., \tau_n^2)$. Again, it is important to note that the prior for $w_j$ depends on $n$ because of the fact that $w_j = w(\mathbf{x}_j)/n$. Also note that we use individual scale parameter $\tau_i^2$ for each regression parameter $w_i$, which allows for different degree of shrinkage in each of the predictor dimensions. As for the priors for $\tau_i^2$, we specify independent inverse gamma priors on each of $\tau_i^2$,

$$\tau_i^{-2} \sim Ga\left(\tau_i^{-2}\left|\frac{k}{2},\frac{k}{2}\tau_0^2\right.\right) \tag{3.23}$$

with shape parameter $k/2$ and scale parameter $k\tau_0^2/2$. This prior on $\tau_i^2$ means that the implied priors for regression parameters $w_i$, on marginalisation with respect to the $\tau_i^2$, are the independent $T$ priors with $k$ degrees of freedom. To introduce the sparsity of $\mathbf{w}$, $k$ is generally specified as a small integer, say $k = 2$. Thus the posteriors of some of the $w_i$ concentrate around zero, which implies that the associated data points play little role in the regression. The parameter $\tau_0^2$ controls the overall scale for the set of $\tau_i^2$. It can be a pre-selected constant or a hyperparameter with its own hyperprior as

$$\tau_0^2 \sim Exp(\tau_0^2|a_0), \tag{3.24}$$

where $a_0$ is the quantity to control overall scale of sparsity. To reflect our a priori belief that most of $\tau_i^2$ and corresponding $w_j$ are close to 0, we can specify a large $a_0$ such that $\tau_0^2$ has a prior heavily concentrated near zero.

In the setting of normal regression, the conjugate prior distribution for $\sigma^2$ is an

inverse gamma distribution,

$$p(\sigma^{-2}) = Ga\left(\sigma^{-2}\left|\frac{n_0}{2}, \frac{n_0}{2}\sigma_0^2\right.\right). \tag{3.25}$$

### 3.3.7 Conditional posteriors

Given the above likelihood and priors, we can get the full conditional posteriors. Under the specified prior (3.22) and likelihood function (3.18), the resulting conditional posterior for $\mathbf{w}$ is a multivariate normal posterior

$$p(\mathbf{w}|D, T, \sigma^2) = N(\mathbf{w}|\widehat{\mathbf{w}}, V_{\mathbf{w}}), \tag{3.26}$$

where

$$\widehat{\mathbf{w}} = \frac{n^2}{\sigma^2} V_{\mathbf{w}} K\mathbf{y},$$

$$V_{\mathbf{w}}^{-1} = \frac{n^2}{\sigma^2}\left(T^{-1} + KK^T\right).$$

Under the specified prior (3.25), the conditional posterior for $\sigma^2$ over $\mathbf{w}$ is the inverse gamma,

$$p(\sigma^{-2}|D, T, \mathbf{w}) = Ga\left(\sigma^{-2}\left|\frac{n_0 + 2n}{2}, \frac{n_0\sigma_0^2 + q}{2}\right.\right),$$

where

$$q = (\mathbf{y} - K^T\mathbf{w})^T(\mathbf{y} - K^T\mathbf{w}) + n^2\mathbf{w}^{\mathbf{T}}T^{-1}\mathbf{w}.$$

In practice, the conditional posterior for $\sigma^2$ can be marginalised over $\mathbf{w}$ to make MCMC more efficient, which leads to the inverse gamma form

$$p(\sigma^{-2}|D, T) = Ga\left(\sigma^{-2}\left|\frac{n_0 + n}{2}, \frac{n_0\sigma_0^2 + q}{2}\right.\right),$$

where

$$q = (\mathbf{y} - K^T\widehat{\mathbf{w}})^T(\mathbf{y} - K^T\widehat{\mathbf{w}}).$$

Thus it is straightforward to draw samples from the joint conditional posterior distribution $p(\mathbf{w},\sigma^{-2}|D,T)$ by drawing $\sigma^{-2}$ from its posterior marginalised over $\mathbf{w}$, i.e., $p(\sigma^{-2}|D,T)$, followed by a drawing of $\mathbf{w}$ given $\sigma^{-2}$ from $p(\mathbf{w}|D,T,\sigma^2)$.

Under the specified prior (3.23), the conditional posteriors for scale parameter $\tau_i^2$ is an independent inverse gamma posterior

$$p(\tau_i^{-2}|w_i,\tau_0^2,D) = Ga\left(\tau_i^{-2}\left|\frac{k+1}{2},\frac{k\tau_0^2+w_i^2}{2}\right.\right) \tag{3.27}$$

conditionally independent over $i$.

Under the specified prior (3.24), the conditional posterior for hyperparameter $\tau_0^2$ is a gamma posterior

$$p(\tau_0^2|T,D) = Ga\left(\tau_0^2\left|\frac{kn}{2}+1,\frac{k}{2}\sum_{i=1}^{n}\tau_i^{-2}+a_0\right.\right).$$

### 3.3.8 Model fitting via MCMC

Given the above conditional posteriors, it is straightforward to implement the posteriors simulation using standard Gibbs sampling that is illustrated in detailed below. The initial values are chosen arbitrarily for each of parameters, and then the samples of parameters are drawn from their conditional posterior distributions at each iteration. The components of each MCMC step for a regression model

$$y_i = \sum_{j=1}^{n} w_j k(\mathbf{x}_j,\mathbf{x}_i)+\epsilon_i,$$

$$\epsilon_i \sim N(0,\sigma^2),$$

are as follows:

- Draw $\sigma^{-2}$ from its marginal over $\mathbf{w}$ posterior

$$p(\sigma^{-2}|D,T) = Ga\left(\sigma^{-2}\left|\frac{n_0+n}{2},\frac{n_0\sigma_0^2+q}{2}\right.\right).$$

- Given the current value of $\sigma^{-2}$, draw a vector of $\mathbf{w} = (w_1, w_2, ...w_n)$ from the multivariate normal posterior

$$p(\mathbf{w}|D, T, \sigma^2) = N(\mathbf{w}|\widehat{\mathbf{w}}, V_{\mathbf{w}}).$$

- Given the current value of $\mathbf{w}$, draw a new diagonal matrix $T = diag(\tau_1^2, \tau_2^2, ..., \tau_n^2)$ from the $n$ independent gamma posteriors

$$p(\tau_i^{-2}|w_i, \tau_0^2) = Ga\left(\tau_i^{-2}\left|\frac{k+1}{2}, \frac{k\tau_0^2 + w_i^2}{2}\right.\right).$$

- Given the current value of $T = diag(\tau_1^2, \tau_2^2, ..., \tau_n^2)$, draw $\tau_0^2$ from the gamma posterior

$$p(\tau_0^2|T) = Ga\left(\tau_0^2\left|\frac{kn}{2} + 1, \frac{k}{2}\sum_{i=1}^{n}\tau_i^{-2} + a_0\right.\right).$$

For a binary classification model with binary output variable $z_i \in \{0, 1\}$, it is straightforward to modify the above MCMC algorithm using probit regression where $\mathbf{y} = (y_1, y_2, ..., y_n)^T$ are the latent variables. MCMC is run to generate the approximate samples from the joint posteriors $p(\mathbf{y}, \mathbf{w}, T|\mathbf{z}, X)$ (Albert and Chib, 1993; Johnson and Albert, 1999). The components of each MCMC steps for a probit regression model

$$z_i = \begin{cases} 1, & \text{if } y_i \geq 0, \\ 0, & \text{if } y_i < 0, \end{cases}$$

$$y_i = \sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}_i) + \epsilon_i,$$

$$\epsilon_i \sim N(0, 1),$$

are then as follows:

- Given the previous value of $\mathbf{y}$, draw a vector of $\mathbf{w} = (w_1, w_2, ...w_n)$ from the multivariate normal posterior

$$p(\mathbf{w}|D, T) = N(\mathbf{w}|\widehat{\mathbf{w}}, V_{\mathbf{w}}).$$

- Given the current value of $\mathbf{w}$, draw a new diagonal matrix $T = diag(\tau_1^2, \tau_2^2, ..., \tau_n^2)$ from the $n$ independent gamma posteriors

$$p(\tau_i^{-2}|w_i, \tau_0^2) = Ga\left(\tau_i^{-2}\left|\frac{k+1}{2}, \frac{k\tau_0^2 + w_i^2}{2}\right.\right).$$

- Given the current value of $T = diag(\tau_1^2, \tau_2^2, ..., \tau_n^2)$, draw $\tau_0^2$ from the gamma posterior

$$p(\tau_0^2|T) = Ga\left(\tau_0^2\left|\frac{kn}{2} + 1, \frac{k}{2}\sum_{i=1}^{n}\tau_i^{-2} + a_0\right.\right).$$

- Given the current value of $\mathbf{w}$, calculate the vector $\widehat{\mathbf{y}} = K^T\mathbf{w}$ with $i^{th}$ element $\widehat{y}_i = \mathbf{k}_i^T\mathbf{w}$. For $i = 1, 2, ..., n$, sample new latent variables $y_i$ from the independent truncated normal posteriors

$$y_i \sim \begin{cases} N(y_i|\widehat{y}_i, 1)I(y_i < 0), & \text{if } z_i = 0, \\ N(y_i|\widehat{y}_i, 1)I(y_i > 0), & \text{if } z_i = 1. \end{cases}$$

  This is directly, efficiently, performed as follows: let $\Phi(\cdot)$ be the standard normal distribution function and $P_i = \Phi(-\widehat{y}_i)$, and draw a sample $v_i$ independently from $U(0, 1)$. Then, caculate

$$y_i = \widehat{y}_i + \Phi^{-1}[z_iP_i + v_i(z_i + (1 - 2z_i)P_i)].$$

  In practical numerical computation, we should take special care of the situations where $P_i$ is close to zero when $z_i = 0$ and $P_i$ is close to one when $z_i = 1$.

We fit the model for the XOR example via MCMC taking $k = 2$ and $a_0 = 100$. The MCMC was run for 2,000 iteration after a burn-in of 1,000. Figure 3.8 shows the simple summary of predictive probabilities for the training data. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The data are labelled by case number and color coded – red for 1 and blue for zero. Figure 3.9 shows a similar summary for the test data. Taking $p_i = 0.5$ ($\widehat{y}_i = 0$) as the hard decision boundary, the error rate for the training data is 0 and the error rate for the test data is 5%, which is similar to that of SVMs.

### 3.3.9 Posterior predictive distribution

Now we apply our Bayesian kernel model to a new input $\mathbf{x}$, and wish to predict the outcome $y$. We first consider classical supervised learning problems. Suppose we only have labeled training data $D = (y_1, \mathbf{x}_1, y_2, \mathbf{x}_2, ..., y_n, \mathbf{x}_n)$ at the time of model training, and we do not have any information about new input $\mathbf{x}$. Thus the built model is based on labeled training data only. In the setting of normal regression, the posterior predictive distribution of $y$ is

$$P(y|D, \mathbf{x}) \propto \int N(y|\sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}), \sigma^2) p(\mathbf{w},\sigma^2|D) d\mathbf{w} d\sigma^2,$$

where $p(\mathbf{w},\sigma^2|D)$ is the joint posterior distribution for model parameters integrating out all of hyperparamters.

With the help of MCMC, the posterior samples of $y$ can be obtained easily. After drawing $(\mathbf{w},\sigma^2)$ from their joint posterior distribution in each iteration, we draw

$$y \sim N(y|\sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}), \sigma^2).$$

### 3.3.10 Learning with labeled and unlabeled data

Recently, there has been great interest in the use of unlabeled data for supervised learning, which is called semi-supervised learning (Bennett and Demiriz, 1998; Blum and Mitchell, 1998; Joachims, 1999; Szummer and Jaakkola, 2001; Zhu *et al.*, 2003). In this scenario, we not only have the labeled training data $D = (y_1, \mathbf{x}_1, y_2, \mathbf{x}_2, ..., y_n, \mathbf{x}_n)$, but also have the unlabeled input data $\mathbf{x}$ at the time of model training. In some real world problems, it is more difficult and expensive to collect fully labeled training data than unlabeled data. Thus, it is important to incorporate unlabeled data into the process of model training. Some studies have shown that the unlabeled input data $\mathbf{x}$ could be used to enhance the learning process and improve the prediction.

In the Bayesian framework, it is straightforward to fit the model with both labeled and unlabeled data by treating the prediction $y$ as a missing value. In Section 3.3.3, we have already derived the prediction function as

$$f_{n+1}(\mathbf{x}) = \sum_{j=1}^{n+1} w_j k(\mathbf{x}_j, \mathbf{x}).$$

We now discuss posterior predictive distribution of $y$ involving uncertainty over model parameters. Let $\mathbf{x}_{n+1} = \mathbf{x}$, and extend the vector $\mathbf{w}$ into a $n+1$ dimensional vector $\mathbf{w} = [w_1, w_2, ..., w_n, w_{n+1}]^T$. In the setting of normal regression, the posterior predictive distribution of $y$ is

$$P(y|D, \mathbf{x}) \propto \int N(y|\sum_{j=1}^{n+1} w_j k(\mathbf{x}_j, \mathbf{x}), \sigma^2) p(\mathbf{w},\sigma^2|D, \mathbf{x}) d\mathbf{w} d\sigma^2.$$

Note that the joint posterior distribution for model parameters $p(\mathbf{w},\sigma^2|D, \mathbf{x})$ is not only based on labeled data $D$, but also the unlabeled data $\mathbf{x}$. To obtain the posterior

samples of $y$, we draw the samples from its conditional posterior

$$y|\mathbf{w},\sigma^2,D,\mathbf{x} \sim N(y|\sum_{j=1}^{n+1} w_j k(\mathbf{x}_j,\mathbf{x}),\sigma^2).$$

Thus, given the initial value of $y$, the conditional posterior can be plugged into the original MCMC procedure.

### 3.3.11 MAP estimation via EM algorithm

In some cases where we are just interested in the posterior modes of parameters instead of the exact posterior inference, it is unnecessary to run a MCMC that requires thousands of iterations. For our Bayesian kernel models, the parameters of interests are the regression parameters $\mathbf{w}$, and other parameters are considered as latent variables. Thus the maximum a posteriori (MAP) estimate of $\mathbf{w}$ can be obtained by maximizing the marginal posterior $p(\mathbf{w}|D)$ with the help of EM algorithm. In this subsection, we will derive the EM algorithm for the binary probit regression model with above likelihood and prior specification.

To simplify the EM algorithm, we specify $\tau_0^2$ as a pre-selected constant instead of an unknown parameter with its own hyperprior. Thus we consider $\mathbf{w}$ as the parameters to be estimated and $T = diag(\tau_1^2, \tau_2^2, ..., \tau_n^2)$ and $\mathbf{y}$ as the latent variables or missing data. Let $\mathbf{w}^{old}$ be the current value, then the $Q$ function is:

$$\begin{aligned} Q(\mathbf{w}|\mathbf{w}^{old}) &= E_{old}(\log p(\mathbf{w}|T,\mathbf{y},D)) \\ &= \int (\log p(\mathbf{w}|T,\mathbf{y},D)) p(T,\mathbf{y}|\mathbf{w}^{old},D) dT d\mathbf{y} \\ &= \int (\log p(\mathbf{w},T,\mathbf{y}|D)) p(T|\mathbf{w}^{old},D) p(\mathbf{y}|\mathbf{w}^{old},D) dT d\mathbf{y}. \end{aligned}$$

Ignoring the terms that are unrelated to $\mathbf{w}$, the logarithm of the joint posterior

density $\log p(\mathbf{w},T,\mathbf{y}|D)$ can be written as

$$\log p(\mathbf{w},T,\mathbf{y}|D) = -\frac{1}{2}\left((\mathbf{y}-K^T\mathbf{w})^T(\mathbf{y}-K^T\mathbf{w}) + \mathbf{w}^{\mathbf{T}}T^{-1}\mathbf{w}\right) + \text{constant}$$

$$= -\frac{1}{2}(\mathbf{w}^T K K^T\mathbf{w} + \mathbf{w}^{\mathbf{T}}T^{-1}\mathbf{w} - \mathbf{y}^T K^T\mathbf{w} - \mathbf{w}^T K\mathbf{y}) + \text{constant}$$

For the E-step of the EM algorithm, we need to calculate the expectation of $\log p(\mathbf{w},T,\mathbf{y}|D)$ over $T,\mathbf{y}$ and conditional on the current value $\mathbf{w}^{old}$:

$$Q(\mathbf{w}|\mathbf{w}^{old}) = E_{old}(\log p(\mathbf{w}|T,\mathbf{y},D)) \tag{3.28}$$

$$= -\frac{1}{2}(\mathbf{w}^T K K^T\mathbf{w} + \mathbf{w}^{\mathbf{T}}E_{old}(T^{-1})\mathbf{w} - E_{old}(\mathbf{y})^T K^T\mathbf{w} - \mathbf{w}^T K E_{old}(\mathbf{y})) + \text{constant}$$

We must now calculate $E_{old}(T^{-1}) = diag(E_{old}(\tau_1^{-2}), E_{old}(\tau_2^{-2}), ..., E_{old}(\tau_n^{-2}))$ and $E_{old}(\mathbf{y}) = [E_{old}(y_1), E_{old}(y_2), ..., E_{old}(y_n)]$. Given the conditional posteriors for $\tau_i^{-2}$ as

$$p(\tau_i^{-2}|w_i) = Ga\left(\tau_i^{-2}\left|\frac{k+1}{2}, \frac{k\tau_0^2 + w_i^2}{2}\right.\right),$$

we can get the expectation of $E_{old}(\tau_i^{-2})$ as:

$$E_{old}(\tau_i^{-2}) = \frac{k+1}{k\tau_0^2 + (w_i^2)^{old}}.$$

The conditional posterior for the latent variables $y_i$ are independent truncated normals. Let $\widehat{y}_i = \mathbf{k}_i^T\mathbf{w}^{old}$, $p_i = N(\widehat{y}_i|0,1)$ and $P_i = \Phi(-\widehat{y}_i)$, the expectation $E_{old}(y_i)$ is

$$E_{old}(y_i) = \widehat{y}_i + (2z_i - 1)\frac{p_i}{z_i - (2z_i - 1)P_i}$$

Thus, the expectation $E_{old}(T^{-1})$ and $E_{old}(\mathbf{y})$ can be calculated analytically in the $Q$ function $Q(\mathbf{w}|\mathbf{w}^{old})$ for the following M-step.

For the M-step, we must now find the $\mathbf{w}$ that maximizes the the $Q$ function $Q(\mathbf{w}|\mathbf{w}^{old})$. For our Bayesian kernel models, the maximization is straightforward

given that fact that the $Q$ function (3.28) has the form of a logarithmic multivariate normal density. Thus, we have

$$\mathbf{w}^{new} = \left(E_{old}(T^{-1}) + KK^T\right)^{-1} KE_{old}(\mathbf{y}).$$

In summary, The EM algorithm can be described as follows:

1. Set a initial value for $\mathbf{y}$ and $T^{-1}$, then calculate the initial estimate of $\mathbf{w}^{(0)}$ as

$$\mathbf{w}^{(0)} = \left(T^{-1} + KK^T\right)^{-1} K\mathbf{y}.$$

2. For $t = 0, 1, .... :$

   (a) E-step: calculate the expectations

$$E_{old}(\tau_i^{-2}) = \frac{k+1}{k\tau_0^2 + (w_i^2)^{(t)}}.$$

   and

$$E_{old}(y_i) = \widehat{y}_i + (2z_i - 1)\frac{N(\widehat{y}_i|0,1)}{z_i - (2z_i - 1)\Phi(-\widehat{y}_i)}$$

$$\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}^{(t)}$$

   As we do in MCMC computation, we should take special care of the situations where $P_i$ is close to zero when $z_i = 0$ and $P_i$ is close to one when $z_i = 1$.

   (b) M-step: calculate the new estimate of $\mathbf{w}$ as

$$\mathbf{w}^{(t+1)} = \left(E_{old}(T^{-1}) + KK^T\right)^{-1} KE_{old}(\mathbf{y}).$$

   (c) Repeat step (a) and step (b) until the log-likelihood converges.

We fit the model for the XOR example again via EM algorithm taking $\tau_0^2 = 1$. Figure Figure 3.10 and Figure Figure 3.11 show the predictive probabilities for the training data and test data respectively. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The data are labelled by case number and color coded – red for 1 and blue for zero. Taking $p_i = 0.5$ ($\widehat{y}_i = 0$) as the hard decision boundary, the error rate for the training data is 0 and the error rate for the test data is 5%, which is similar to that of SVMs.

## 3.4 Orthogonalized kernel models

Essentially, the above kernel model is a regression model on the kernel matrix $K$. In practice, it is not a good modelling strategy to build the regression model on the kernel matrix $K$ directly, since there is possible multicollinearity in matrix $K$. For example, Figure 3.12 shows the kernel matrix for the XOR training dataset whose data points are reorganized to show the cluster structure of the kernel matrix. The kernel function is the Gaussian kernel (3.3) with $\rho^2 = 1$. The multicollinearity in the covariate matrix will cause serious correlation among parameter $w_1, w_2, ..., w_n$ and bad mixing in MCMC. Moreover, to calculate the conditional posterior distribution of $\mathbf{w}$ shown in (3.26), we have to invert a $n \times n$ matrix $V_{\mathbf{w}}$ per every iteration, which is computationally expensive for large $n$ in the scale of $O(n^3)$. To solve these problems, we use methods similar to those of Bayesian SVD regression (West, 2000; West *et al.*, 2002; Spang *et al.*, 2002) and build the regression model based on the reduced-rank approximation to the orthogonalized kernel matrix.

### 3.4.1 Singular value decomposition of kernel matrix

There are various orthogonalization methods, and different orthogonal bases can lead to different predictive distributions. For simplicity, we just use the singular value

decomposition. For a $n \times n$ symmetric matrix $K$, there exists an orthogonal matrix $F$ (Basilevsky, 1983) such that

$$K = F^T DF \tag{3.29}$$

where

- $D = diag(d_1, d_2, ..., d_n)$ is the diagonal matrix of positive singular values, ordered as $d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$; and
- $F$ is the $n \times n$ orthogonal matrix with $F^T F = FF^T = I$. We can consider $F$ as the factor matrix with the column $\mathbf{f}_i$ presenting the $n$-vector of $i^{th}$ kernel SVD factor across all $n$ samples.

Then our model can be build on the orthogonal SVD factor $\mathbf{f}_i$ instead of original kernel vector $\mathbf{k}_i = F^T D\mathbf{f}_i$. We can rewrite our model as

$$\begin{aligned} y_i &= \mathbf{k}_i^T \mathbf{w} + \epsilon_i, \\ &= \mathbf{f}_i^T DF\mathbf{w} + \epsilon_i, \\ &= \mathbf{f}_i^T \gamma + \epsilon_i, \end{aligned} \tag{3.30}$$

where $\gamma = DFw$ is the implied $n$-vector of regression parameter for SVD factors.

As discussed previously, the amount of computation needed is in the scale of $O(n^3)$ for the regression on $n$-dimensional predictor variables, which is infeasible for datasets with large sample sizes. To reduce the computational cost for large sample sizes problem, we use the reduced rank approximation to the $n \times n$ kernel matrix $K$. For a small number $m \leq n$, let $D_{m\times m} = diag(d_1, d_2, ..., d_m)$, and $F_{m\times n}$ be the first $m$ rows of $F$. We can then approximate the kernel matrix $K$ with

$$K_{n\times n} = F_{m\times n}^T D_{m\times m} F_{m\times n}.$$

Let SVD factor $\mathbf{f}_i$ be the $i^{th}$ column of the factor matrix $F_{m\times n}$, which is a $m$-vector. Then the amount of computation for our regression model (3.30) can be reduced to

the scale of $O(m^3)$. Note that we can use a much more efficient algorithm (Press and others, 1992) to calculate the first $m$ principle factors than calculating SVD of $n \times n$ matrix $K$. To simplify presentation, we will still use the notation of SVD and let $D$ and $F$ represent $D_{m\times m}$ and $F_{m\times n}$.

In practice, we find that we can approximate the $K$ setting $m \ll n$ without significant decrease in the accuracy of the solution. For example, in the kernel matrix of XOR training data, the SVD factors with several largest eigenvalues can account for most of the linear dependence structure. The upper frame of Figure 3.13 provides display of elements $d_i^2$ as a percentage of total $\sum_{i=1}^{n} d_i^2$, the lower frame show the percentage of variance explained by the largest $k$ factors, i.e., $(\sum_{i=1}^{k} d_i^2)/(\sum_{i=1}^{n} d_i^2)$. In our model fitting procedures, we choose the SVD factors with 10 largest eigenvalues that have already explained 99.5% of linear structure.

### 3.4.2 Likelihood and priors

Given the observed value of $\mathbf{y}$ and $X$, the likelihood for $m$-dimensional regression parameter $\gamma$ is

$$
\begin{aligned}
p(\mathbf{y}|X,\gamma) &= \exp\left\{-\frac{1}{2}(\gamma-\widehat{\gamma})^T(\frac{1}{\sigma^2}FF^T)(\gamma-\widehat{\gamma})\right\}, \qquad (3.31)\\
&= \exp\left\{-\frac{1}{2\sigma^2}(\gamma-\widehat{\gamma})^T(\gamma-\widehat{\gamma})\right\},\\
&= \prod_{i=1}^{m}\exp\left\{-\frac{(\gamma_i-\widehat{\gamma}_i)^2}{2\sigma^2}\right\},
\end{aligned}
$$

where $\widehat{\gamma}_i$ is the the $i^{th}$ element of the least-squares vector

$$\widehat{\gamma} = F\mathbf{y}.$$

With the inherent orthogonality of SVD factors, the conjugate priors for $\gamma$ are

independent normal priors

$$p(\gamma|X,T)=\prod_{i=1}^{m} N(\gamma_i|0,\sigma^2\tau_i^2), \tag{3.32}$$

$$= N(\gamma|\mathbf{0},\sigma^2 T),$$

where $\tau_i$ are the prior scale parameters with $T = diag(\tau_1^2, \tau_2^2, ..., \tau_m^2)$. We use the individual scale parameter $\tau_i$ for each regression parameter $\gamma_i$, which allow for major differences among $\gamma_i$, because there are possibly substantial variations in the effects of factors as discussed in West (2000), West *et al.* (2002) and Spang *et al.* (2002). Note that the independent priors for $\gamma_i$ implies a singular normal prior for the regression parameters $\mathbf{w}$ on the original kernel vector $\mathbf{k}_i$

$$p(\mathbf{w}|X,T) = N(\mathbf{w}|0,\Sigma_{\mathbf{w}})$$

where the singular precision matrix is

$$\Sigma_{\mathbf{w}}^{-1} = F^T D T^{-1} D F$$

This class of structured prior is call generalized $g$-priors (West, 2000; West, 2003) because it extends and generalizes the class of standard $g$-priors (Zellner, 1986) whose precision matrix

$$\Sigma_{\mathbf{w}}^{-1} = g F^T D D F,$$

where $g$ is a constant. The motivation behind generalized $g$-priors is that the priors shares the structure of likelihood function with additional scale parameter $\tau_i$ that allow different scaling in each of the orthogonal factor directions.

The specification of priors on the rest of parameters are similar to those in section 3.4. The prior for $\tau_i^2$ is an independent inverse gamma prior that is suggested by conditional conjugacy, i.e.,

$$\tau_i^{-2} \sim Ga\left(\tau_i^{-2} \middle| \frac{k}{2}, \frac{k}{2}\tau_0^2\right)$$

with shape parameter $k/2$ and scale parameter $k\tau_0^2/2$. The $\tau_0^2$ can be a pre-selected constant or a hyperparameter with its own hyperprior

$$\tau_0^2 \sim Exp(\tau_0^2|a_0),$$

where $a_0$ is the quantity to control overall scale of sparsity. A large $a_0$ allows that the posteriors probability of some of regression parameters $\gamma_i$ is concentrated at zero, which implies that the associated factors play little roles in the regression.

In the setting of normal linear regression, the conjugate prior distribution for $\sigma^2$ is inverse gamma distribution as

$$p(\sigma^{-2}) = Ga\left(\sigma^{-2}\left|\frac{n_0}{2}, \frac{n_0}{2}\sigma_0^2\right.\right).$$

### 3.4.3 Posteriors

Given the above likelihood and priors, we can get the full conditional posteriors that is very similar to those of original Bayesian kernel models in subsection 3.4.4. In fact, the calculation is simpler due to the the inherent orthogonality of SVD factors. Under the specified prior (3.32) and likelihood function (3.31), the resulting conditional posteriors for $\gamma$ is a simple $m$ independent normal posteriors

$$\begin{aligned} p(\gamma|\mathbf{y},X,T) &= N(\gamma|\gamma^*,\sigma^2 G), \\ &= \prod_{i=1}^{m} N(\gamma_i|\gamma_i^*, \sigma^2 g_i), \end{aligned}$$

where $\gamma_i^*$ is the $i^{th}$ element of the posterior mean vector

$$\begin{aligned} \gamma^* &= \widehat{\gamma}\frac{\tau_i^2}{1+\tau_i^2}, \\ &= F\mathbf{y}\frac{\tau_i^2}{1+\tau_i^2}, \end{aligned}$$

and $G = diag(g_1, g_2, ..., g_m)$ with

$$g_i = \frac{\tau_i^2}{1+\tau_i^2}.$$

Note that the implied conditional posteriors for the regression parameters $\mathbf{w}$ on the original kernel vector $\mathbf{k}_i$ is a singular normal posterior that is not obviously uniquely defined due to the singularity structure. As shown in West (2000), however, this seeming identification problem is solved under the specification of fixed prior mean, here of 0 for $\gamma$.

The conditional posterior for the rest of parameters are similar to those in Subsection 3.4.4. The marginal conditional posterior for $\sigma^2$ over $\gamma$ is an inverse gamma as

$$p(\sigma^{-2}|D,T) = Ga\left(\sigma^{-2}\left|\frac{n_0+n}{2}, \frac{n_0\sigma_0^2+q}{2}\right.\right),$$

where

$$q = (\mathbf{y}-K^T\gamma^*)^T(\mathbf{y}-K^T\gamma^*).$$

Thus it is straightforward to draw samples from the joint conditional posterior distribution $p(\gamma,\sigma^{-2}|D,T)$ by drawing $\sigma^{-2}$ from its posterior marginalised over $\gamma$, i.e., $p(\sigma^{-2}|D,T)$, followed by a drawing of $\gamma$ given $\sigma^{-2}$ from $p(\gamma|D,T,\sigma^2)$.

Under the specified prior (3.23), the conditional posteriors for scale parameter $\tau_i^2$ is an independent inverse gamma posterior

$$p(\tau_i^{-2}|\gamma_i,\tau_0^2,D) = Ga\left(\tau_i^{-2}\left|\frac{k+1}{2}, \frac{k\tau_0^2+\gamma_i^2}{2}\right.\right).$$

Under the specified prior (3.24), the conditional posteriors for hyperparameter $\tau_0^2$ is a gamma posterior

$$p(\tau_0^2|T,D) = Ga\left(\tau_0^2\left|\frac{km}{2}+1, \frac{k}{2}\sum_{i=1}^{n}\tau_i^{-2}+a_0\right.\right).$$

### 3.4.4 Model fitting via MCMC

One of the key advantages of orthogonalized kernel models over original kernel models is that MCMC based computation may be carried out in $m$-dimensional regression parameter space for $\gamma$ rather than possibly very large $n$-dimensional regression parameter space for $\mathbf{w}$ in large sample sizes problem. Moreover, the conditional posteriors for $\gamma$ and $\tau_i^2$ are independent posteriors due to the inherent orthogonality of SVD factors, which can provide much better mixing and convergence for MCMC. It is straightforward to implement the posterior simulation using standard Gibbs sampling that is illustrated in detail below. The initial values are chosen arbitrarily for each of parameters, and then the samples of parameters are drawn from their conditional posterior distributions at each iteration. The components of each MCMC step for the regression model

$$y_i = \mathbf{f}_i^T \gamma + \epsilon_i,$$
$$\epsilon_i \sim N(0, \sigma^2),$$

are as follows:

- Draw $\sigma^{-2}$ from its marginal over $\gamma$ posterior

$$p(\sigma^{-2}|D,T) = Ga\left(\sigma^{-2}\left|\frac{n_0+n}{2}, \frac{n_0\sigma_0^2+q}{2}\right.\right).$$

- Given the current value of $\sigma^{-2}$, draw a vector of $\gamma = (\gamma_1, \gamma_2, ...\gamma_m)$ from the $m$ independent normal posteriors

$$p(\gamma_i|\mathbf{y},X,T) = N(\gamma_i|\gamma_i^*, \sigma^2 g_i).$$

- Given the current value of $\gamma$, draw a new diagonal matrix $T = diag(\tau_1^2, \tau_2^2, ..., \tau_m^2)$ from the $m$ independent gamma posteriors

$$p(\tau_i^{-2}|\gamma_i, \tau_0^2) = Ga\left(\tau_i^{-2}\left|\frac{k+1}{2}, \frac{\tau_0^2 k+\gamma_i^2}{2}\right.\right).$$

- Given the current value of $T = diag(\tau_1^2, \tau_2^2, ..., \tau_m^2)$, draw $\tau_0^2$ from the gamma posterior

$$p(\tau_0^2|T) = Ga\left(\tau_0^2 \left| \frac{km}{2} + 1, \frac{k}{2}\sum_{i=1}^{n}\tau_i^{-2} + a_0 \right.\right).$$

For a binary classification model with binary output variable $z_i \in \{0, 1\}$, it is straightforward to modify the above MCMC algorithm using probit regression where $\mathbf{y} = (y_1, y_2, ..., y_n)^T$ are the latent variables. MCMC is run to generate the approximate samples from the joint posteriors $p(\mathbf{y}, \gamma, T|\mathbf{z}, X)$. The components of each MCMC steps for the probit regression model

$$z_i = \begin{cases} 1, & \text{if } y_i \geq 0, \\ 0, & \text{if } y_i < 0, \end{cases}$$

$$y_i = \sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}_i) + \epsilon_i,$$

$$\epsilon_i \sim N(0, 1),$$

are as follows:

- Given the previous values of $\mathbf{y}$, draw a vector of $\gamma = (\gamma_1, \gamma_2, ...\gamma_m)$ from the $m$ independent normal posteriors

$$p(\gamma_i|\mathbf{y}, X, T) = N(\gamma_i|\gamma_i^*, g_i).$$

- Given the current value of $\gamma$, draw a new diagonal matrix $T = diag(\tau_1^2, \tau_2^2, ..., \tau_m^2)$ from the $m$ independent gamma posteriors

$$p(\tau_i^{-2}|\gamma_i, \tau_0^2) = Ga\left(\tau_i^{-2} \left| \frac{k+1}{2}, \frac{\tau_0^2 k + \gamma_i^2}{2} \right.\right).$$

- Given the current value of $T = diag(\tau_1^2, \tau_2^2, ..., \tau_m^2)$, draw $\tau_0^2$ from the gamma posterior

$$p(\tau_0^2|T) = Ga\left(\tau_0^2 \left| \frac{km}{2} + 1, \frac{k}{2}\sum_{i=1}^{m}\tau_i^{-2} + a_0 \right.\right).$$

- Given the current value of $\gamma$, calculate the vector $\widehat{\mathbf{y}} = \mathbf{F}^T\gamma$ with $i^{th}$ element $\widehat{y}_i = \mathbf{f}_i^T\gamma$. For $i = 1, 2, ..., n$, sample new vlaue of latent variables $y_i$ from independent truncated normal posteriors

$$y_i \sim \begin{cases} N(y_i|\widehat{y}_i, 1)I(y_i < 0), & \text{if } z_i = 0, \\ N(y_i|\widehat{y}_i, 1)I(y_i > 0), & \text{if } z_i = 1. \end{cases}$$

  In practice, let $\Phi(\cdot)$ be the standard normal distribution function and $P_i = \Phi(-\widehat{y}_i)$; draw a sample $v_i$ independently from $U(0, 1)$, and calculate

$$y_i = \widehat{y}_i + \Phi^{-1}[z_i P_i + v_i(z_i + (1 - 2z_i)P_i)].$$

We fit the model for the XOR example via the MCMC taking $m = 10$, $k = 2$ and $a_0 = 100$. The MCMC was run for 2,000 iteration after a burn-in of 1,000. Figure 3.14 shows the simple summary of predictive probabilities for the training data. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{f}_i^T\gamma$. The data are labelled by case number and color coded – red for 1 and blue for zero. Figure 3.15 shows a similar summary for the test data. Taking $p_i = 0.5$ ($\widehat{y}_i = 0$) as the hard decision boundary, the error rate for the training data is 0 and the error rate for the test data is 1%, which is better than the Bayesian kernel model built on the kernel matrix directly.

### 3.4.5 Calculation of Bayes' factors

One of major concerns in the orthogonalized kernel models is the selection of $m$, the number of principal components based on the $m$ largest eigenvalues. In general, we wish to select a small number to reduce the computational cost and alleviate the over fitting. However, we may risk excluding some important components that have small eigenvalues but are highly correlated with the responses. In XOR example, we run

the above MCMC analysis when $m = 10$, and plot the posterior mean of $\gamma$ in upper frame of Figure 3.16. We can clearly see that the $4th$ principal component is most important for regression though its eigenvalue is much smaller that first 3 principal components that have already explained majority of linear structure. In practice, it will be useful to compare the different models with different $m$ and choose the best model. In the Bayesian framework, we can calculate the Bayes factor

$$BF = \frac{p(\mathbf{y}|M_i)}{p(\mathbf{y}|M_j)}$$

to compare the two competing models $M_i$ and $M_j$. Moreover, if we have a collection of candidate models $\{M_1, M_2, ..., M_q\}$, we can calculate the posterior probability of model $M_i$

$$p(M_i|\mathbf{y}) = \frac{p(\mathbf{y}|M_i)p(M_i)}{\sum_{j=1}^{q} p(\mathbf{y}|M_j)p(M_j)},$$

then we can use model averaging to make the prediction or calculate other quantity of interest. As we can see, the key here is to calculate the marginal distribution of $\mathbf{y}$

$$p(\mathbf{y}|M_i) = \int p(\mathbf{y}|\theta, M_i)p(\theta|M_i)d\theta.$$

Though it is not a easy problem generally, it is straightforward to estimate the marginal density in our orthogonalized kernel models since we fit the model via Gibbs sampling given the close-form full conditional distributions. In the following, we will apply the idea in Chib (1995) to the binary probit regression model. To simply our illustration, we suppress the model indicator $M_i$ in our notation to estimate the the marginal density $p(\mathbf{z})$.

First, we consider the probit regression model with $\tau_0^2$ fixed. By Bayes' rule, we have

$$p(\mathbf{z}) = \frac{p(\mathbf{z}|\gamma,T)p(\gamma,T)}{p(\gamma,T|\mathbf{z})}.$$

Since the identity holds true for any value of ($\gamma$,$T$), we can just choose one single point of ($\gamma$,$T$) from its posterior estimate, e.g., the posterior mean $(\widehat{\gamma}, \widehat{T})$, for the calculation. To improve computational stability, we calculate it in the log scale

$$\log p(\mathbf{z}) = \log p(\mathbf{z}|\widehat{\gamma}, \widehat{T}) + \log p(\widehat{\gamma}, \widehat{T}) - \log p(\widehat{\gamma}, \widehat{T}|\mathbf{z}).$$

The first two terms are available explicitly. We now show how to estimate $p(\widehat{\gamma}, \widehat{T}|\mathbf{z})$ using the output of Gibbs sampling. Writing

$$\log p(\widehat{\gamma}, \widehat{T}|\mathbf{z}) = \log p(\widehat{T}|\widehat{\gamma}, \mathbf{z}) + \log p(\widehat{\gamma}|\mathbf{z}),$$

we can get $p(\widehat{T}|\widehat{\gamma}, \mathbf{z})$ from the conditional posterior density for $T$. As for $p(\widehat{\gamma}|\mathbf{z})$, we can estimate it using the Rao-Blackwellized mixture estimate as

$$\begin{aligned} p(\widehat{\gamma}|\mathbf{z}) &= \int p(\widehat{\gamma}|T, \mathbf{y}, \mathbf{z}) p(T, \mathbf{y}|\mathbf{z}) dT d\mathbf{y} \\ &\approx \frac{1}{I} \sum_{i=1}^{I} p(\widehat{\gamma}|T^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}), \end{aligned}$$

where $T^{(i)}, \mathbf{y}^{(i)}$ are the posterior samples from the output of Gibbs sampling. Thus our estimate of $p(\mathbf{z})$ in log scale is

$$\log p(\mathbf{z}) = \log p(\mathbf{z}|\widehat{\gamma}, \widehat{T}) + \log p(\widehat{\gamma}, \widehat{T}) - \log p(\widehat{T}|\widehat{\gamma}, \mathbf{z}) - \left( \frac{1}{I} \sum_{i=1}^{I} p(\widehat{\gamma}|T^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}) \right).$$

Then the marginal density and the corresponding Bayes factor can be calculated after the exponential transformation.

For XOR example, we run the Gibbs sampler and estimate the $p(\mathbf{z})$ for 10 different models with $m = 1, 2, ..., 10$. The lower frame of Figure 3.16 plots the values of $\log p(\mathbf{z})$. It shows a significant increase in $m = 4$, which implies the importance of $4th$ principal component in the regression model.

If $\tau_0^2$ has its own hyperprior (3.23), the computation is a little complicated but still feasible. By Bayes' rule, we have

$$p(\mathbf{z}) = \frac{p(\mathbf{z}|\widehat{\gamma},\widehat{\tau}_0^2,\widehat{T})p(\widehat{\gamma},\widehat{\tau}_0^2,\widehat{T})}{p(\widehat{\gamma},\widehat{\tau}_0^2,\widehat{T}|\mathbf{z})}.$$

where

$$p(\widehat{\gamma},\widehat{\tau}_0^2,\widehat{T}|\mathbf{z}) = p(\widehat{\tau}_0^2|\widehat{\gamma},\widehat{T},\mathbf{z})p(\widehat{T}|\widehat{\gamma},\mathbf{z})p(\widehat{\gamma}|\mathbf{z}).$$

The first term $p(\widehat{\tau}_0^2|\widehat{\gamma},\widehat{T},\mathbf{z})$ is available explicitly from the conditional posterior density for $\widehat{\tau}_0^2$, and the third term $p(\widehat{\gamma}|\mathbf{z})$ can be estimated by the Rao-Blackwellized mixture of $p(\widehat{\gamma}|T^{(i)},\widehat{\tau}_0^{2(i)},\mathbf{y}^{(i)},\mathbf{z})$ as discussed above. As for the second term $p(\widehat{T}|\widehat{\gamma},\mathbf{z})$, we have

$$p(\widehat{T}|\widehat{\gamma},\mathbf{z}) = \int p(\widehat{T}|\tau_0^2,\widehat{\gamma},\mathbf{z})p(\tau_0^2|\widehat{\gamma},\mathbf{z})d\tau_0^2$$

$$\approx \frac{1}{I}\sum_{i=1}^{I} p(\widehat{T}|\widehat{\tau}_0^{2(i)},\widehat{\gamma},\mathbf{z}),$$

where $\widehat{\tau}_0^{2(i)} \sim p(\tau_0^2|\widehat{\gamma},\mathbf{z})$ which is not available from existing output of Gibbs sampling. To get these samples, we have to run the Gibbs sampling for an additional $I$ iterations with only two conditionals , i.e.,

$$p(\tau_0^2|T,\widehat{\gamma},\mathbf{z}) \text{ and } p(T|\tau_0^2,\widehat{\gamma},\mathbf{z}).$$

Thus, with additional sampling required, the final estimate of $p(\mathbf{z})$ can be calculated.

### 3.4.6 MAP estimation via EM algorithm

The EM algorithm for MAP estimation of $\gamma$ is very similar to the that of $\mathbf{w}$ in Subsection 3.4.6. In summary, the EM algorithm can be described as follows:

1. Set a initial value for $\mathbf{y}$ and $T^{-1}$, then calculate the initial estimate of $\gamma^{(0)}$ as

$$\gamma^{(0)} = \left(T^{-1} + I\right)^{-1} F\mathbf{y}.$$

2. For $t = 0, 1, .... :$

   (a) E-step: calculate the expectations

$$E_{old}(\tau_i^{-2}) = \frac{k+1}{k\tau_0^2 + (\gamma_i^2)^{(t)}},$$

   and

$$E_{old}(y_i) = \widehat{y}_i + (2z_i - 1)\frac{N(\widehat{y}_i|0,1)}{z_i - (2z_i - 1)\Phi(-\widehat{y}_i)},$$

   where

$$\widehat{y}_i = \mathbf{f}_i^T \gamma^{(t)}.$$

   (b) M-step: calculate the new estimate of $\gamma$ as

$$\gamma^{(t+1)} = \left(E_{old}(T^{-1}) + I\right)^{-1} F E_{old}(\mathbf{y}).$$

   (c) Repeat step (a) and step (b) until the log-likelihood converges.

We fit the model for the XOR example again via EM algorithm. Figure 3.17 and Figure 3.18 show the predictive probabilities for the training data and test data respectively. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{f}_i^T\gamma$. The data are labelled by case number and color coded – red for 1 and blue for zero. Taking $p_i = 0.5$ ($\widehat{y}_i = 0$) as the hard decision boundary, the error rate for the training data is 0 and the error rate for the test data is 1%, which is similar to that of MCMC.

### 3.4.7 Interpretation as kernel principal component analysis

Principal component analysis (PCA) (Jolliffe, 1986; Jackson, 1991) is an orthogonal basis transformation of coordinate system. The new coordinate values are called principal components. PCA can be performed by solving an eigenvalue system. Start with standard SVD of $p \times n$ data matrix $X$ (Golub and Van Loan, 1989), we have

$$X = ABF$$

or

$$\mathbf{x}_i = AB\mathbf{f}_i, \text{ for } i = 1, 2, ..., n$$

where

- $A$ is the $p \times n$ SVD loading matrix with orthonomal columns so that $A^T A = I$;
- $B = diag(b_1, b_2, ..., b_n)$ is the diagonal matrix of positive singular values, ordered as $b_1 \geq b_2 \geq \cdots \geq b_n \geq 0$; and
- $F$ is the $n \times n$ orthogonal matrix with $F^T F = FF^T = I$. The $i^{th}$ column of $F$ is denoted as $i^{th}$ factor $\mathbf{f}_i$ that is a linear combination of original $p$ predictor variables.

It is often found that a small number of factors are sufficient to represent most of structure of data and PCA is widely used to reduce the dimensionality or extract the structure from high-dimensional datasets. However, linear PCA cannot always detect all important structure in the input space. It is possible that there exists nonlinear structure in the data, but linear PCA cannot detect it since it is only a linear combination of original $p$ predictor variables. Figure 3.19 show a simple illustrative example that clearly show that the linear PCA cannot capture the nonlinear structure

in the data. To solve this problem, we can transform the input space $R^p$ into the feature space $F$ by a nonlinear mapping

$$\phi : R^p \rightarrow F,$$

then perform linear PCA on the feature space

$$\phi(\mathbf{x}_i) = AB\mathbf{f}_i,\ i = 1, 2, ..., n. \tag{3.33}$$

Hence, the factor $\mathbf{f}_i$ is a combination of nonlinear transformation of original $p$ predictor variables, which could hopefully extract interesting nonlinear structures in the data.

As discussed previously, the feature space $F$ could have a very high, or even infinite, dimensionality. So it is infeasible to perform PCA on such feature space directly. Instead, the idea of kernel expansion could be applied to solve this problem. In the following, we will show how the orthogonalized kernel model can be connected the PCA on the feature space $F$.

The kernel matrix $K$ can be decomposed as following

$$K = F^T DF.$$

Or the $i^{th}$ and $j^{th}$ element of kernel matrix $K$, $k_{ij}$, can be decomposed as

$$k_{ij} = \mathbf{f}_i^T D\mathbf{f}_j$$

As we discussed it in the previous chapter, $k_{ij}$ can be interpreted as the dot product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space

$$\begin{aligned} k_{ij} &= k(\mathbf{x}_i, \mathbf{x}_j), \\ &= \phi(\mathbf{x}_i^T)\phi(\mathbf{x}_j) \end{aligned}$$

where $\phi(\cdot)$ is the implicit nonlinear transformation that may or may not be known. Let $D = B^T B$ and plug equation (3.33) into the kernel matrix, we have

$$\begin{aligned} k_{ij} &= k(\mathbf{x}_i, \mathbf{x}_j), \\ &= \phi(\mathbf{x}_i^T)\phi(\mathbf{x}_j), \\ &= \mathbf{f}_i^T B^T A^T AB\mathbf{f}_j, \\ &= \mathbf{f}_i^T D\mathbf{f}_j, \end{aligned}$$

which is exactly the SVD of kernel matrix $K$. Thus the factor $\mathbf{f}_i$ derived by the decomposition of kernel matrix can be interpreted as the PCA factor in high-dimensional feature space. Though $\phi$ could be a nonlinear map into possibly high, or infinite, dimensional space $F$, we do not need the actual function and mapped patterns explicitly. Instead, all necessary calculations are done by using kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ in the original input space. This method is called kernel PCA and more detailed description can be found in Schoelkopf *et al.* (1997). Figure 3.20 shows a simple example of the kernel PCA that is equivalent to the linear PCA in the nonlinear feature space by polynomial transformation.

With this interpretation, our orthogonalized kernel regression model

$$\begin{aligned} y_i &= \mathbf{k}_i^T \mathbf{w} + \epsilon_i, \\ &= \mathbf{f}_i^T \gamma + \epsilon_i, \end{aligned}$$

can be interpreted as an ordinary linear regression model on factors that are extracted by nonlinear PCA, i.e.,

$$\begin{aligned} \phi(\mathbf{x}_i) &= AB\mathbf{f}_i, \\ y_i &= \mathbf{f}_i^T \gamma + \epsilon_i. \end{aligned}$$

Note that this interpretation is different from the Bayesian factor kernel model that will be discussed in Chapter 4.

## 3.5 Inference on kernel parameters

All computations above are based on the assumption that the parameters in the kernel $K$ are known or fixed, as is common practice with support vector machines and other kernel models in the frequentist framework. It will be much more useful if the kernel parameters can be adjusted adaptively by the model fitting algorithms. Though it is a difficult task in SVMs, it is relatively easy and straightforward to be implemented in our Bayesian kernel models with the help of MCMC. In this section, we will introduce a new MCMC algorithm for the inference on kernel parameters.

We now consider the Gaussian kernel function

$$k(\mathbf{x}_j, \mathbf{x}_i) = \exp\left(-\sum_{l=1}^{p} \rho_l^2 (x_{li} - x_{lj})^2\right),$$

where $\rho_l^2$ is an individual scale parameter for each dimension $l = 1, 2, ..., p$. This class of kernel functions is closely related to the principle of automatic relevance determination (Neal and Hinton, 1995). The 'relevance' of each predictor variable $X_l$ is determined by the corresponding scale parameter $\rho_l^2$. For example, if $\rho_l^2$ is close to zero, then the corresponding predictor variable $X_l$ will have little effects on the kernel function. The effect of scale parameters on $f(\mathbf{x})$ in two-dimensional space is shown in Figure 3.21. On the left side $\rho_1^2 = \rho_2^2 = 20$ while on the right $\rho_1^2 = 1$ and $\rho_2^2 = 20$.

### 3.5.1 Fully Bayesian inference

Note that we use $\rho_l^2$ in the kernel function to make sure that it is non-negative. However, to simplify the computation, we make all the inference on the $\rho_l$, assuming $\rho_l > 0$, instead of $\rho_l^2$. We now specify the hierarchical priors for the kernel parameters $\rho = (\rho_1, \rho_2, ...\rho_p)^T$ in a manner similar to that for the regression coefficients.

Specifically, the kernel parameter $\rho_l$ has the truncated normal prior

$$\rho_l|g_l^2 \sim N(\rho_l|0, g_l^2)I(\rho_l > 0), \tag{3.34}$$

or equivalently,

$$\rho_l^2|g_l^2 \sim Ga\left(\rho_l^2\left|\frac{1}{2}, \frac{1}{2g_l^2}\right.\right),$$

where $g_l^2$ are the prior scale parameters with $G = diag(g_1^2, g_2^2, ..., g_p^2)$. The $\rho_l$ are assumed independent. Note that we use individual scale parameter $g_l^2$ for each kernel parameters $\rho_l$, which allow for major differences among $\rho_l$, because there are possibly substantial variations in the effects of each predictor dimensions. The prior for $g_l^2$ is an independent inverse gamma prior that is suggested by conditional conjugacy, i.e.,

$$g_l^{-2} \sim Ga\left(g_l^{-2}\left|\frac{k}{2}, \frac{k}{2}g_0^2\right.\right)$$

with shape parameter $k/2$ and scale parameter $kg_0^2/2$. As usual, we can specify that $\rho_l$ has a prior concentrated near zero to reflect our a priori belief that many predictors have small or little effects on the kernel function in the case where there are many irrelevant predictors.

Under the above priors and likelihood function for normal regression models, the conditional posterior distribution for $\rho$ is in the form of:

$$\begin{aligned}
p(\rho|D, \mathbf{w}, G) &\propto p(D|\mathbf{w}, \rho, G)p(\rho|G) \\
&= \prod_{i=1}^{n} N\left(y_i\left|\sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}_i)\right., \sigma^2\right) \prod_{l=1}^{p} p(\rho_l) \\
&= \prod_{i=1}^{n} N\left(y_i\left|\sum_{j=1}^{n} w_j \exp\left(-\sum_{l=1}^{p} \rho_l^2 (x_{li} - x_{lj})^2\right)\right., \sigma^2\right) \prod_{l=1}^{p} N(\rho_l|0, g_l^2)I(\rho_l > 0).
\end{aligned}$$

The conditional posterior distribution for $g_l^2$ is an independent inverse gamma pos-

terior

$$p(g_l^{-2}|\rho_l^2, D) = Ga\left(g_l^{-2}\left|\frac{k+1}{2}, \frac{kg_0^2+\rho_l^2}{2}\right.\right) \tag{3.35}$$

independently over $l$.

Due to the difficulty of direct sampling from the above conditional posterior distribution for $\rho_l$, we use the Metropolis algorithm to draw the samples of $\rho_l$. The proposal distribution can be a simple random walk. For example, we might use a proposal distribution such as

$$J(\rho_l^*|\rho_l) \sim N(\rho_l^*|\rho_l, \sigma_\rho^2 g_l^2),$$

or, equivalently,

$$J(\rho^*|\rho) \sim N(\rho^*|\rho, \sigma_\rho^2 G),$$

where $\sigma_\rho^2$ can be a pre-specified value or be adjusted adaptively during tuning steps to meet a range of accept/reject rate. Note that the proposal distribution here depends on the individual scale parameter $g_l^2$ that allows that each $\rho_l$ to have the individual "jumping" scales.

In summary, the MCMC steps for kernel parameters are as follows:

- Given the current value of $\mathbf{w}$, $G$ and $\rho$, to sample a new value of $\rho^{new}$,
  - Sample a candidate point $\rho^*$ from the proposal distribution

$$J(\rho^*|\rho) \sim N(\rho^*|\rho, \sigma_\rho^2 G).$$

  - Calculate the ratio of conditional posterior densites

$$r = \frac{p(\rho^*|D, \mathbf{w}, G)}{p(\rho|D, \mathbf{w}, G)}.$$

– if $r \geq 1$, set $\rho^{new} = \rho^*$;

if $r \leq 1$, set $\rho^{new} = \begin{cases} \rho^*, & \text{with probability } r; \\ \rho, & \text{with probability } 1-r. \end{cases}$

- Given the new value of $\rho^{new}$, draw a new diagonal matrix $G = diag(g_1^2, g_2^2, ..., g_p^2)$ from the $p$ independent inverse gamma posteriors

$$p(g_l^{-2}|\rho_l^2, D) = Ga\left(g_l^{-2} \middle| \frac{k+1}{2}, \frac{kg_0^2 + \rho_l^2}{2}\right).$$

Then those two components can be plugged into the MCMC procedure in Subsection 3.4.5 and Subsection 3.5.4 to construct the MCMC over the full parameter space. Note that this MCMC algorithm is independent of the form of kernel functions, so it can be easily applied to various kernel functions.

### 3.5.2 A discrete model for kernel parameters

Though the above Metropolis algorithm provides a possible way to make fully inference about $\rho$, we found that it tends to converge very slowly when $p$ is large. In practice, we found that the predictive performance of Bayesian kernel models is insensitive to the value of $\rho_l$ when $X_l$ is relevant, and it is not quite necessary to get the accurate estimate of $\rho_l$. For example, for 2 dimensional XOR data, the Bayesian kernel model with $\rho = (1, 1)$ and $\rho = (0.5, 0.5)$ gives very similar results. However, in the situations where many predictors are irrelevant, the performance of Bayesian kernel models is very sensitive to the exclusion/inclusion of the right predictors. In such situations, it is more important for us to know if $\rho_l^2$ is zero or nonzero than the exact inference about $\rho$. We now introduce an empirical discrete updating algorithm via Gibbs sampling, which converges faster than the Metropolis algorithm without significant decrease in the predictive accuracy in many practical cases. We do this using alternative priors for the $\rho_l$, as follows.

Assume that $\rho_l$ only has a set of $m$ discrete values that are pre-selected, $\rho_l \in \{h_1, h_2, ..., h_k, ..., h_m\}$. Specify the prior for $\rho_l$ as a discrete distribution on those points, namely

$$\rho_l \sim \sum_{k=1}^{m} \pi_k \delta(\rho_l - h_k)$$

where $\delta(x)$ is the point mass function at zero. The prior point mass $\pi_i = p(\rho_l = h_i)$ can be specified to be proportional to the continuous prior density (3.34) to introduce the sparsity of $\rho$, as

$$\pi_k \propto g_l^{-1} \exp\left(-\frac{h_k^2}{2g_l^2}\right).$$

Let us consider the situation of choosing the value of $\rho_l$ in one specific dimension given other kernel parameters $\rho_{-l} = \{\rho_1, \rho_2, ..., \rho_{l-1}, \rho_{l+1}, ..., \rho_p\}$ in the Gaussian kernel function. We define

$$k^{(-l)}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{u \neq l} \rho_u^2 (x_{ui} - x_{uj})^2\right),$$

so that the full kernel can be written as :

$$k(\mathbf{x}_j, \mathbf{x}_i) = k^{(-l)}(\mathbf{x}_i, \mathbf{x}_j) \exp\left(-\rho_l^2 (x_{li} - x_{lj})^2\right).$$

In the setting of normal regression, we have the likelihood function

$$p(y_i | \rho_l = h_k, \rho_{-l}, \mathbf{w}, \sigma^2) = N\left(y_i \left| \sum_{j=1}^{n} w_j k^{(-l)}(\mathbf{x}_i, \mathbf{x}_j) \exp\left(-h_k^2 (x_{li} - x_{lj})^2\right), \sigma^2 \right.\right).$$

For each possible choice $\rho_l = h_k$, where $k = 1, ..., m$, we calculate

$$\begin{aligned} r_k &= \prod_{i=1}^{n} p(y_i | \rho_l = h_k, \rho_{-l}, \mathbf{w}, \sigma^2) \pi_k, \\ &= \prod_{i=1}^{n} N\left(y_i \left| \sum_{j=1}^{n} w_j k^{(-l)}(\mathbf{x}_i, \mathbf{x}_j) \exp\left(-h_k^2 (x_{li} - x_{lj})^2\right), \sigma^2 \right.\right) \pi_k. \end{aligned}$$

Let $r_0 = \sum_{j=1}^{m} r_j$, then the conditional posterior probability of $\rho_l = h_k$ is

$$p(\rho_l = h_k | D, \rho_{-l}, \mathbf{w}, \sigma^2) = \frac{r_k}{r_0}, \tag{3.36}$$

and the corresponding conditional posterior for $\rho_l$ is

$$\rho_l \sim \sum_{k=1}^{m} \frac{r_k}{r_0} \delta(\rho_l - h_k).$$

In summary, the Gibbs sampling components for kernel parameters will create a sequence of $\{\rho_1, \rho_2, ..., \rho_p\}$ through the following steps

- Given the current value of $\mathbf{w}$, $G$ and $\rho$, for $l = 1, 2, ..., p$,
  - for $k = 1, 2, ..., m$, calculate

$$\pi_k = g_l^{-2} \exp\left(-\frac{h_k^2}{2g_l^2}\right)$$

    and

$$r_k = \prod_{i=1}^{n} N\left(y_i \middle| \sum_{j=1}^{n} w_j k^{(-l)}(\mathbf{x}_i, \mathbf{x}_j) \exp\left(-h_k^2 (x_{li} - x_{lj})^2\right), \sigma^2\right) \pi_k.$$

  - let $r_0 = \sum_{j=1}^{m} r_j$, draw a sample $v \sim U(0, 1)$.
  - if $(\sum_{j=1}^{k} r_j)/r_0 \leq v \leq (\sum_{j=1}^{k+1} r_j)/r_0$, set $\rho_l = h_k$.
- Given the new value of $\rho$, draw a new diagonal matrix $G = diag(g_1^2, g_2^2, ..., g_p^2)$ from the $p$ independent inverse gamma posteriors

$$p(g_l^{-2} | \rho_l^2, D) = Ga\left(g_l^{-2} \middle| \frac{k+1}{2}, \frac{k g_0^2 + \rho_l^2}{2}\right).$$

Then above two components can be plugged into the MCMC procedures in Subsection 3.4.5 to construct the MCMC over full parameter space.

For the XOR data, we add 8-dimensional random noise, $x_{3i}, ..., x_{10i}$, to the training data to test if our algorithm can identify the irrelevant features. We set $\{h_1, h_2, ..., h_m\} = \{0, 0.2, 0.5, 1, 2, 4\}$, which works quite well for many cases. The MCMC was run for 10,000 iteration after a burn-in of 5,000. Figure 3.22 plots the means of $\rho_1, \rho_2, ..., \rho_{10}$. As we can see that the means of $\rho_1, \rho_2$ are much larger than the means of $\rho_3, \rho_4, ..., \rho_{10}$, which indicates that the algorithm can identify the relevant predictors successfully. Figure 3.24 shows the simple summary of predictive probabilities for the training data. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The data are labelled by case number and color coded – red for 1 and blue for zero. Figure 3.25 shows a similar summary for the test data. Note that there are increases of uncertainty in predictive probabilities compared to the previous analysis when the kernel parameter is fixed, which is due to the introduction of the noise in data and the uncertainty in kernel parameters. Taking $p_i = 0.5$ ($\widehat{y}_i = 0$) as the hard decision boundary, the error rate for the training data is 0 and the error rate for the test data is 5%. Note that the SVMs's error rate on the same data is 45%. Our result is much better.

### 3.5.3 MAP estimation via ECM algorithm

In the cases where we are just interested in the MAP estimate, we can still incorporate inference on kernel parameters $\rho$ into the EM algorithm in Subsection 3.4.6. Now the parameters of interests are the regression parameters $\mathbf{w}$ and kernel parameters $\rho$. All other parameters are considered as latent variables. Thus the maximum a posteriori (MAP) estimate of $\mathbf{w}$ and $\rho$ can be obtained by maximizing the marginal posterior

$p(\mathbf{w}, \rho|D)$. In this subsection, we will derive the EM algorithm for the binary probit regression model.

To simplify our illustration, we denote kernel matrix $K(\rho)$ as a function of $\rho$; the element $k_{ij}(\rho)$ is defined by the kernel function (3.4). Thus $\mathbf{w}$ and $\rho$ are the parameters to be estimated and $T = diag(\tau_1^2, \tau_2^2, ..., \tau_n^2)$, $G = diag(g_1^2, g_2^2, ..., g_p^2)$ and $\mathbf{y}$ is the latent variable or "missing data". We then have:

$$\begin{aligned} & Q(\mathbf{w}, \rho|\mathbf{w}^{old}, \rho^{old}) \\ = \; & E_{old}(\log p(\mathbf{w}, \rho|T, G, \mathbf{y}, D)), \\ = \; & \int (\log p(\mathbf{w}, \rho|T, G, \mathbf{y}, D)) p(T, G, \mathbf{y}|\mathbf{w}^{old}, D) dT d\mathbf{y}, \\ = \; & \int (\log p(\mathbf{w}, \rho, T, G, \mathbf{y}|D)) p(T|\mathbf{w}^{old}, D) p(G|\rho^{old}, D) p(\mathbf{y}|\mathbf{w}^{old}, D) dT d\mathbf{y}. \end{aligned}$$

Ignoring the terms that are unrelated to $\mathbf{w}$ and $\rho$, the logarithm of the joint posterior density $\log p(\mathbf{w}, T, \mathbf{y}|D)$ can be written as

$$\begin{aligned} \log p(\mathbf{w}, \rho, T, G, \mathbf{y}|D) &= -\frac{1}{2}\left((\mathbf{y} - K(\rho)^T\mathbf{w})^T(\mathbf{y} - K(\rho)^T\mathbf{w}) + \mathbf{w}^{\mathbf{T}} T^{-1}\mathbf{w} + \rho^{\mathbf{T}} G^{-1}\rho\right) \\ &= -\frac{1}{2}(\mathbf{w}^T K(\rho) K(\rho)^T\mathbf{w} + \mathbf{w}^{\mathbf{T}} T^{-1}\mathbf{w} - \mathbf{y}^T K(\rho)^T\mathbf{w}, \\ & -\mathbf{w}^T K(\rho)\mathbf{y} + \rho^{\mathbf{T}} G^{-1}\rho. \end{aligned}$$

For the E-step of the EM algorithm, we need to calculate $E_{old}(\log p(\mathbf{w}, \rho, T, G, \mathbf{y}|D))$ over $T, G, \mathbf{y}$ and conditional on the current value $\mathbf{w}^{old}$,$\rho^{old}$. This is

$$\begin{aligned} Q(\mathbf{w}, \rho|\mathbf{w}^{old}, \rho^{old}) &= E_{old}(\log p(\mathbf{w}, \rho|T, G, \mathbf{y}, D)), \\ &= -\frac{1}{2}(\mathbf{w}^T K(\rho) K(\rho)^T\mathbf{w} + \mathbf{w}^{\mathbf{T}} E_{old}(T^{-1})\mathbf{w} - E_{old}(\mathbf{y})^T K(\rho)^T\mathbf{w}, \\ & -\mathbf{w}^T K(\rho) E_{old}(\mathbf{y}) + \rho^{\mathbf{T}} E_{old}(G^{-1})\rho + \text{constant}. \end{aligned}$$

Given the conditional posterior for $\tau_i^{-2}$ (3.27) and $g_i^{-2}$ (3.36), $E_{old}(T^{-1})$ and $E_{old}(G^{-1})$ can be calculated analytically as

$$E_{old}(\tau_i^{-2}) = \frac{k+1}{k\tau_0^2 + (w_i^2)^{old}},$$

and

$$E_{old}(g_i^{-2}) = \frac{k+1}{kg_0^2 + (\rho_i^2)^{old}}.$$

The conditional posteriors for latent variables $y_i$ are independent truncated normals. Let $\widehat{y}_i = \mathbf{k}(\rho^{old})_i^T \mathbf{w}^{old}$, $p_i = N(\widehat{y}_i|0,1)$ and $P_i = \Phi(-\widehat{y}_i)$. The expectation $E_{old}(y_i)$ is

$$E_{old}(y_i) = \widehat{y}_i + (2z_i - 1)\frac{p_i}{z_i - (2z_i - 1)P_i}.$$

For the M-step, we must now find the $\mathbf{w}$ and $\rho$ that maximizes the the $Q$ function $Q(\mathbf{w}, \rho|\mathbf{w}^{old}, \rho^{old})$. Since $K(\rho)$ is a nonlinear function of $\rho$, it is difficult to find the analytical solution. Instead, we use a series of conditional maximization (CM) to replace the single M-step. The components of $(\mathbf{w}, \rho)$ are calculated one at a time, leaving the other component at its previous value. Given the current value of $\rho$ as $\rho^{(s)}$, we can find the analytical solution for $\mathbf{w}^{(s+1)}$ as

$$\mathbf{w}^{(s+1)} = \left(E_{old}(T^{-1}) + K(\rho^{(s)})K(\rho^{(s)})^T\right)^{-1} K(\rho^{(s)})E_{old}(\mathbf{y}).$$

To find the $\rho^{(s+1)}$ that maximizes the the $Q$ function $Q(\mathbf{w}^{(s+1)}, \rho|\mathbf{w}^{old}, \rho^{old})$, we have to apply numerical nonlinear optimization procedures, such as the conjugate gradient algorithm that is available in Matlab.

In summary, the EM algorithm can be described as follows:

1. Set a initial value for $\mathbf{y}$ and $T^{-1}$ and $\rho$, then calculate the initial estimate of $\mathbf{w}^{(0)}$ as

$$\mathbf{w}^{(0)} = \left(T^{-1} + K(\rho)K(\rho)^T\right)^{-1} K(\rho)\mathbf{y}.$$

2. For $t = 0, 1, ...$ :

(a) E-step: calculate the expectations

$$E_{old}(\tau_i^{-2}) = \frac{k+1}{k\tau_0^2 + (w_i^2)^{(t)}},$$

$$E_{old}(g_i^{-2}) = \frac{k+1}{kg_0^2 + (\rho_i^2)^{(t)}}.$$

and

$$E_{old}(y_i) = \widehat{y}_i + (2z_i - 1)\frac{N(\widehat{y}_i|0,1)}{z_i - (2z_i-1)\Phi(-\widehat{y}_i)},$$

where

$$\widehat{y}_i = \mathbf{k}(\rho^{(t)})_i^T \mathbf{w}^{(t)}.$$

(b) CM-step: For $s = 0, 1, ...$ :

i. calculate the new estimate of $\mathbf{w}^{(s+1)}$ as

$$\mathbf{w}^{(s+1)} = \left(E_{old}(T^{-1}) + K(\rho^{(s)})K(\rho^{(s)})^T\right)^{-1} K(\rho^{(s)})E_{old}(\mathbf{y}).$$

ii. find the $\rho^{(s+1)}$ that maximizes the the $Q$ function $Q(\mathbf{w}^{(s+1)}, \rho|\mathbf{w}^{old}, \rho^{old})$ using conjugate gradient algorithm.

iii. Repeat step (i) and step (ii) until the log-likelihood converges.

(c) Set $\mathbf{w}^{(t)} = \mathbf{w}^{(s+1)}$ and $\rho^{(t+1)} = \rho^{(s+1)}$.

(d) Repeat step (a) and step (b) until the log-likelihood converges.

We fit the model for the XOR example with 8-dimensional random noise, $x_{3i}, ..., x_{10i}$ via ECM algorithm. Figure 3.23 plots the value of $\rho_1, \rho_2, ..., \rho_{10}$. As we can see that the value of $\rho_1, \rho_2$ are much larger than the value of $\rho_3, \rho_4, ..., \rho_{10}$, which indicates that the algorithm can identify the relevant predictors successfully. Figure 3.26 and Figure 3.27 show the predictive probabilities for the training data and test data respectively. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{k}_i^T\mathbf{w}$. The data are labelled by case number and color coded – red for 1

and blue for zero. Taking $p_i = 0.5$ ($\widehat{y}_i = 0$) as the hard decision boundary, the error rate for the training data is 0 and the error rate for the test data is 5%, which is similar to that of MCMC.

Note that it is possible for the numerical nonlinear optimization procedure and corresponding ECM algorithm to find the local maximum instead of global maximum though we have not conducted any investigation into this issue. One possible solution to this problem is to use the stochastic ECM (SECM). In the E-step of above ECM algorithm, we draw a sample of $T^{-1}, G^{-1}$ and $\mathbf{y}$ from their conditional posteriors. Then the CM steps are conditional on those samples instead of the expectations. SECM algorithm generates a Markov chain of $(\mathbf{w}, \rho, T^{-1}, G^{-1}, \mathbf{y})$ that can be used in the inference of interested parameters.

## 3.6 Experimental results

Besides the artificial XOR data examples, we have tried our Bayesian kernel models on some real world benchmark datasets to evaluate the classification performance. The comparison with SVMs, RVM and other quoted classifiers are provided.

MATLAB implementation of SVM is from the Spider project that are available in http://www.kyb.tuebingen.mpg.de/bs/people/spider/main.html. The Gaussian kernel function

$$k(\mathbf{x}_j, \mathbf{x}_i) = \exp\left(-\rho^2 \sum_{h=1}^{p} (x_{hi} - x_{hj})^2\right), \tag{3.37}$$

is used and the parameter $\rho^2$ is pre-selected or chosen by cross validation.

MATLAB implementation of RVM is available in Microsoft's RVM site: http://research.microsoft.com/mlp/rvm/relevance.htm. The same Gaussian kernel function and kernel parameters in SVM are used.

Two Bayesian kernel models are applied here: BKM1 is the orthogonalized kernel

models. We choose the SVD factors with largest $k$ eigenvalues that have explained 95% of linear structure. BKM1 uses the exact same Gaussian kernel function and kernel parameters as in SVM and RVM. It is trained via EM algorithm. BKM2 uses the Gaussian kernel function

$$k(\mathbf{x}_j, \mathbf{x}_i) = \exp\left(-\sum_{l=1}^{p} \rho_l^2 (x_{li} - x_{lj})^2\right),$$

where $\rho_l^2$ is an individual scale parameter for each dimension. The model is trained via ECM algorithm where the kernel parameters are trained in CM step as described in previous section.

In the following subsections, the comparison results for different benchmark datasets are reported. All continuous variables are standardized so that they have zero mean and unit variance on the training dataset.

### 3.6.1 Diabetes in Pima Indians

The data were collected by the US National Institute of Diabetes and Digestive and Kidney Diseases (Smith et al, 1988). It is the result of diabetes testing for a population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona. The data set includes 8 predictor variables and 1 binary response variable. We used the data as made available in Ripley (1994) with his training test split of 200 training data 322 test data respectively. The baseline error obtained by simply classifying each record as coming from a diabetic gives rise to an error of 33%. Table 3.1 reports the number of test errors by different classifiers. The results of neural network and linear discriminant are from Ripley (1996). The performance of BKM2 is the best of the methods in comparison, and both Bayesian kernel models give performance comparable to SVM and RVM.

### 3.6.2 Leptograpsus crabs

The data are from Campbell & Mahon (1974) on the morphology of rock crabs of genus Leptograpsus. In this problem we attempt to classify the sex of crabs on the basis of five anatomical attributes with an optional additional color attribute. There are 50 specimens available for crabs of each sex and color making a total of 200 labelled examples. These are split into a training set of 20 crabs of each sex and color, making 80 training examples with the other 120 examples used as the test set. Table 3.2 reports the number of test errors by different classifiers. Again the results of neural network and linear discriminant are from Ripley (1996). The performance of BKM1 and BKM2 are the best of the methods in comparison.

### 3.6.3 Titanic

The data is available in http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm. It includes 3 predictor variables and 1 binary response variable. We use a total of 10 training/test splits provided. Table 3.3 reports the test errors rate by different classifiers. The result of Kernel Fisher Discriminant is from Mika *et al.* (1999). The result of various AdaBoost methods are from Rätsch *et al.* (2001). The performance of BKM1 and BKM2 are comparable to results provided by those state-of-art nonlinear classifiers.

### 3.6.4 Breast cancer

The data was obtained from the University Medical Center, Inst. of Oncology, Ljubljana, Yugoslavia. It is available in
http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm. It includes 9 predictor variables and 1 binary response variable. The number of train data is : 200 and number of test data is 77. We use a total of 10 training/test splits provided. Table

3.4 reports the test errors rate by different classifiers. The result of Kernel Fisher Discriminant is from Mika *et al.* (1999). The result of various AdaBoost methods are from Rätsch *et al.* (2001). The performance of BKM2 is among the best of the methods in comparison.

| Method | Pima Indians |
|---|---|
| Neural network | 75 |
| Linear discriminant | 67 |
| SVM | 64 |
| RVM | 64 |
| BKM1 | 65 |
| BKM2 | 63 |

**Table 3.1**: Number of test errors by different classifiers on Pima Indians dataset: The results of neural network and linear discriminant are from Ripley (1996). Both Bayesian kernel models give comparable performance with SVM and RVM .

| Method | crabs |
|---|---|
| Neural network | 3 |
| Linear discriminant | 8 |
| SVM | 4 |
| RVM | 3 |
| BKM1 | 2 |
| BKM2 | 1 |

**Table 3.2**: Number of test errors by different classifiers on Leptograpsus crabs: The results of neural network and linear discriminant are from Ripley (1996). The performance of BKM1 and BKM2 are the best of the methods in comparison.

| Method | Titanic |
|---|---|
| Kernel Fisher Discriminant | 23.25% |
| AdaBoost with RBF-Network | 22.58% |
| LP Reg-AdaBoost | 23.98% |
| QP Reg-AdaBoost | 22.71% |
| AdaBoost Reg | 22.64% |
| SVM | 23.55% |
| RVM | 24.02% |
| BKM1 | 23.11% |
| BKM2 | 23.61% |

**Table 3.3**: Test errors rate by different classifiers on Titanic: The result of Kernel Fisher Discriminant is from Mika(1999). The result of various AdaBoost methods are from Ratsch (2001). The performance of BKM1 and BKM2 are comparable to results provided by those state of art nonlinear classifiers.

| Method | Breast cancer |
|---|---|
| Kernel Fisher Discriminant | 24.77% |
| AdaBoost with RBF-Network | 30.36% |
| LP Reg-AdaBoost | 26.79% |
| QP Reg-AdaBoost | 25.91% |
| AdaBoost Reg | 26.51% |
| SVM | 26.98% |
| RVM | 29.56% |
| BKM1 | 28.23% |
| BKM2 | 25.12% |

**Table 3.4**: Test errors rate by different classifiers on Breast cancer: The result of Kernel Fisher Discriminant is from Mika(1999). The result of various AdaBoost methods are from Ratsch (2001). The performance of BKM2 is among the best of the methods.

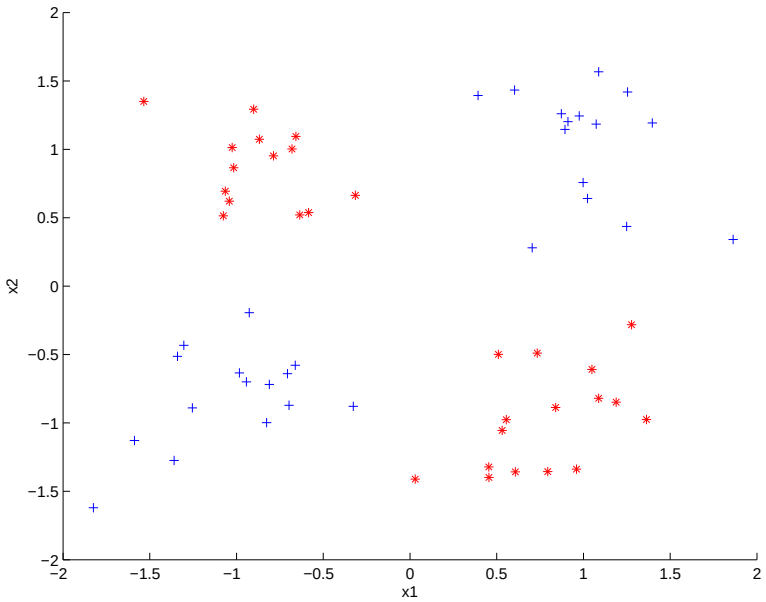


**Figure 3.1**: 2-dimensional XOR training data: 60 independent samples are drawn from a mixture of Gaussians as training data. Blue '+' represents class 0 and red 'o' represents class 1.

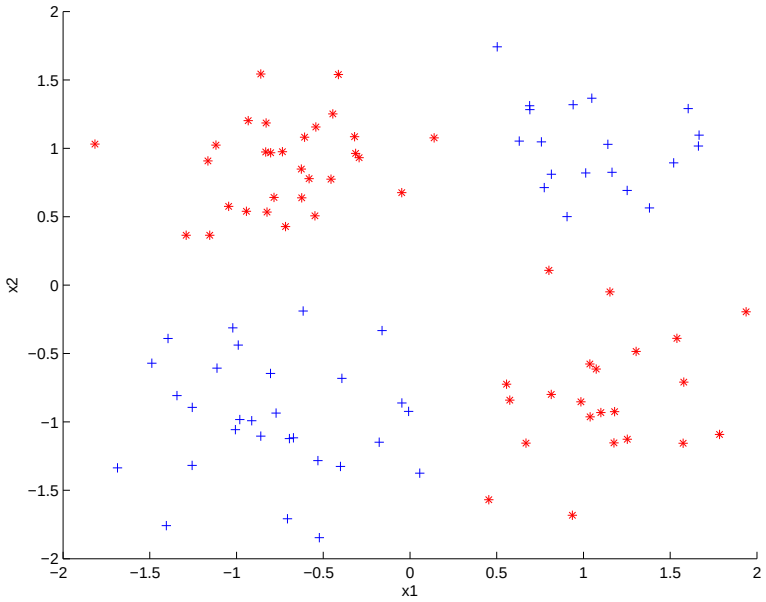


**Figure 3.2**: 2-dimensional XOR data: 100 independent samples are drawn from a mixture of Gaussians as test data. Blue '+' represents class 0 and red 'o' represents class 1.

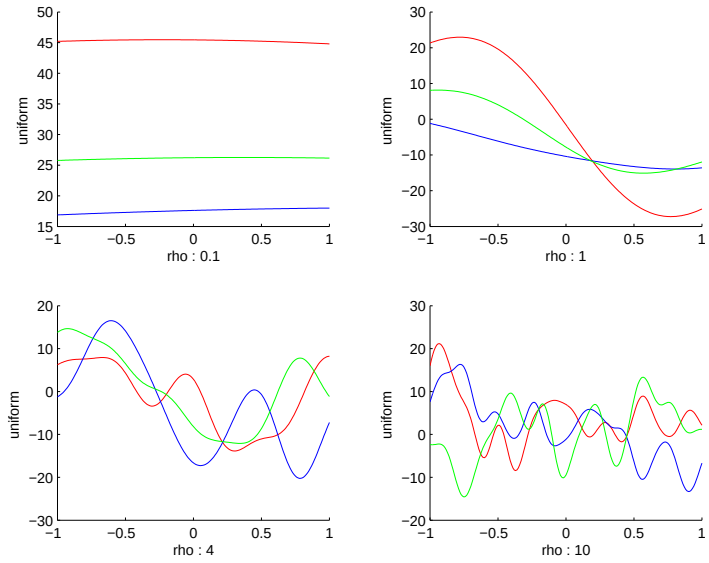


**Figure 3.3**: One-dimensional simulations of $f(\mathbf{x})$: $F \sim U(-1, 1)$. Four figures correspond to the four different Gaussian kernel parameters $\rho = \{0.1, 1, 4, 10\}$. Each of figures shows three samples of function $f(x)$. It is shown that $f(x)$ is just a stationary Gaussian process with stationary smoothing kernel $k$.

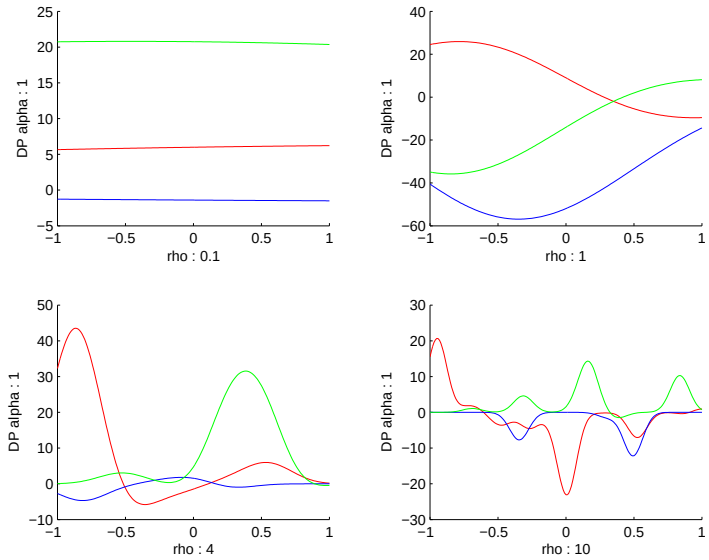


**Figure 3.4**: One-dimensional simulations of $f(\mathbf{x})$: $F \sim DP(F|F_0, \alpha)$ with $\alpha = 1$. Four figures correspond to the four different Gaussian kernel parameters $\rho = \{0.1, 1, 4, 10\}$. Each of figures shows three samples of function $f(x)$. It is shown that the function $f(x)$ is no longer stationary due to the Dirichlet process prior, and it is clearer with larger value of $\rho$.

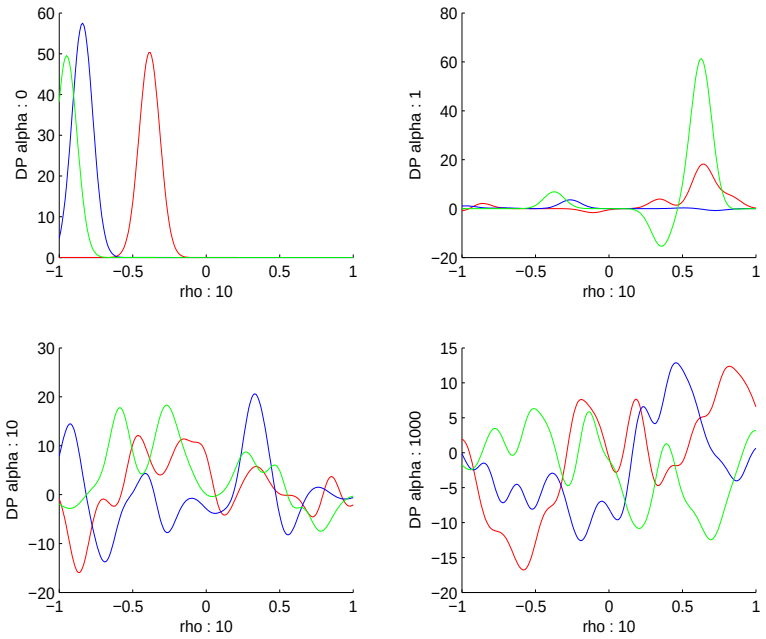


**Figure 3.5**: One-dimensional simulations to show the effect of different scalar precision parameters $\alpha$ on $f(x)$: $F \sim DP(F|F_0, \alpha)$ with fixed $\rho = 10$. Four figures correspond to the four different scalar precision parameters $\alpha = \{0, 1, 10, 1000\}$. Each of figures shows three samples of function $f(x)$. With the increase of the $\alpha$, we have more confidence on our baseline prior $F_0$, and $f(x)$ looks more like a stationary Gaussian process.

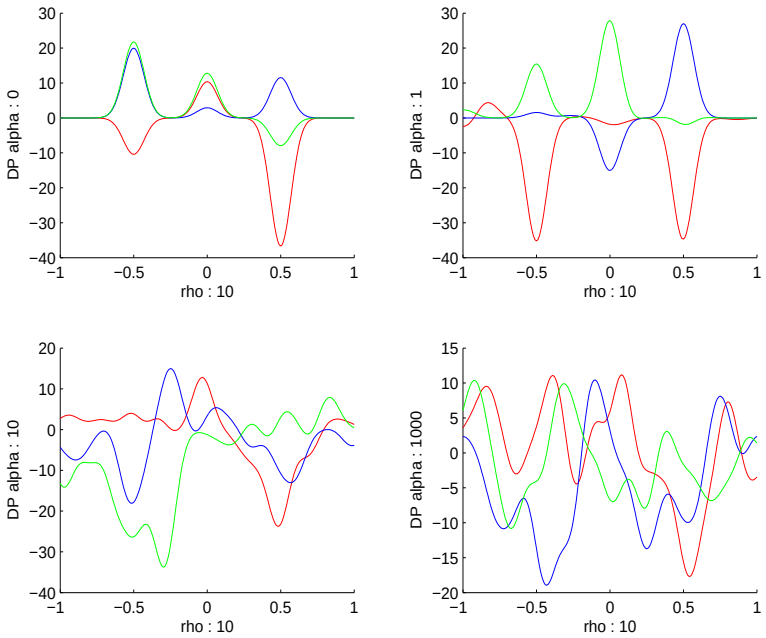


**Figure 3.6**: One-dimensional simulations to show the effect of different scalar precision parameters $\alpha$ on $f(x)$: $F \sim DP(F|F_0, \alpha)$ with fixed $\rho = 10$. we assume that the first three samples of $\mathbf{u}$ are $\mathbf{u}_1 = -0.5, \mathbf{u}_2 = 0, \mathbf{u}_3 = 0.5$. Four figures correspond to the four different scalar precision parameters $\alpha = \{0, 1, 10, 1000\}$. Each of figures shows three samples of function $f(x)$. As we can see, the function $f(x)$ has a high concentration on initial distinct values for small $\alpha$. But as we increase the $\alpha$, $\mathbf{u}$ has more distinct values drawn from $F_0$. Thus the baseline prior $F_0$ has more smoothing effects on $f(x)$. It is consistent with the fact that the asymptotic mean of $k_n$ is an increasing function of $\alpha$ (West 1992). The larger $\alpha$ implies larger number of distinct values.

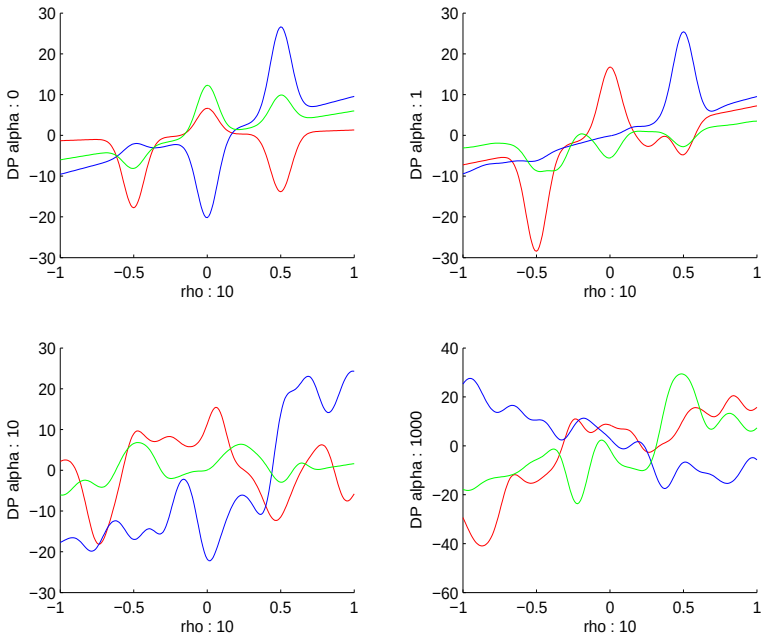


**Figure 3.7**: One-dimensional simulations with sum of Gaussian kernel and linear kernel: $F \sim DP(F|F_0, \alpha)$ with fixed $\rho = 10$. The kernel function is the sum of Gaussian kernel and linear kernel. Four figures correspond to the four different scalar precision parameters $\alpha = \{0, 1, 10, 1000\}$. Each of figures shows three samples of function $f(x)$.

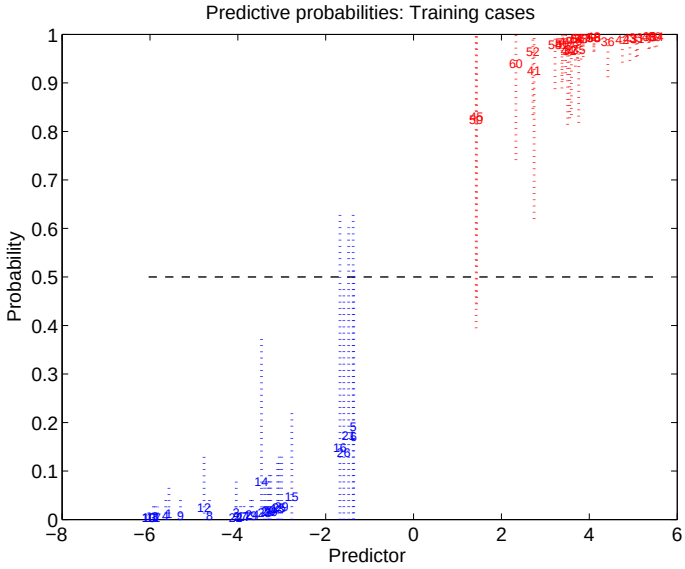


**Figure 3.8**: Bayesian kernel models via MCMC: summary of predictive probabilities for the training data in the XOR example. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The data are labelled by case number and color coded – red for 1 and blue for zero.

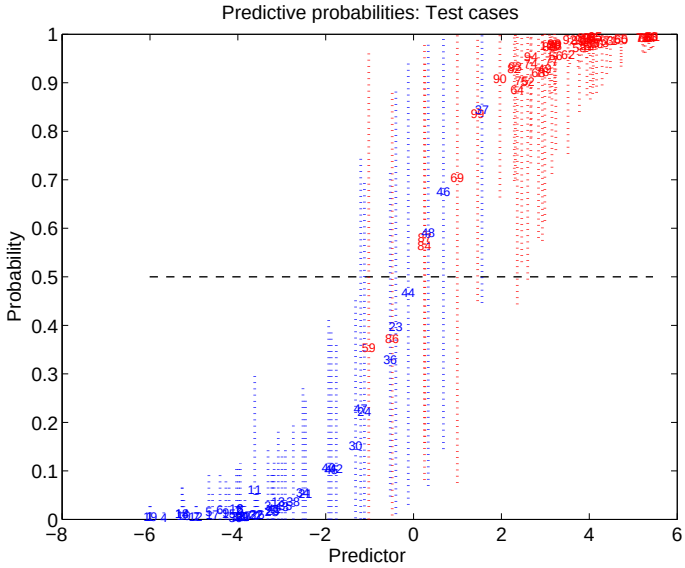


**Figure 3.9**: Bayesian kernel models via MCMC: summary of predictive probabilities for the test data in the XOR example. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The data are labelled by case number and color coded – red for 1 and blue for zero.

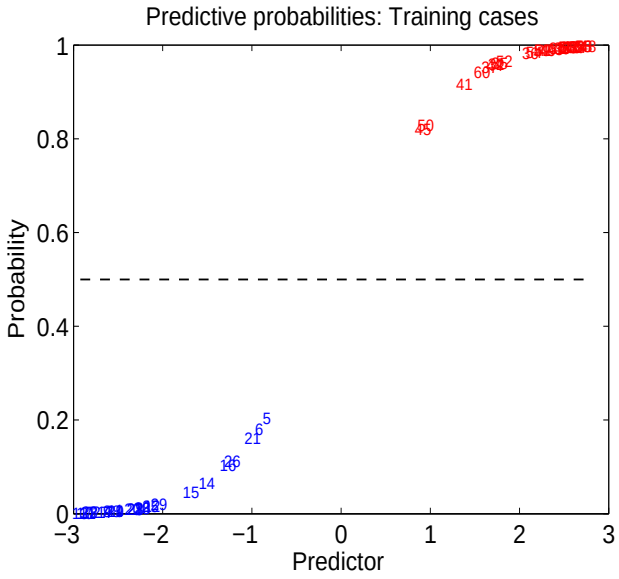


**Figure 3.10**: Bayesian kernel models via EM: predictive probabilities for the training data in the XOR example. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The data are labelled by case number and color coded – red for 1 and blue for zero.

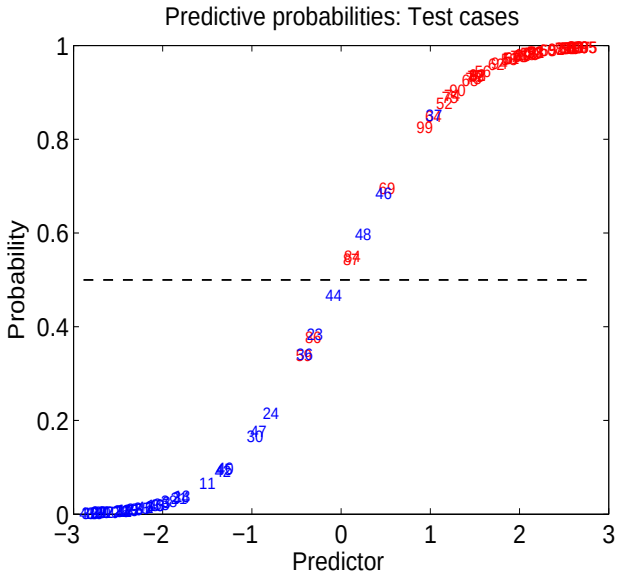


**Figure 3.11**: Bayesian kernel models via EM: predictive probabilities for the test data in the XOR example. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The data are labelled by case number and color coded – red for 1 and blue for zero. The error rate for the test data is 5%, which is similar to that of SVMs.

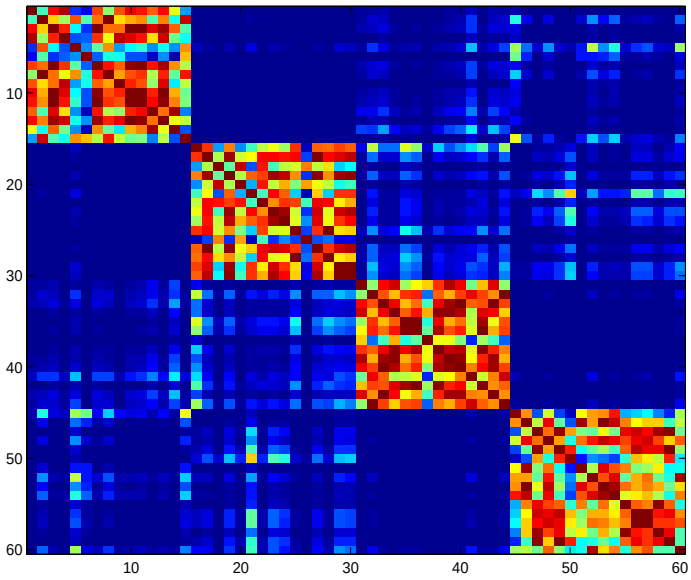


**Figure 3.12**: Kernel matrix for the XOR training data: The kernel function is the Gaussian kernel (3.3) with $\rho^2 = 1$. Data points are reorganized to show the cluster structure of the kernel matrix.

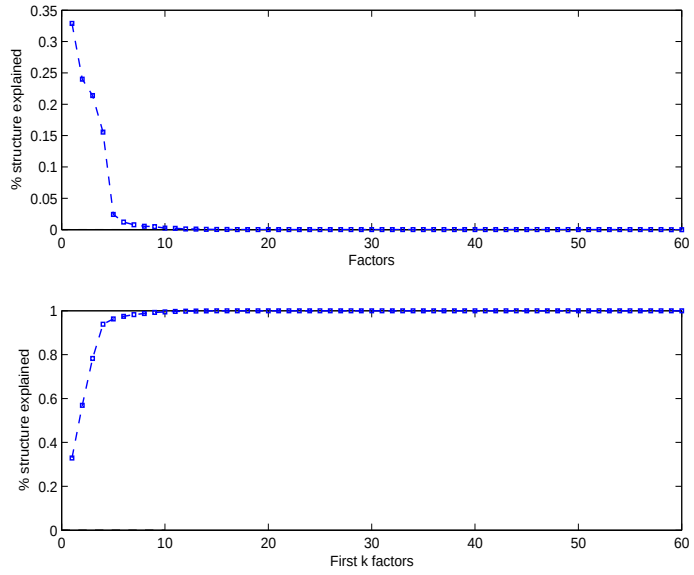


**Figure 3.13**: The percentage of variance explained by the SVD factors: The upper frame provides display of elements $d_i^2$ as a percentage of total $\sum_{i=1}^{n} d_i^2$, the lower frame shows the percentage of variance explained by the largest $k$ factors, i.e., $(\sum_{i=1}^{k} d_i^2)/(\sum_{i=1}^{n} d_i^2)$. It is suggested that we can approximate $K$ setting $m \ll n$ without significant decrease in the accuracy of the solution.

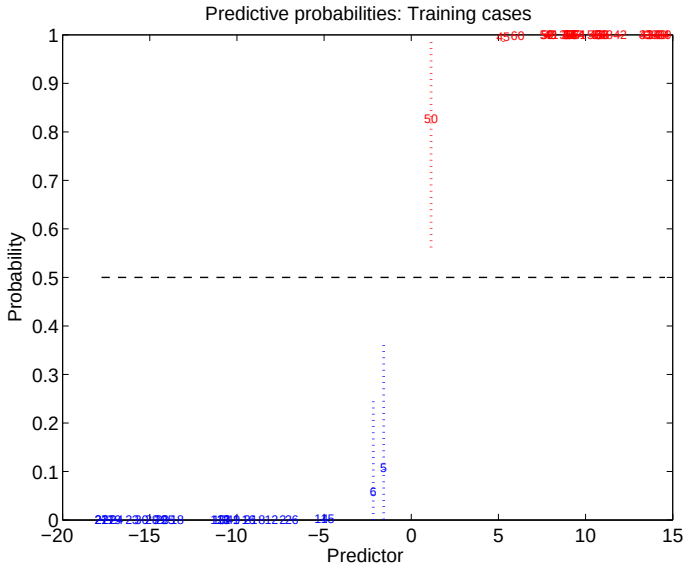


**Figure 3.14**: Orthogonalized kernel models via MCMC: summary of predictive probabilities for the training data in the XOR example. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$.

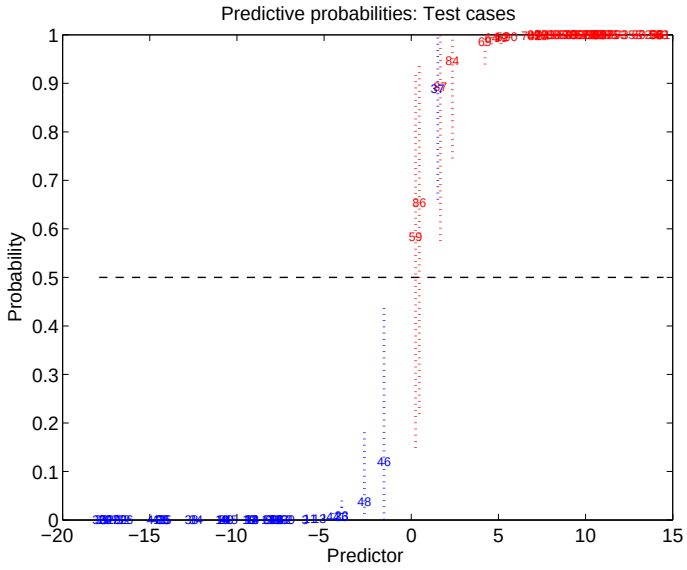


**Figure 3.15**: Orthogonalized kernel models via MCMC: summary of predictive probabilities for the test data in the XOR example. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The error rate for the test data is 1%, which is better than the Bayesian kernel model built on the kernel matrix directly.

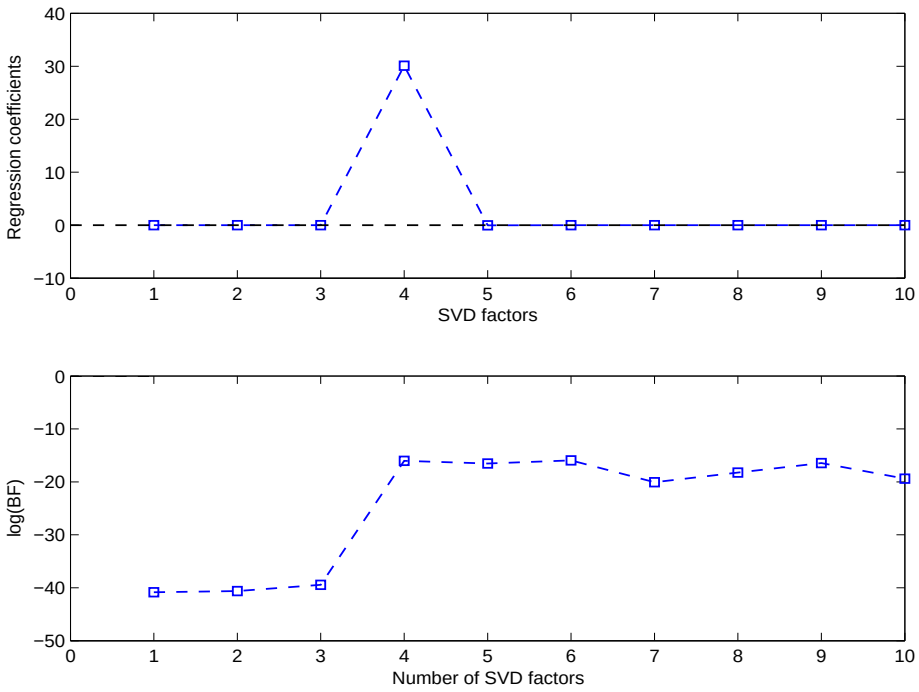


**Figure 3.16**: Bayes factors: The upper frame plots the posterior mean of $\gamma$ in the MCMC analysis of XOR data with $m = 10$. We can clearly see that the $4th$ principal component is most important for regression though its eigenvalue is much smaller that first 3 principal components that have already explained majority of linear structure. The lower frame shows the value of marginal distribution in log scale, $\log p(\mathbf{z})$, for 10 different models with $m = 1, 2, ..., 10$. It shows a significant increase in $m = 4$, which implies the importance of $4th$ principal component in the regression model.

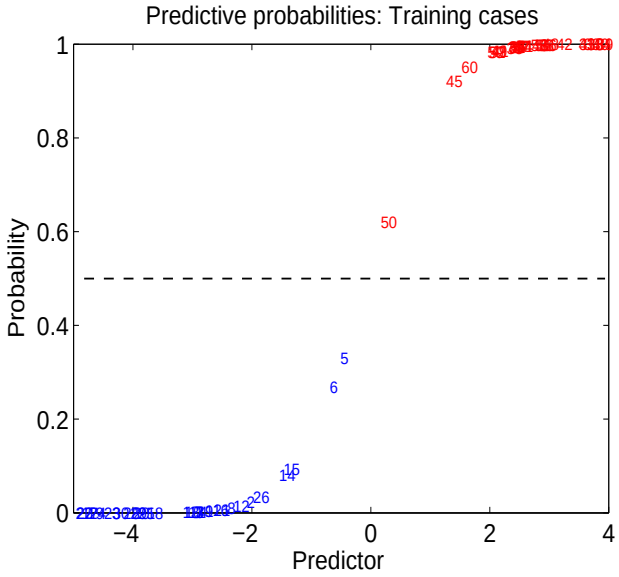


**Figure 3.17**: Orthogonalized kernel models via EM: predictive probabilities for the training data in the XOR example. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$.

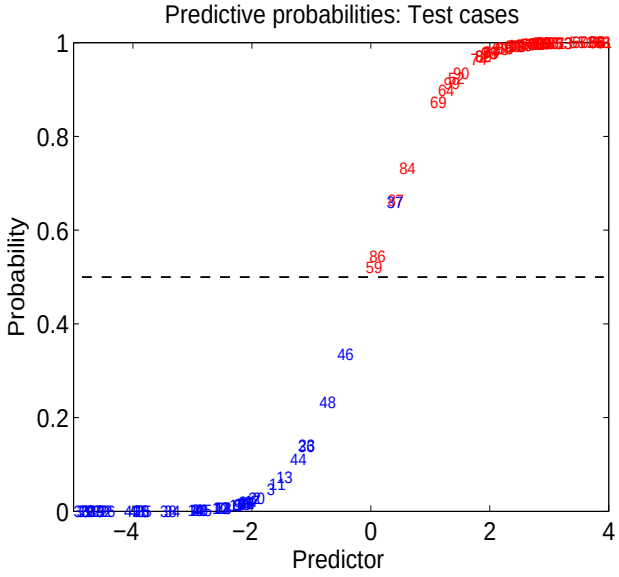


**Figure 3.18**: Orthogonalized kernel models via EM: predictive probabilities for the test data in the XOR example. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The error rate for the test data is 1%, which is similar to that of MCMC.

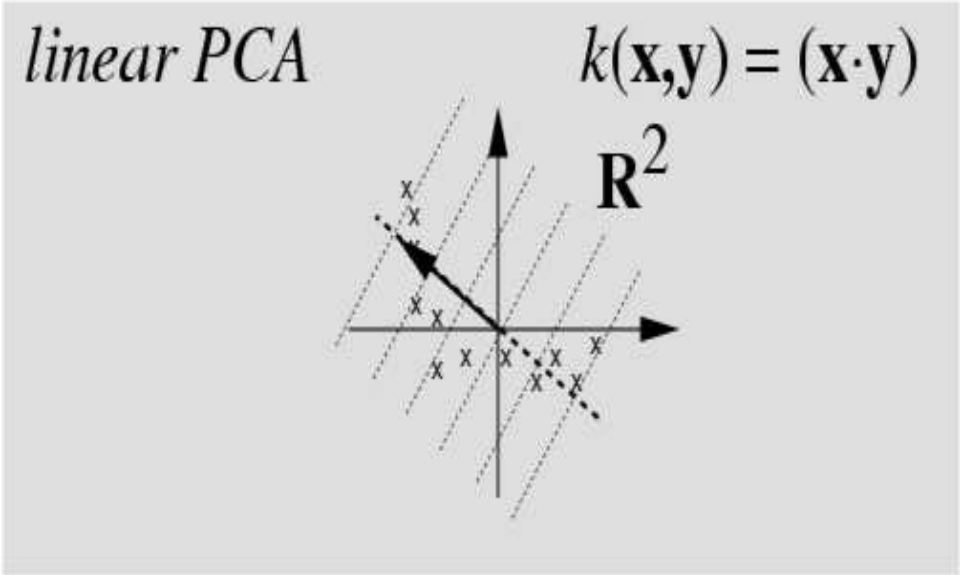


**Figure 3.19**: Illustrative example of linear PCA on the data with nonlinear structure, copied from Schoelkopf (1997). We can see that the linear PCA cannot capture the nonlinear structure in the data.

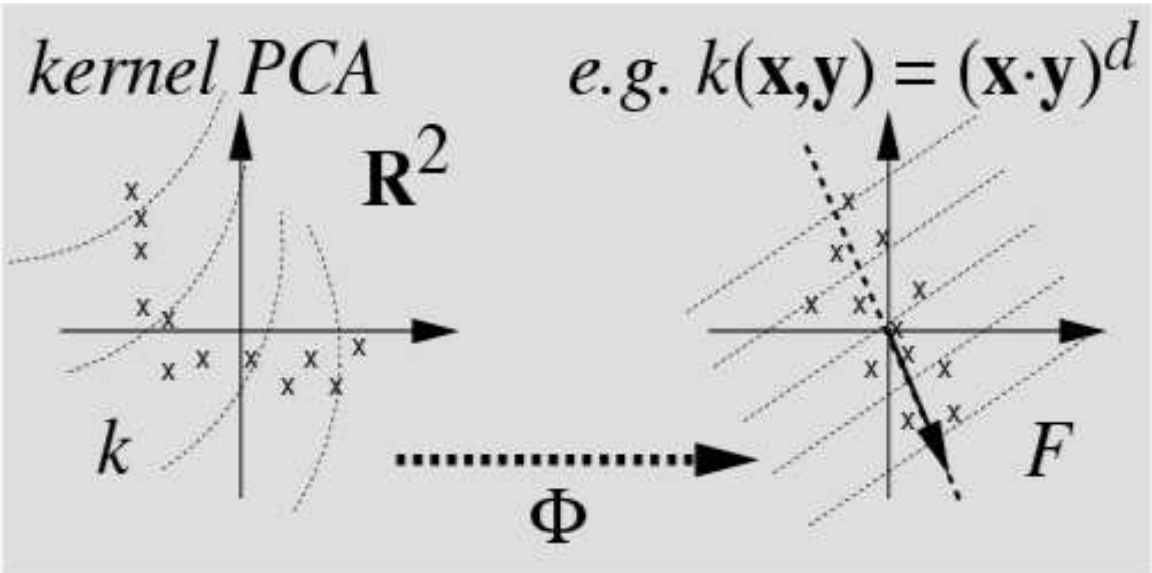


**Figure 3.20**: Illustrative example of kernel PCA, copied from Schoelkopf (1997). All necessary calculations are done by using polynomial kernel function in the original input space, which is equivalent to the linear PCA in the nonlinear feature space by polynomial transformation.

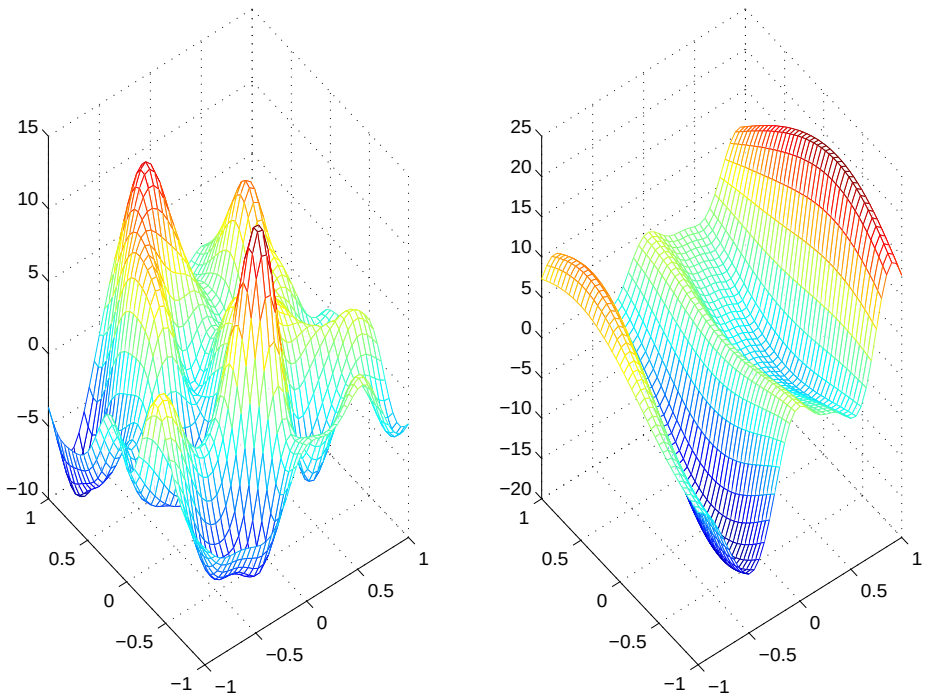


**Figure 3.21**: Simulations in two-dimensional space: The effect of scale parameters on $f(\mathbf{x})$ in two-dimensional space. On the left side $\rho_1^2 = \rho_2^2 = 20$, while on the right $\rho_1^2 = 1$ and $\rho_2^2 = 20$.

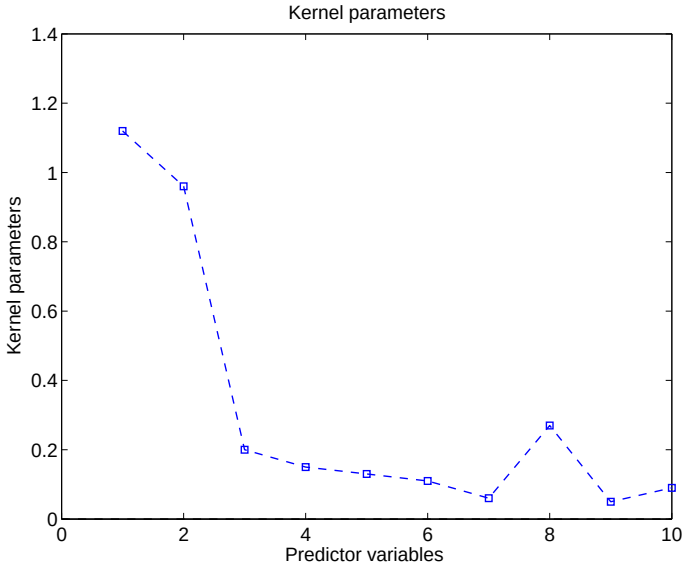


**Figure 3.22**: Estimation of kernel parameters via MCMC: In the XOR example, we add 8-dimensional random noises, $x_{3i}, ..., x_{10i}$, to the training data. This figure plots the posterior means of $\rho_1, \rho_2, ..., \rho_{10}$ after MCMC analysis. As we can see that the means of $\rho_1, \rho_2$ is much larger than the means of $\rho_3, \rho_4, ..., \rho_{10}$, which indicate that the algorithm can identify the relevant predictors successfully.

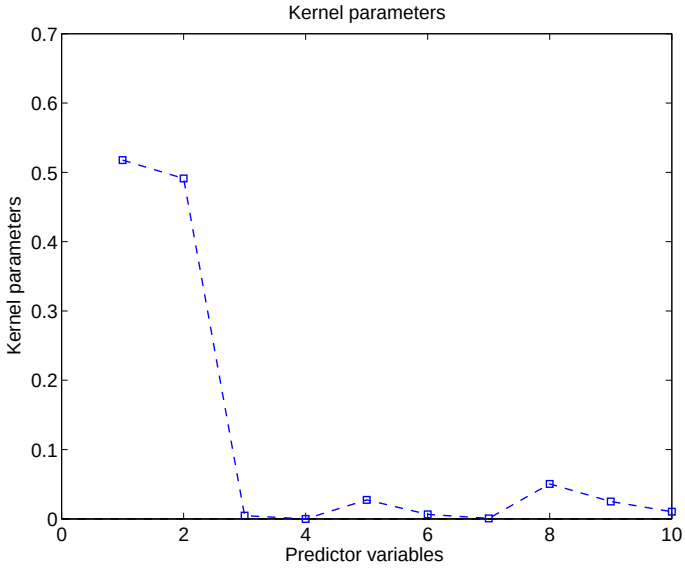


**Figure 3.23**: Estimation of kernel parameters via ECM: In the XOR example, we add 8-dimensional random noises, $x_{3i}, ..., x_{10i}$, to the training data. This figure plots the estimates of $\rho_1, \rho_2, ..., \rho_{10}$ after ECM analysis. As we can see that the means of $\rho_1, \rho_2$ is much larger than the means of $\rho_3, \rho_4, ..., \rho_{10}$, which indicate that the algorithm can identify the relevant predictors successfully.

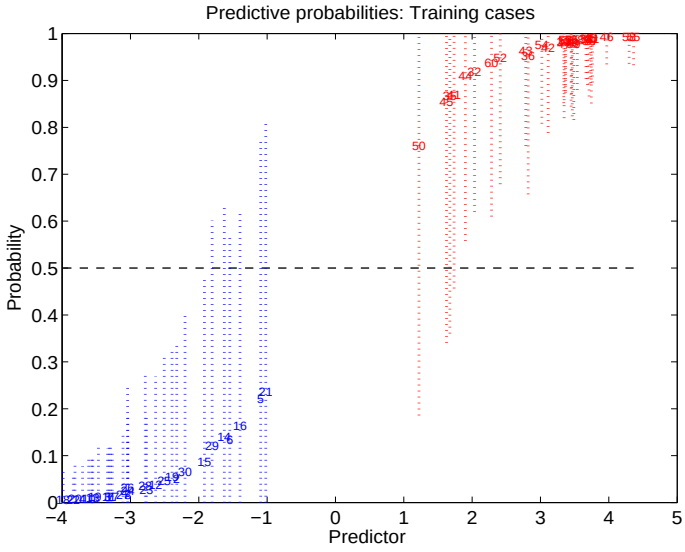


**Figure 3.24**: Bayesian kernel models with the estimation of kernel parameters via MCMC: summary of predictive probabilities for the training data in 10-dimensional XOR example. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$.

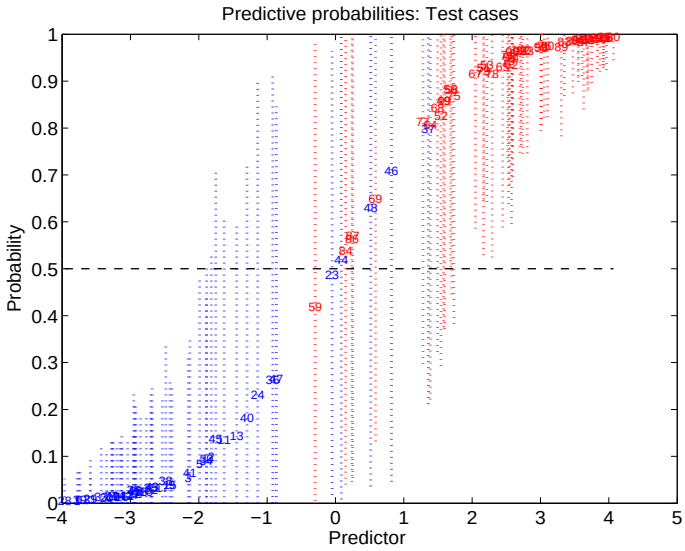


**Figure 3.25**: Bayesian kernel models with the estimation of kernel parameters via MCMC: summary of predictive probabilities for the training data in 10-dimensional XOR example. The posterior means of predictive probabilities $p_i = \Phi(\widehat{y}_i)$, with approximate 90% uncertainty interval, are plotted against posterior means of predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$. The error rate for the test data is 5%. Note that the SVMs's error rate on the same data is 45%. Our result is much better.

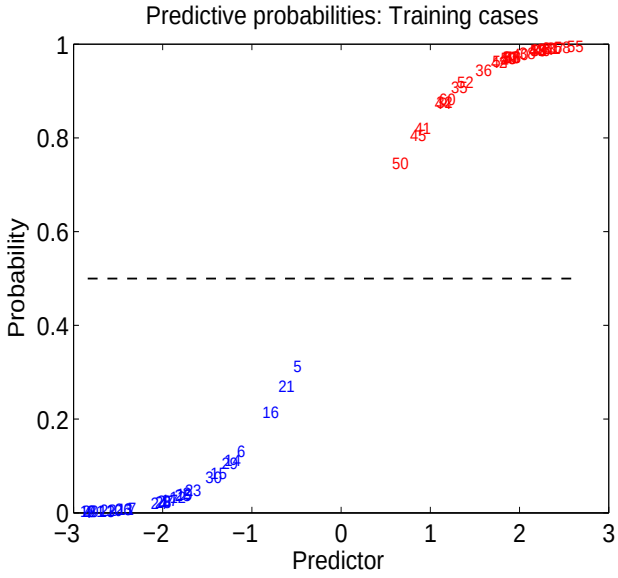


**Figure 3.26**: Bayesian kernel models with the estimation of kernel parameters via ECM: predictive probabilities for the training data in the 10-dimensional XOR example. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$.

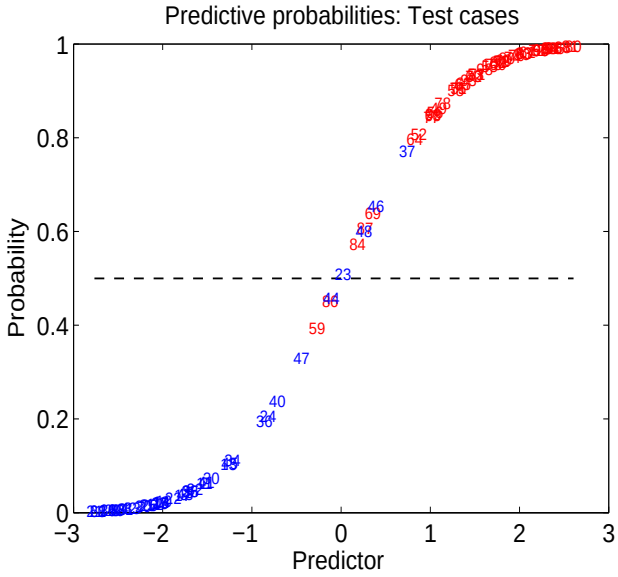


**Figure 3.27**: Bayesian kernel models with the estimation of kernel parameters via ECM: predictive probabilities for the test data in the 10-dimensional XOR example. The point estimate of predictive probabilities $p_i = \Phi(\widehat{y}_i)$ are plotted against the predictors $\widehat{y}_i = \mathbf{k}_i^T \mathbf{w}$.

# Chapter 4

# Conclusions and future work

The current thesis consists of two major parts. In the first part, a method to estimate gene expression indexes in the Bayesian framework is developed. In the second part, a new class of Bayesian kernel models is proposed. This Chapter presents some of the current ideas about possible extensions of each part.

## 4.1 Bayesian estimation of gene expression index

In Chapter 1, A Bayesian model of gene expression index is proposed. In terms of model specifications and applications, it is possible to extend the Bayesian model in the following ways.

### 4.1.1 Random effect models

One important assumption of LW model and our Bayesian model is that the probe-specific affinity $\phi_j$ is the same across the arrays, which is suggested by the observed reproducibility of probe-specific effects. However, it is more realistic to allow different probe-specific affinities cross the arrays due to the potentially complex experimental sources of variation.

We could introduce a random effect model,

$$y_{ij} = \phi_{ij}\theta_i + \varepsilon_{ij}, \tag{4.1}$$
$$\varepsilon_{ij} \sim N(\varepsilon_{ij}|0, \sigma_j^2/\lambda_{ij}),$$

where $\phi_{ij}$ is the probe-specific affinity for $jth$ probe pair in $ith$ array. We can specify the following hierarchical prior for $\phi_{ij}$,

$$\phi_{ij} \sim N(\phi_{ij}|\mu_j, \tau_j^2)I(\phi_{ij} > 0),$$

where $\mu_j$ and $\tau_j^2$ are the hyperparameters for $jth$ probe pair. The hyperpriors for these hyperparameters can be specified as uninformative priors,

$$\mu_j \sim U(\mu_j|0, \infty),$$
$$\tau_j^{-2} \sim Ga\left(\tau_j^{-2} \left| \frac{n_0}{2}, \frac{n_0}{2}\right.\right),$$

where $n_0$ is a small value.

The full conditional posteriors for $\phi_{ij}$, $\mu_j$, $\tau_j^2$ can be easily derived and plugged into the Gibbs sampling procedure in Chapter 1. This random effect model can be further improved if we know more about the experiment design, such as cell line, treatment, replicate etc.

### 4.1.2 PM only model

In oligonucleotide microarrays, hybridization of MM probes should be attributed to either cross-hybridization, or background signal caused by the hybridization of cell debris and salts to the probes. The rationale behind the use of difference of PM and MM is that PM-MM can be used to correct non-specific binding. However, various exploratory analysis (Naef *et al.*, 2001; Irizarry *et al.*, 2003a) suggest that MM may detect signal as well as non-specific binding. Based on the observations that PM-MM could remove the signal, researchers propose various measures based

only on PM or background corrected PM. One popular measure is robust multi-array average (RMA) (Irizarry *et al.*, 2003a; Irizarry *et al.*, 2003b). We can also extend our Bayesian analysis to these background adjusted PM based models. Assuming each array has a common background, a random effect lognormal model for background corrected PM can be

$$\begin{aligned} \log(PM_{ij} - BG_i) &= \alpha_i + \beta_{ij} + \varepsilon_{ij}, \\ \varepsilon_{ij} &\sim N(\varepsilon_{ij}|0, \sigma_j^2/\lambda_{ij}), \\ \lambda_{ij} &\sim Ga(\lambda_{ij}|\frac{k}{2}, \frac{k}{2}), \end{aligned} \tag{4.2}$$

where $BG_i$ is the estimated background for chip $i$. Irizarry *et al.* (2003a) suggests that $\log(BG_i)$ can be estimated by the mode of $\log(MM)$ distribution. $\alpha_i$ is the log scale expression index for array $i$, which can be considered as $\log(\theta_i)$ in the original Bayesian model (1.1). $\beta_{ij}$ is the probe-specific affinity for $jth$ probe pair in $ith$ array, which can be considered as $\log(\phi_{ij})$ in the random effect model (4.1). To make the model identifiable, we can impose constraints on $\beta_{ij}s$ as $\sum_j \beta_{ij} = 0$. This model can be considered as a multiplicative model on the original scale, while the original Bayesian model (1.1) is an additive model.

The prior specification for $\alpha_i$, $\beta_{ij}$ and $\sigma_j^2$ is similar to the original Bayesian model (1.1) and the random effect model (4.1) except that the positive constraints are removed.

It is possible to extend this model to account for the scaling factor for each array for the purpose of linear normalization. Let $PM_{ijk}$ be the PM probe data for chip $i$, probe $j$, and gene $k$. Assuming the normalization can be done linearly, i.e., each array has a common scaling factor, a random effect model is

$$\begin{aligned} \log(PM_{ijk} - BG_i) &= \gamma_i + \alpha_{ik} + \beta_{ijk} + \varepsilon_{ijk}, \\ \varepsilon_{ijk} &\sim N(\varepsilon_{ijk}|0, \sigma_{jk}^2/\lambda_{ijk}), \end{aligned}$$

where $\gamma_i$ is the scaling factor for array $i$. To make the model identifiable, we have to impose constraints on $\gamma_i$s as $\sum_i \gamma_i = 0$.

### 4.1.3 Integration with high-level analysis

Currently, all probe-level analyses of microarray data are independent of high-level analysis, such as clustering, classification and regulatory network. Some possible limitations include:

- The probe-level analyses, such as Bayesian model-based methods, not only can provide point estimate of gene expression index, but also can provide the posterior distribution to access the uncertainty. However, most of current high-level analysis just use the point estimate discarding the uncertainty totally.

- The current model based gene expression index are estimated gene by gene independently. The results for a specific gene $k$ could be greatly affected by the factors like cross-hybridization and image contamination. Sometimes high-level analysis could provide useful information of other genes to make adjustment for the results of this specific gene $k$. For example, suppose that gene $k$ is found to be highly correlated to a group of genes by high-level analysis or biological experiments, the expression index of gene $k$ is supposed to be highly correlated those of other genes too. However, some local image contaminations distort the probe data for gene $k$ in some arrays, which result in the wrong estimate of $\alpha_{ik}$s. By pooling the expression index of those correlated genes, it will be possible to correct the estimates of $\alpha_{ik}$ to a certain extent. In that case, it will be useful to integrate the information in high-level analysis into the probe-level analysis, which will improve the results of both analyses.

While it could be difficult to conduct a fully integrated analysis in classical way, Bayesian analysis with the help of MCMC provides such possibilities. In the setting

of binary probit regression, an integrated Bayesian model including model-based estimation of gene expression index, factor analysis and probit regression could be

$$\log(PM_{ijk} - BG_i) = \alpha_{ik} + \beta_{ijk} + \varepsilon_{ijk},$$

$$\alpha_i = B\mathbf{f}_i + \mathbf{v}_i,$$

$$y_i = \mathbf{f}_i^T \lambda + \epsilon_i,$$

where $\alpha_i = [\alpha_{i1}, \alpha_{i2}..., \alpha_{ip}]^T$ is a $p$-dimensional vector of gene expression index for $p$ genes in $ith$ array, $\mathbf{f}_i$ is a $k$-dimensional vector of latent factors, $B$ is the $p \times k$ factor loading matrix, $\lambda$ is the $k$-dimensional vector of regression coefficient and $y_i$ is the latent variable corresponding to the binary output variable $z_i$ in $ith$ sample. The detailed illustration of factor regression model can be found in West (2000) and West (2003) .

Note that we now make the inference of $\alpha_{ik}$ corresponding to the gene $k$ from the joint posterior distribution $p(\alpha_{i1}, ..., \alpha_{ik}, ..., \alpha_{ip}, \mathbf{f}_i, \lambda | \mathbf{PM}, \mathbf{y})$. As we can see, it does not only depend on $\mathbf{PM}$, the probe data for all $p$ genes, but also depends on $\mathbf{y}$ which includes the clinical or physiological states, while the estimation of $\alpha_{ik}$ in original model (4.2) only depends on the probe data corresponding to the gene $k$.

## 4.2 Bayesian kernel models

In Chapter 3, we propose a novel Bayesian kernel model whose performance on some benchmark datasets are among the best of tested nonlinear models, including the state-of-art support vector machines. One of major advantages of our Bayesian kernel models over SVMs is that it is very flexible to be extended in various ways. In terms of applications, future works include the extension to survival analysis, multinomial choice models, multivariate regression, time series analysis etc. In the following, we discuss some possible future works in term of model structures.

### 4.2.1 Additive models

When the responses have linear relationship with some of covariates while having nonlinear relationship with others. For example, the responses are generated by the following function

$$y_i(X_1, X_2) = 0.5x_{1i} + 0.8\sin(x_{2i}) + \epsilon_i.$$

An additive model with a linear term and a nonlinear kernel term would be more appropriate, i.e.,

$$f(\mathbf{x}) = \mathbf{x}^T\beta + \sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}). \tag{4.3}$$

It is straightforward to implement the additive model by extending the MCMC procedures of Bayesian kernel models in Chapter 3. Let

$$f_1(\mathbf{x}) = \mathbf{x}^T\beta,$$

$$f_2(\mathbf{x}) = \sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}).$$

We can develop a two-step posterior simulation algorithm by Gibbs sampling, which is also call backfitting (Tibshirani and Hastie, 1998). In the first step, we draw the samples of conditional posterior for the parameters in the linear model given the sparse kernel model $f_2(\mathbf{x})$ at their "current" imputed values in the MCMC. Given the prior for linear regression coefficients $\beta$ as:

$$\beta \sim N(\beta|0, \Sigma_\beta),$$

where $\Sigma_\beta$ could be a spherical matrix $\sigma_\beta^2 I_p$ or diagonal matrix $diag(\sigma_1^2, ..., \sigma_p^2)$. The conditional posterior distribution given $f_2$ and other hyperparameters is as follows:

$$\beta|D, f_2, \Sigma_\beta, \sigma^2 \sim N(\beta|\widehat{\beta}, V_\beta^{-1}),$$

where

$$\widehat{\beta} = \frac{1}{\sigma^2} V_\beta^{-1} X(\mathbf{y} - f_2),$$

and

$$V_\beta = \Sigma_\beta^{-1} + \frac{1}{\sigma^2} X X^T.$$

In the second step, we draw samples of the conditional posterior for the parameters in the kernel model, given the linear model $f_1(\mathbf{x})$ at their "current" imputed values in the MCMC. It can be done easily by replacing $\mathbf{y}$ with $\mathbf{y} - f_1$ in the previous analysis shown in Section 3.

### 4.2.2 Tree models

Tree based models date back to Morgan and Sonquist (1963) and Morgan and Messenger (1973). Breiman *et al.* (1984) popularized it and propose a new binary recursive partitioning algorithm, regression and classification trees (CART). CART partition the predictors space into a finite number of homogeneous set, and then fit a simple model in each one in the regression or classification framework. It is a simple yet powerful method. In this subsection, we suggest how some ideas from kernel models can be extended to the tree based models.

Classic CART is based on recursive partition of the original coordinate. The splits at each node are restricted to be of the form $X_j \leq t$. Though it works great on many cases, it does have limitations on some situations. For example, in the XOR problem, a simple nonlinear transformation will be a better choice for the recursive partition than the original coordinate. We may extend the tree models in the following ways:

- A straightforward idea is to partition the predictors space not only on the original coordinate $X_1, X_2, ..., X_p$, but also on the nonlinear basis functions and its interaction $\phi(X_i), \phi(X_i, X_j), ....$ Due to the extremely large number of such

basis functions, an ideal choice is the kernel function we use in the kernel models. We may simply add the kernel matrix $K_{n\times n}$ to the original design matrix, so that the final design matrix for the tree model becomes a $(p+n)\times n$ matrix

$$X^{new} = \left[ \begin{array}{c} X_{p\times n} \\ K_{n\times n} \end{array} \right].$$

- Another idea is to allow the nonlinear discrimination splitting rule at each node. Breiman *et al.* (1984) had already considered linear discrimination splitting rule by allowing linear combination of predictors. Thus the splitting rule is the form of $\sum \beta_j X_j \leq t$. Based on similar idea, Utgoff (1988) proposed perceptron trees, and Strömberg *et al.* (1991) (and Sankar and Mammone (1993)) proposed neural trees. To allow nonlinear combination of predictors, Guo and Gelfand (1992) included feed-forward neural networks in the CART. Similarly, we can consider a CART in which some of splitting rules are based on the kernel models, i.e., $\sum_j w_j k(\mathbf{x}_j, \mathbf{x}) \leq t$, while some of them are still based on the original predictors. One potential benefit of nonlinear discrimination splitting rules is that it could create a simple tree with only a few nodes, in the case that original CART requires a large number of nodes to solve a complex problem. Of course, there is a trade-off between the power of splitting rules within a tree and the complexity of resulting tree. Thus it raises the problem of choosing the optimal tree, which is more complex than the classical CART.

Instead of choosing the optimal tree, we can develop both tree models and kernel models in the Bayesian framework, and then apply Bayesian model averaging. Key references for Bayesian tree models are Chipman *et al.* (1998), Denison *et al.* (1998), Holmes *et al.* (1999) and Chipman *et al.* (2002). A Bayesian tree model can be described as a sampling distribution of the responses $Y$ conditional on predictor $X$ and tree model $T$. The tree model $T$ can be explicitly defined as an array of parameters

which can be inferred from the data in the Bayesian framework. Then a Bayesian tree model with nonlinear discrimination splitting rule will include three kinds of parameters: the parameters specifying tree structure, the parameters specifying kernel models served as splitting rule and the parameters specifying sampling distribution at the leaf nodes. Future works will include the prior specification on the parameters and the MCMC algorithm to draw samples in the both model and parameter space.

### 4.2.3 Mixtures of experts

Mixtures of experts (ME) and hierarchical mixtures of experts (HME) are conditional mixtures of regression/classification models. The key references are Jacobs *et al.* (1991), Jordan and Jacobs (1992) and Jordan and Jacobs (1994). They can be regarded as generalizations to the standard mixture models. In mixtures of experts, the 'experts' are typically regression or classification models that are relatively simple, and 'experts' are combined by a set of local mixing weights called the 'gating networks' which can depend on the predictors. In hierarchical mixtures of experts, this mixing process is done recursively. Consequently, the HME model has a tree-structure and can be considered as a tree-based methods with soft probabilistic split.

ME is based on the assumption that the process generating the data can be decomposed into a set of sub-processes defined on the regions of predictor space. For each data item, the conditional probability distribution of $y_i$ given $\mathbf{x}_i$ and model parameters $\Theta$ is given by the mixture density

$$P(y_i|\mathbf{x}_i, \Theta) = \sum_l \pi_l P(y_i|\mathbf{x}_i, U_l)$$

where

- $l$ is a label representing region $l$. It is chosen from a multinomial distribution with covariate-dependent probability $\pi_l = P(l|\mathbf{x}_i, V)$, where $V = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_L]$

is the matrix of 'gating networks' parameters for the multinomial distribution.

- $P(y_i|\mathbf{x}_i, U_l)$ is the conditional probability distribution of $y_i$ given $\mathbf{x}_i$ in each region $l$, where $U_l$ is the corresponding 'experts' parameters.

In classical ME and HME, both 'gating networks' and 'experts' are a linear model or generalized linear model of predictors. For example, in a ME regression model, the output of 'gating networks' is

$$\pi_l = \frac{\exp(\eta_l)}{\sum_1^L \exp(\eta_k)},$$

$$\eta_l = \mathbf{x}^T \mathbf{v}_l.$$

and the 'experts' have the form

$$P(y_i|\mathbf{x}_i, \beta_l, \sigma_l^2) = N(y_i|\mathbf{x}_i^T \beta_l, \sigma_l^2),$$

where $(\beta_l, \sigma_l^2)$ is the regression parameters for 'expert' $l$.

While the classical MEs try to model nonlinear relationship using a mixture of simple linear models, the nonlinear kernel models can be integrated into ME to produce simpler model, in the case that original ME requires a large number of 'experts' to solve a complex problem. One possible way is to introduce nonlinear functions in some of 'gating networks', i.e., $\eta_l = \sum_j w_j k(\mathbf{x}_j, \mathbf{x})$, which is similar to the nonlinear discrimination splitting rule in tree-based models. A more interpretable way is to introduce nonlinear functions in some of 'experts'. For example, a simple ME with two experts has the form

$$P(y_i|\mathbf{x}_i, \Theta) = \pi N(y_i|\mathbf{x}_i^T \beta, \sigma^2) + (1-\pi) N(y_i| \sum_j w_j k(\mathbf{x}_j, \mathbf{x}_i), \sigma^2),$$

whicg is supposed to work well when some of data has linear relationship with responeses while others have nonlinear relationship.

By introducing latent variable $z_i \in \{0, 1\}$, where $\pi = P(z_i = 1)$, we can rewrite the model as

$$y_i = z_i \mathbf{x}_i^T \beta + (1 - z_i) \sum_j w_j k(\mathbf{x}_j, \mathbf{x}_i) + \epsilon_i,$$

$$\epsilon_i \sim N(\epsilon_i | 0, \sigma^2),$$

and implement it via MCMC.

### 4.2.4 Robust regression

The Bayesian kernel regression model with Gaussian noise could lead to poor results when there are outliers in the data. In general, we can obtain a robust regression model by replacing the Gaussian noise with a heavy-tailed noise distribution, such as Student-t distribution, which allows for the possibility of extreme observations. While it is difficult to implement this idea in support vector regression models, it is quite straightforward in the Bayesian kernel models. Based on the fact that a Student-t distribution can be interpreted as a mixture of normal distributions with a common mean and variances distributed as inverse gamma, we can write a robust kernel model as

$$y_i = \sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}_i) + \epsilon_i,$$

$$\epsilon_i \sim N(\epsilon_i | 0, V_i \sigma^2),$$

$$V_i^{-2} \sim Ga\left(V_i^{-2} \middle| \frac{k}{2}, \frac{k}{2}\right),$$

where $k$ is the degrees of freedom for implied Student-t distribution.

The conditional posterior for $V_i$ is simply an inverse gamma posterior

$$p(V_i^{-2} | y_i, X, \mathbf{w}_j, \sigma^2) = Ga(V_i^{-2} | \frac{k+1}{2}, \frac{k\sigma^2 + (y_i - \sum_{j=1}^{n} w_j k(\mathbf{x}_j, \mathbf{x}_i))^2}{2\sigma^2}),$$

then this component can be plugged into the MCMC procedures in Chapter 3 to construct the MCMC implementation of robust kernel models.

### 4.2.5 Nonlinear factor regression models

The "large $p$ small $n$ problem" refer to a class of ill-posed statistical problems in which the sample size $n$ is substantially smaller than the number of predictor variables $p$. One context is the regression models with large sets of higher-order interactions between predictor variables, in which the predictors are discretized versions of continuous functions or values of processes, such as spectral profiles or time series. Another key motivating application comes from the molecular phenotyping using large-scale gene expression as predictors of clinical or physiological states. In this scenario, the number of genes is the several or ten of thousands, whereas the samples of data are fewer than one hundreds due to the huge cost and effort required by current microarray technologies. West (2000) and West (2003) proposed Bayesian factor regression models that have shown great results on those applications. In this subsection, we discuss the ideas that extend them to nonlinear models using kernel models.

A simple nonlinear factor regression model can be obtained by building the kernel models on the orthogonal SVD factor $\mathbf{f}_i$ instead of original input vector $\mathbf{x}_i = AD\mathbf{f}_i$, i.e.,

$$\begin{aligned} \mathbf{x}_i &= AD\mathbf{f}_i, \\ y_i &= \sum_{j=1}^{n} \omega_j k(\mathbf{f}_i, \mathbf{f}_j) + \epsilon_i, \\ \epsilon_i &\sim N(\epsilon_i|0, \sigma^2), \end{aligned} \tag{4.4}$$

where

- $A$ is the $p \times n$ SVD loading matrix with orthonormal columns so that $A^T A = I$;

- $D = diag(d_1, d_2, ..., d_n)$ is the diagonal matrix of positive singular values, ordered as $d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$; and

- $F$ is the $n \times n$ orthogonal matrix with $F^T F = FF^T = I$. The $i^{th}$ column of $F$ is denoted as $i^{th}$ factor $\mathbf{f}_i$ that is a linear combination of original $p$ predictor variables.

The detailed discussion of principal component regression models can be found in West (2000). The implementation of (4.4) is straightforward by simply replacing $\mathbf{x}_i$ with $\mathbf{f}_i$ in our Bayesian kernel models.

West (2003) proposed the factor regression models by replacing the principal component analysis with more general Bayesian factor analysis models. The corresponding nonlinear model can be written as

$$\begin{aligned} \mathbf{x}_i &= B\mathbf{f}_i + \mathbf{v}_i, \\ y_i &= \sum_{j=1}^{n} \omega_j k(\mathbf{f}_i, \mathbf{f}_j) + \epsilon_i, \\ \mathbf{v}_i &\sim N(\mathbf{v}_i|0, \Psi), \\ \epsilon_i &\sim N(\epsilon_i|0, \sigma^2), \end{aligned} \tag{4.5}$$

where

- $B$ is the $p \times k$ factor loading matrix, whose element is $b_{ij}$( gene $i$,factor $j$) with the prior

$$b_{ij} \sim N(b_{ij}|0, 1).$$

- $\mathbf{f}_i$ is the $k-$dimensional latent factors for case $i$ with the prior

$$\mathbf{f}_i \sim N(\mathbf{f}_i|0, D),$$

where $D$ is diagonal, which implies that $k$-components in $\mathbf{f}_i$ are, a priori, uncorrelated.

- $\mathbf{v}_i$ is the independent $p-$dimensional normal vectors with

$$v_i \sim N(v_i|0, \Psi),$$

  where the covariance matrix $\Psi$ is diagonal. It implies that the input variable $\mathbf{x}_i$ is uncorrelated conditional on the latent factors.

Under the factor models, the variance-covariance structure of input data is constrained as

$$\Omega = BB^T + \Psi.$$

which implies that the factor models attempt to explain the variance-covariance structure in the observed data by putting all the variance unique to each coordinate in the diagonal covariance matrix $\Psi$, and putting all the correlation structure into the factor loading matrix $B$, which is a linear mapping of $k-$dimensional latent factor space to $p-$dimensional data space.

Note that in Bayesian factor regression models, we model the joint distribution of covariates and responses $P(Y, X|\theta, \phi)$, where $\theta$ and $\phi$ are the parameters in the factor model and regression model respectively, and $\theta$ and $\phi$ are dependent. Thus the latent factor $\mathbf{f}_i$ is not only based on covariates $X$, but also based the responses $Y$, which share the similar ideas as Partial Least Squares (PLS). The implementation of (4.5) via MCMC requires the posterior sampling of $\mathbf{f}_i$, which is quite different from the implementation of linear Bayesian factor regression models illustrated in West (2003).

In the factor kernel regression model (4.5), the conditional posterior distribution for $\mathbf{f}_i$ is

$$p(\mathbf{f}_i|\mathbf{x}_i, \mathbf{y}, B, \Psi, \mathbf{f}_{-i}, W, D) \propto p(\mathbf{x}_i|\mathbf{f}_i, B, \Psi)p(\mathbf{y}|\mathbf{f}_i, \mathbf{f}_{-i}, W)p(\mathbf{f}_i|\mathbf{f}_{-i}, D), \tag{4.6}$$

where

$$p(\mathbf{y}|\mathbf{f}_i, \mathbf{f}_{-i}, W) = \prod_{i=1}^{n} N\left(y_i \middle| \sum_{j=1}^{n} \omega_j k(\mathbf{f}_i, \mathbf{f}_j), 1\right),$$

and

$$p(\mathbf{f}_i|\mathbf{f}_{-i}, D) = p(\mathbf{f}_i|D).$$

It is difficult for us to draw the samples of $\mathbf{f}_i$ directly from its conditional posterior distribution. Instead, we use the Metropolis algorithm to draw the samples of $\mathbf{f}_i$. Denoting $\mathbf{f}_i^{(t)}$ as the $t$th sample of $\mathbf{f}_i$, a appropriate choice of proposal is the random walk, i.e.,

$$\mathbf{f}_i^* \sim N(\mathbf{f}_i^*|\mathbf{f}_i^{(t)}, \Delta),$$

where $\Delta$ could be simple spherical matrix $\sigma^2 I_{k\times k}$ or be adjusted adaptively during tune-in steps.

In some situations, the second term of (4.6), $p(\mathbf{y}|\mathbf{f}_i, \mathbf{f}_{-i}, W)$, plays a much less important role than the first term $p(x_i|\mathbf{f}_i, B, \Psi)$. This motivates us to use the conditional posterior distribution of $\mathbf{f}_i$ in the standard factor model for importance sampling or as an independent proposal:

$$\mathbf{f}_i^* \sim N(\mathbf{f}_i^*|\widehat{\mathbf{f}}_i, V_{\mathbf{f}_i}^{-1}),$$

$$\widehat{\mathbf{f}}_i = B^T\Psi^{-1}\mathbf{x}_i,$$

$$V_{\mathbf{f}_i} = B^T\Psi^{-1}B + D^{-1}.$$

In the situation when the second term $p(\mathbf{y}|\mathbf{f}_i, \mathbf{f}_{-i}, W)$ does play an important role, we can use a mixture of independent proposal and random walk proposal:

$$\mathbf{f}_i^* \sim \pi N(\mathbf{f}_i^*|\mathbf{f}_i^{(t)}, \Delta) + (1-\pi)N(\mathbf{f}_i^*|\widehat{\mathbf{f}}_i, V_{\mathbf{f}_i}^{-1}),$$

where $\pi$ is a specified value dependent on how well the independent proposal performs.

# References


Aizerman, M., Braverman, E. and Rozonoer, L. (1964) Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, **25,** 821 – 837.

Akaike, H. (1974) A new look at statistical model identification. *IEEE Transactions on Automatic Control*, **19,** 716–723.

Albert, James H. and Chib, Siddhartha (1993) Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, **88,** no. 422, 669–679.

Antoniak, C. (1974) Mixtures of Dirichlet processes with applications to Bayesian non-parametric problems. *Annals of Statistics*, **2,** 1152–1174.

Aronszajn, N. (1950) Theory of reproducing kernels. *Transactions of the American Mathematical Society*, **68,** 337 – 404.

Basilevsky, A. (1983) *Applied matrix algebra*. New York: North-Holland.

Bennett, K. and Demiriz, A. (1998). Semi-Supervised Support Vector Machines, in Advances in Neural Information Processing Systems 11.

Bertsekas, D. P. (1995) *Nonlinear Programming*. Belmont, MA: Athena Scientific.

Bishop, C. M. and Tipping, M. E. (2003) Bayesian regression and classification. *Advances in Learning Theory: Methods, Models and Applications*, **190,** 267285.

Blum, Avrim and Mitchell, Thomas (1998) Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, San Francisco*, pp. 92–100. Morgan Kaufmann.

Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992) A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, Pittsburgh, PA* (ed. D. Haussler), pp. 144–152. ACM Press.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984) *Classification and Regression Trees*. Chapman and Hall, New York.

Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Jr, M. and Haussler, D. (2000) Knowledge-based analysis of microarray gene expression data by using suport vector machines. In *Proc. Natl. Acad. Sci.*, vol. 97, pp. 262–267.

Brown, Patrick O. and Botstein, David (1999). Exploring the new world of the genome with DNA microarrays. *Nature Genetics Supplement*, 21: p. 33-37.

Burges, Christopher J. C. (1998) A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, **2,** no. 2, 121–167.

Carlisle, A.J., Prabhu, V.V., Elkahloun, A., Hudson, J., Trent, J.M., Linehan, W.M., Williams, E.D., Emmert-Buck, M.R., Liotta, L.A., Munson, P.J. and Krizman, D.B. (2000). Development of a prostate cDNA microarray and statistical gene expression analysis package. *Mol Carcinog*, May;28(1):12-22.

Chen, Y., Dougherty, E.R. and Bittner, M.L. (1997). Ratio-based decisions and the quantitative analysis of cDNA microarray images. *J. Biomed. Optics*, 2(4): p. 364-374.

Chernick, M. R., Murthy, V. K. and Nealy, C. D. (1985) Application of bootstrap and other resampling techniques: evaluation of classifier performance. *Pattern Recognition Letters*, **3,** 167–178.

Cheung, V.G., Morley, M., Aguilar, F., Massimi, A., Kucherlapati, R. and Childs, G. (1999). Making and reading microarrays. *Nature Genetics Supplement*, 21: p. 15-19.

Chib, Siddhartha (1995) Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, **90,** no. 432, 1313–1321.

Chipman, Hugh A., George, Edward I. and McCulloch, Robert E. (1998) Bayesian CART model search. *Journal of the American Statistical Association*, **93,** 935–947.

Chipman, Hugh A., George, Edward I. and McCulloch, Robert E. (2002) Bayesian treed models. *Machine Learning*, **48,** no. 1/3, 299–320.

Consortium, I.H.G.S. (2001) Initial sequencing and analysis of the human genome. *Nature*, **409,** 860–921.

Denison, D.G.T., Mallick, B.K. and Smith, A.F.M. (1998) A Bayesian CART algorithm. *Biometrika*, **85,** 363–377.

Duane, S., Kennedy, A. D., Pendleton, B. J and Roweth, Roweth (1987) Hybrid Monte Carlo. *Physics Leters* B, **195,** no. 2, 216–222.

Duggan, D.J. (1999). Expression profiling using cDNA microarrays. *Nature Genetics Supplement*, 21: p. 10.

Efron, B. (1982) *The Jackknife, the Bootstrap and Other Resampling Plans*. Philadelphia: SIAM.

Eisen (1999). SCANALYZE, University of Berkely, CA: Berkely.

Escobar, M.D. (1994) Estimating Normal means with a Dirichlet process prior. *Journal of the American Statistical Association*, **89,** 268–277.

Escobar, M.D. and West, M. (1995) Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, **90,** 577–588.

Fahlman, S. E. and Lebiere, C. (1990) The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2. Proceedings of the 1989 Conference, San Mateo, CA* (ed. D. S. Touretzky), pp. 524–532. Morgan Kaufmann.

Ferguson, T.S. (1973) A Bayesian analysis of some non-parametric problems. *Annals of Statistics*, **1,** 209–230.

Ferguson, T.S. (1974) Prior distributions on spaces of probability measures. *Annals of Statistics*, **2,** 615–629.

Ferguson, T.S. (1983) Bayesian density estimation by mixtures of Normal distributions. *Recent advances in Statistics (M. Rizvi, J. Rustagi, and D. Siegmund, eds.); Academic Press, New York*, 287–302.

Frean, M. (1990) The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, **2,** no. 2, 198–209.

Friedman, J. H. (1991) Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, **19,** 1–141.

Girosi, Federico, Jones, Michael and Poggio, Tomaso (1995) Regularization theory and neural networks architectures. *Neural Computation*, **7,** no. 2, 219–269.

Golub, G. H. and Van Loan, C. F. (1989) *Matrix Computations*, Second edn. Baltimore: Johns Hopkins University Press.

Green, P. J. (1995) Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, **82,** 711–732; Workshop on Model Criticism of Highly Structured Stochastic Systems, Weisbaden, Sept 1994.

Green, P.J. and Richardson, S. (1998) Modelling heterogeneity with and without the Dirichlet process. *Scandinavian Journal of Statistics*, **28,** 335–375.

Guo, H. and Gelfand, S. B. (1992) Classification trees with neural network feature extraction. *IEEE Transactions on Neural Networks.*, **3,** 923–933.

Hastie, T., Tibshirani, R. and Friedman, J. (2001) *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer Verlag.

Hastie, T. J. and Tibshirani, Robert J. (1990) *Generalized Additive Models*. London: Chapman & Hall.

Higdon, Dave (2000). Space and Space-Time Modeling Using Process Convolutions, ISDS discussion papers, Duke University.

Holmes, C. C. and Mallick, B. K. (1998) Bayesian radial basis functions of variable dimension. *Neural Computation*, **10,** no. 5, 1217–1233.

Holmes, C.C., Denison, D.G.T. and Mallick, B.K. (1999) Bayesian partitioning for classification and regression. *Technical Report, Imperial College London*.

Hughes, Timothy R., Mao, Mao, Jones, Allan R., Burchard, Julja, Marton, Matthew J., Karen W. Shannon2, Michael Ziman Steven M. Lefkowitz2, Schelter, Janell M., Meyer, Michael R., Kobayashi, Sumire, Davis, Colleen, Dai, Hongyue, He, Yudong D., Stephaniants, Sergey B., Cavet, Guy, Walker, Wynn L., West, Anne, Coffey, Ernest, Shoemaker, Daniel D., Stoughton, Roland, Blanchard, Alan P., Friend, Stephen H. and Linsley, Peter S. (2001). Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer. *Nature Biotechnology*. 19: p. 342-347.

Insua, R. and Muller, D. (1998). Feedforward neural networks for nonparametric regression, Technical Report 98–02, Institute of Statistics and Decision Sciences, Duke University. Available at http://www.stat.duke.edu.

Irizarry, RA, Hobbs, B, Collin, F, Beazer-Barclay, YD, Antonellis, KJ, Scherf, U and Speed, TP (2003a) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, **4,** no. 2, 249–264.

Irizarry, Rafael. A., Bolstad, Benjamin M., Collin, Francois, Cope, Leslie M., Hobbs, Bridget and Speed, Terence P. (2003b) Summaries of affymetrix genechip probe level data. *Nucleic Acids Research*, **31,** no. 4.

Jackson, J. E. (1991) *A User's Guide to Principal Components*. New York: Wiley.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J. and Hinton, G. E. (1991) Adaptive mixtures of local experts. *Neural Computation*, **3,** 79–87.

Joachims, Thorsten (1999) Transductive inference for text classification using support vector machines. In *Proc. 16th International Conf. on Machine Learning*, pp. 200–209. Morgan Kaufmann, San Francisco, CA.

Johnson, Valen E. and Albert, James H. (1999) *Ordinal Data Modeling*. Berlin: Springer-Verlag.

Jolliffe, I. T. (1986) *Principal Component Analysis*. New York: Springer.

Jordan, M. I. and Jacobs, R. A. (1992) Hierarchies of adaptive experts. In *Advances in Neural Information Processing Systems 4* (eds J. E. Moody, S. J. Hanson and R. P. Lippmann), pp. 985–992. San Mateo, California: Morgan Kaufmann.

Jordan, M. I. and Jacobs, R. A. (1994) Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, **6,** 181–214.

Kohonen, T. (1995) *Self-Organizing Maps*. Berlin: Springer.

Le Cun, Y., Denker, J. S. and Solla, S. A. (1990) Optimal brain damage. In *Advances in Neural Information Processing Systems 2. Proceedings of the 1989 Conference, San Mateo, CA* (ed. D. S. Touretzky), pp. 598–605. Morgan Kaufmann.

Lee, Herbert, Higdon, David, Bi, Zhuoxin, Ferreira, Marco and West, Mike (2002) Markov random field models for high-dimensional parameters in simulations of fluid flow in porous media. *Technometrics*, **44,** no. 3, 230–241.

Li, C. and Wong, W. H. (2001a) Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection. PNAS, **98,** no. 1, 31–36.

Li, C. and Wong, W. H. (2001b) Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error application. *Genome Biology*, **2,** no. 8, 1–11.

Lipshutz, R.J., Fodor, S.P., Gingeras, T.R. and Lockhart, D.J. (1999). High density synthetic oligonucleotide arrays. *Nature Genetics Supplement*, 21: p. 20-24.

Lockhart, D., Dong, H., Byrne, M., Follettie, M., Gallo, M., Chee, M., Mittmann, M., Wang, C., Kobayashi, M., Horton, H. and Brown, E. (1996). Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*. 1996 Dec;14(13):1675-80.

Lowe, D. (1995) Radial basis function networks. In *The Handbook of Brain Theory and Neural Networks* (ed. Michael A. Arbib), pp. 779–783. Cambridge, Massachusetts: MIT Press.

MacKay, David J. C. (1994) Bayesian methods for backpropagation networks. In *Models of Neural Networks III* (eds E. Dormany, J. L. van Hemmen and K. Schulten). New York: Springer-Verlag.

MacKay, David J. C. (1995) Bayesian non-linear modelling for the 1993 energy prediction competition. In *Maximum Entropy and Bayesian Methods, Santa Babara 1993* (ed. G. Heidbreder). Dordrecht: Kluwer.

Mercer, J. (1909) Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London* A, **209,** 415–446.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B. and Müller, K.-R. (1999) Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX* (eds Y.-H. Hu, J. Larsen, E. Wilson and S. Douglas), pp. 41–48. IEEE.

Moody, J. and Darken, C. (1989) Fast learning in networks of locally-tuned processing units. *Neural Computation*, **1,** no. 2, 281–294.

Morgan, J. N. and Messenger, R. C. (1973) *THAID*: a sequential search program for the analysis of nominal scale dependent variables. Survey Research Center, Institute for Social Research, University of Michigan.

Morgan, J. N. and Sonquist, J. A. (1963) Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, **58,** 415–434.

Naef, Felix, Lim, Daniel A., Patil, Nila and Magnasco, Marcelo O. (2001). From features to expression: High-density oligonucleotide array analysis revisited. *Tech Report* 1, 1-9.

Neal, R. M. (1998). Regression and classification using Gaussian process priors

(with discussion), in J. M. Bernardo, et al (editors) Bayesian Statistics 6, Oxford University Press, pp. 475-501.

Neal, Radford M. and Hinton, Geoffrey (1995) Bayesian learning for neural networks. Ph.D. Thesis.

Platt, J. (1991) A resource-allocating network for function interpolation. *Neural Computation*, **3,** no. 2, 213–225.

Poggio, T. (1975) On optimal nonlinear associative recall. *Biological Cybernetics*, **19,** 201–209.

Press, W. H. *et al.* (1992) Numerical recipes in C (second edition). *Cambridge University Press.*

Rasmussen, C. (1996) Evaluation of Gaussian processes and other methods for non-linear regression. Ph.D. Thesis. Department of Computer Science, University of Toronto. ftp://ftp.cs.toronto.edu/pub/carl/thesis.ps.gz.

Rasmussen, Carl Edward (1996) Bayesian regression using Gaussian process priors. In *Meet. American Statistical Association.* Invited talk.

Rätsch, G., Onoda, T. and Müller, K.-R. (2001) Soft margins for AdaBoost. *Machine Learning*, **42,** no. 3, 287–320; also NeuroCOLT Technical Report NC-TR-1998-021.

Richardson, S. and Green, P. (1997) On bayesian analysis of mixtures with an unknown number of components. *Jrnl. Royal Stat. Soc.*, **59,** 731–792.

Ripley, Brian D. (1996) *Pattern Recognition and Neural Networks.* Cambridge, United Kingdom: Cambridge University Press.

Rissanen, J. (1987) Stochastic complexity. *Journal of the Royal Statistical Society* B, **49,** no. 3, 223–239.

Sankar, A. and Mammone, R. J. (1993) Growing and pruning neural tree networks. *IEEE Trans. on Computers*, **42,** 291–299.

Schena, M., Shalon, D., Davis, R.W. and Brown, P.O. (1995). Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science.* 1995 Oct 20;270(5235):467-70.

Schena, M., Shalon, D., Heller, R., Chai, A., Brown, P.O. and Davis, R.W. (1996).

Parallel human genome analysis: Microarray-based expression monitoring of 1000 genes. PNAS, 1996. 93(20): p. 10614-10619.

Schoelkopf, B., Smola, A. J. and Mueller, K.-R (1997) Kernel principal component analysis. *Lecture Notes in Computer Science*, **1327,** 583–596.

Schölkopf, B. (1997) *Support Vector Learning*. Munich: R. Oldenbourg Verlag.

Schölkopf, B., Burges, C. J. C. and Smola, A. J. (1999) *Advances in Kernel Methods — Support Vector Learning*. Cambridge, MA: MIT Press.

Schölkopf, B. and Smola, A. J. (2002) *Learning with Kernels*. Cambridge, MA: MIT Press.

Schwarz, G. (1978) Estimating the dimension of a model. *Annals of Statistics*, **6,** 461–464.

Shalon, S. Smith D. and Brown, P. (1996). A DNA microarray system for analyzing complex DNA samples using two-color fluorescent probe hybridization. *Genome Researc*h, 6: p. 639.

Shawe-Taylor, John and Cristianini, Nello (2004) *Kernel Methods for Pattern Analysis*. Cambridge, MA: Cambridge University Press.

Southern, E. M. (1998). Method and apparatus for analysing polynucleotide sequences. UK Patent Application, Isis Innovation. 03 May, 1988.

Southern, K. Mir E. and Shchepinov, M. (1999). Molecular interactions on microarrays. *Nature Genetics Supplement*, 21: p. 5-9.

Spang, R, Zuzan, H., West, M., Nevins, J., Blanchette, C. and Marks, J.R. (2002) Prediction and uncertainty in the analysis of gene expression profiles. *Silico Biol.*, **2,** no. 3, 369–381.

Stone, M. (1974) Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society series* B, **36,** 111–147.

Strömberg, J. E., Zrida, J. and Isaksson, A. (1991) Neural trees—using neural nets in a tree classifier structure. In *IEEE International Conference on Acoustics, Speech and Signal Processing (Toronto, 1991), Long Beach, CA*, pp. 137–140. IEEE Press.

Suykens, J. A. K. and Vandewalle, J. (1999) Least squares support vector machine classifiers. *Neural Process. Lett.*, **9,** no. 3, 293–300.

Szummer, Martin and Jaakkola, Tommi (2001) Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, vol. 14.

Tibshirani, Robert and Hastie, Trevor (1998) Bayesian backfitting. Technical Report. Department of Statistics, Stanford University.

Tipping, Michael E. (2001) Sparse Bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, **1,** 211–244.

Tipping, Michael E. and Bishop, Christopher M. (1997) Probabilistic principal component analysis. Technical Report NCRG/97/010. Neural Computing Research Group, Aston University, Aston St, Birmingham, B4 7ET, UK.

Utgoff, P. E. (1988) Perceptron trees: a case study in hybrid concept representations. In *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence, St Paul, San Mateo, CA* (eds R. G. Smith and T. M. Mitchell), pp. 601–606.

Vapnik, V. (1995) *The Nature of Statistical Learning Theory.* New York: Springer Verlag.

Vapnik, V. (1998) *Statistical Learning Theory.* New York: Wiley.

Venter, J. and *et al.* (2001) The sequence of the human genome. *Science*, **291,** 1304–51.

Wahba, Grace (1990) *Spline models for observational data.* SIAM [Society for Industrial and Applied Mathematics].

Wahba, Grace (1999) Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV. *Advances in kernel methods: support vector learning*, 69–88.

Watson, A., Mazumder, A., Stewart, M. and Balasubramanian, S. (1998). Technology for microarray analysis of gene expression. *Current Opinion in Biotechnology*, 9: p. 609-614.

West, M. (1984). Outlier models and prior distributions in Bayesian linear regression, Journal of the Royal Statistical Society (Ser. B), vol. 46 (1984), pp. 431-439 .

West, M. (1992) Hyperparameter estimation in Dirichlet process mixture models. *ISDS discussion papers, Duke University*, **92-03**.

West, M. (2000) Bayesian regression analysis in the "Large p, Small n" paradigm. *ISDS discussion papers, Duke University*, **00-22**.

West, M. (2003) Bayesian factor regression models in the "Large p, Small n" paradigm. *In: Bayesian Statistics 5; Oxford University Press, Oxford*.

West, M., Müller, P. and Escobar, M. (1994) Hierarchical priors and mixture models, with application in regression and density estimation. *In: Aspects of Uncertainty: A tribute to D. V. Lindley, (A.F.M. Smith and P. Freeman editors); Wiley publisher*, 63–386.

West, M., Nevins, J., Marks, J.R., Spang, R and Zuzan, H. (2002) DNA microarray data analysis and regression modeling for genetic expression profiling. *Methods for gene expression analysis, ed. G.Parmigiani*.

Williams, C. K. I. (1998) Prediction with gaussian processes: From linear regression to linear prediction and beyond. In *Learning and Inference in Graphical Models* (ed. M. I. Jordan). Kluwer. To appear. Also: Technical Report NCRG/97/012, Aston University.

Williams, C. K. I. and Rasmussen, C. E. (1996) Gaussian processes for regression. In *Advances in Neural Information Processing Systems* (eds D. S. Touretzky, M. C. Mozer and M. E. Hasselmo), vol. 8. Cambridge, MA: MIT Press.

Williams, Christopher K. I. and Barber, David (1998) Bayesian classification with gaussian processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, **20,** no. 12, 1342–1351.

Wodicka, L., Dong, H., Mittmann, M., Ho, M.H. and Lockhart, D.J. (1997). Genome-wide expression monitoring in Saccharomyces cerevisiae. *Nature Biotechnology*. 15: p. 1359-1367.

Zellner, A. (1986) On assessing prior distributions and bayesian regression analysis with g-prior distributions. *Bayesian Inference and Decision Techniques,P. Goel and A. Zellner, Eds.*, 233–243.

Zhu, J. and Hastie, T. (2001). Kernel logistic regression and the import vector machine. In *Proc. of Neural Information Processing Systems*. Submitted.

Zhu, X., Ghahramani, Z. and Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In ICML, volume 20, 2003.