AN INVESTIGATION INTO THE USE OF GAUSSIAN PROCESSES FOR THE ANALYSIS OF MICROARRAY DATA

SIAH KENG BOON

NATIONAL UNIVERSITY OF SINGAPORE

2004

AN INVESTIGATION INTO THE USE OF GAUSSIAN PROCESSES FOR THE ANALYSIS OF MICROARRAY DATA

SIAH KENG BOON

(B.Eng.(Hons.), NUS)

A DISSERTATION SUBMITTED FOR

THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

To my family and friends

Acknowledgements

I wish to express my deepest gratitude and appreciation to my two supervisors, Associate Professor Chong Jin Ong and Associate Professor S. Sathiya Keerthi for their instructive guidance and constant personal encouragement during the period of my research.

I gratefully acknowledge the financial support provided by the National University of Singapore through Research Scholarship that enabled my studies.

My appreciations also goes to Mr. Yee Choon Seng, Mrs. Ooi, Ms. Tshin, Madam Hamidah and Mr. Zhang for the giving numerous facility support in the laboratory, which help to complete the project smoothly.

I would like to thank my family and friends for their love and support through my life.

I am also fortunate to have met many talented research fellows in the Control Laboratory. I am sincerely grateful for their friendship, especially to Chu Wei, Lim Boon Leong, Duan Kaibo, Manojit Chattopadhyay, Qian Lin, and Liu Zheng.

I also want to thank Shevade Shirish Krishnaji and Radford Neal for their help in my research project.

Table of contents

Summary

Microarray technologies are powerful tools which allow us to quickly observe the changes at the differential expression levels of the entire complement of the genome (cDNA) under different induced conditions. Under these different conditions, it is believed that important information and clues to their biological functions can be found.

In the past decade, numerous microarray experiments were performed. However, due to the large amount of data, it is difficult to analyze the data manually. Recognizing this problem, some researchers have applied machine learning techniques to help them understand the data (Alizadeh et al., 2000; Alon et al., 1999; Brown et al., 2000; Golub et al., 1999; Hvidsten et al., 2001). Most of them tried to do classification on these data, in order to differentiate two different possible classes, e.g tumor and non-tumor or two different types of tumors. Generally, the main characteristic of the microarray data is that it has large number of genes but rather small number of examples. This means that it is possible to have a lot of redundant and irrelevant genes in the dataset. Thus, it is useful to apply feature selection tools to select a set of useful genes, before feeding into a machine learning techniques. These two areas, i.e. gene microarray classification and feature selection, are the main tasks for this thesis.

We have applied Gaussian Processes with Monte Carlo Markov Chain (MCMC) treatment as classification tool, and Automated Relevance Determination (ARD) in Gaussian Processes as feature selection tool for the microarray data. Gaussian Processes with MCMC treatment is based on a Bayesian probabilistic framework to make prediction (Neal, 1997). It is a very powerful classifier and best suited for

problem with a small number of examples. However, the application of Bayesian modelling scheme into the interpretation of Microarray dataset is yet to be investigated.

In this thesis, we have used this machine learning to study the application of Gaussian Processes with MCMC treatment in four datasets, namely Breast cancer dataset, Colon cancer dataset, Leukaemia dataset and Ovarian cancer dataset.

It will be expensive to directly apply Gaussian Processes on the datasets. Thus, filter methods, namely Fisher Score and Information Gain, are used for the first level of feature selection process. Comparisons are done upon these two methods. We have found out that these two filter methods, generally, gave comparable results.

To estimate the quality of the selected feature, we use the technique of external cross-validation (Ambroise and McLachlan, 2002), which gives an unbiased average test accuracy. In this technique, the training data is split into different folds. The gene selection procedure is executed, each time using training dataset that excludes one fold. Testing will be done on the fold omitted. From this average test accuracy, the combination of filter methods and ARD feature selection methods gives results that are comparable to those in the literature (Shevade and Keerthi, 2002). Though it is expected that average test accuracy is higher than validation test accuracy, the average test accuracy obtained is also considerably good, particularly in Breast Cancer dataset and Colon Cancer Dataset.

List of Figures

List of Tables

Chapter 1

Introduction

In the recent years, biological data are being produced at a phenomenal rate. On average, the amount of data found in databases such as GenBank double in less than 2 years time (Luscombe et al., 2001). Besides, there are also many other projects, closely related to the gene expression studies and protein structure studies, that are adding numerous amount of information to the field. This surge in data has heightened the need to process them. As a result, computers have become an indispensable element to biological research. Since the advent of information age, computers are used to handle large quantities of data and investigate complex relations, which may be observed in the data. The combination of these two fields has given rise to a new field, Bioinformatics.

The pace of data collection has been once again speeded up with the arrival of DNA microarray technologies (Genetics, 1999). It is one of the new breakthroughs in experimental molecular biology. With thousands of gene expression processed in parallel, microarray techniques are producing huge amount of valuable data rapidly. The raw microarray data are images, which are then transformed to gene expression matrices or tables. These matrices have to be evaluated if further knowledge concerning the underlying biological process is to be extracted out. As the data is huge, studying the microarray data manually is not possible. Thus, to evaluate and classify the microarray data, different methods in machine learning are used, both supervised and unsupervised methods (Bronzma and Vilo, 2001).

In this thesis, the focus will be on a supervised method, i.e. the outputs of the training examples are known and the purpose is to predict the output of new example. In most of the cases, the outputs are either of two classes. Hence, the task is to classify a particular example of the microarray data, predicting it to be tumor or non-tumor (Colon Cancer dataset), or differentiating it between two different cancer types (Leukaemia dataset).

In most cases, the number of examples in a typical microarray data set is small. This is so because the cost of applying and evaluating different conditions and evaluating on the samples is relatively high. Yet, the data is very large due to the huge number of genes involved, ranging from a few thousands to hundreds of thousands. Thus, it is expected that most of the genes are irrelevant or redundant. Generally, these irrelevant and redundant features would not be helpful in the prediction process. In fact, there are many cases in which irrelevant and redundant features decrease the performance of the machine learning algorithm. Thus, feature selection tools are needed. It is hoped that by applying feature selection methods on microarray datasets, we are able to eliminate a substantial number of irrelevant and redundant features. This will improve the machine learning process as well as reduce the computational effort required. This is the motivation of the thesis.

1.1 Literature Review

There are several papers working on these two areas, i.e., gene microarray classification and feature selection in gene microarray datasets. Furey et al. (2000) have employed Support Vector Machines to classify three datasets, which are Colon Cancer, Leukaemia and Ovarian datasets. Brown et al. (2000) also applied Support Vector Machines in gene Microarray datasets. Even though the number of examples available is low, the authors are still able to obtain low testing errors. Thus, the method is popular. Besides Support Vector Machines, Li et al. (2001a) have combined a Genetic Algorithm and the k-Nearest Neighbor method to dis-

criminate between different classes of samples, while Bendor et al. (2000) used a Nearest Neighbor method with Pearson Correlation. Nguyen and Rocke (2002) used Logistic Discrimination and Quadratic Discriminant Analysis for predicting human tumor samples. Naive Bayes method (Keller et al., 2000) is also employed. Also, Dudoit and Speed (2000) employed a few methods, namely, Nearest Neighbor, Linear Discriminant Analysis, Classification tree with Boosting, and Bagging for gene expression classification. Meanwhile, Shevade and Keerthi (2002) have proposed a new and efficient algorithm based on Gauss-Seidel method to address gene expression classification problem. Recently, Long and Vega (2003) used Boosting methods to obtain cross validation estimates for the Microarray datasets.

For gene selection, Furey et al. (2000), Golub et al. (1999), Chow et al. (2001) and Slonim et al. (2000) made used of Fisher Score as the gene selection tool. Weston et al. (2000) also used information in kernel space of Support Vector Machines as a feature selection tool to compare with Fisher Score. Guyon et al. (2002) have introduced Recursive Feature Elimination based on Support Vector Machines to select relevant genes in gene expression data. Besides, Li et al. (2001b) has used Automated Relevance Determination (ARD) of Bayesian techniques to select relevant genes. Ben-Dor et al. (2000) have examined Mutual Information Score, as well as Threshold Number of Misclassification to find relevant features in gene microarray data.

In this thesis, we will investigate the usefulness of Gaussian Processes with Monte Carlo Markov Chain (MCMC) treatment, as the classifier for the microarray datasets. Gaussian Processes is an attractive method for several reasons. It is based on the Bayesian formulation and such a formulation is known to have good generalization property in many implementations. Instead of making point estimates (Li et al., 2001b), the method makes use of MCMC to sample on the evidence distribution. Besides this probabilistic treatment, it is also a well known fact that the method performs well with small number of examples and many features. We will also make use of the Automated Relevance Determination (ARD)

that is inherent in Gaussian Processes as the feature selection tool. We will discuss Gaussian Processes, MCMC and ARD in detail in Chapter 3.

As mentioned, we have used the probabilistic framework of Gaussian Processes, with the external cross validation methodology to predict as well as select relevant features. Based on the design, we can observe encouraging results. Except for the Leukaemia dataset, the results of the other three datasets show that the methodology perform competitively, compared with the results of Shevade and Keerthi (2002). However. we would like to emphasize that it is not the aim of this project to solve the problem and come out with a set of genes which we will claim as the cause of cancers. But, we would like to highlight a small number of genes which the Gaussian Processes methodology, have identified as the relevant genes in the data. We hope that this method can be a tool to help the biologists shorten their time to find out the genes responsible of certain disease. With the knowledge gain, they may apply necessary procedures or drugs to prevent the disease.

1.2 Organization of Thesis

The thesis is arranged in the following way. Chapter 2 describes feature selections methods for two classes problem, which are used in the thesis. An introduction of Gaussian Processes is given in Chapter 3. Gaussian Processes is a learning model for regression and classification tasks with a Bayesian framework. This framework is described in detail. Also, MCMC and ARD will be discussed in Chapter 3 as well. In Chapter 4, DNA microarray technology will be briefly mentioned, together with a short description of the four microarray datasets used throughout the thesis. Then, Chapter 5 describes some experiments to gain a better understanding on the Gaussian Processes. Chapter 6 concentrates on the methodology of the experiments. Then, the results and discussions based on the results obtained are covered in Chapter 7. Lastly, the conclusions of the thesis are in Chapter 8.

Chapter 2

Feature Selection

The microarray data are known to be of very high dimension (corresponding to the number of genes) and having few examples. Typically, the dimension is in the range of thousands or tens of thousands while the number of examples lies in the range of tens. Many of these features are redundant and irrelevant. Thus, it is a natural tactic to select a subset of features (i.e. the genes in this case) using feature selection.

Generally, feature selection is an essential step that removes irrelevant and redundant data. Feature selection methods can be categorized into two common approaches: the wrapper method and the filter method (Kohavi and John, 1996; Mark, 1999).

Wrapper method includes the machine-learning algorithm in evaluating the importance of features in predicting the outcome. This method is supported with the thought that bias of a particular induction algorithm should be taken into account when selecting features. A general wrapper architecture is described in Figure 2.1.

Wrapper method conducts a search on the number of input features. The techniques used can be forward selection (a search begins with the empty set of features and add in feature or a set of features with certain criteria), backward elimination (a search begins with the full set of features) or best first search (a search that allows backtracking along search path). Wrapper method will also

Figure 2.1: Architecture of wrapper method.

Figure 2.2: Architecture of filter method.

require feature evaluation function together with the learning algorithm to estimate the final accuracy of feature selection. The function can be a re-sampling technique such as k-fold cross validation or leave-one-out cross validation. Since wrapper method is tuned to interactions between an induction algorithm and its training data, it generally gives better results than filter method. However, to provide such an interaction, the learning algorithm is repeatedly called. This, in practice, may be too slow and computationally expensive for large datasets.

As for filter method, a heuristic is used, which is based on the characteristics of the data, to evaluate the usefulness of the features before evaluation with the learning algorithm. Independently on the learning algorithm, filter method is generally much faster than wrapper method. Thus, it is suitable for data of high dimensionality and many features. A general filter architecture is shown in Figure 2.2.

For most of the cases, filter method fails to recognize the correlation among the features. Filter method also requires the user to set a level of acceptance on

choosing the features to be selected, which requires experience of the user.

In this project, before applying Automated Relevance Determination (ARD), we use filter methods to reduce the number of features. This is mainly to avoid the huge dimension of the raw data to be fed into Gaussian Processes. The filter method that is being used here is Fisher Score and Information Gain. These two filter methods are widely used in Pattern Recognition for two-class problems. We will discuss these two filter methods in the next two sections.

2.1 Fisher Score

Fisher Score is an estimate of how informative a given data is, based on means and variances of two classes of the data. The method is only suitable for continuous values with two classes. The formulation of the Fisher Score is given as

$$FisherScore = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{2.1}$$

where μ_i and σ_i are the mean and standard deviation of data from class i.

The numerator of (2.1) is a measure of the distance between the two means of classes. Intuitively, if the two means are far apart, it is easier for the data to be recognized as two classes. Thus, if the numerator value is high, it means that the data is informative to differentiate the class.

However, just using the means are not sufficient. For example, a feature is not a strong feature if its means of the two classes are very much different and, at the same time, the variances of the two classes are also huge (i.e. the data of each class are widely spread). The situation will be even worse if the variance is so huge that there is much overlap region of the data. Thus, the denominator of (2.1) is introduced to overcome this situation.

Thus, Fisher Score is a measurement of the data in term of its distribution. The value of the score is high if the two means of the classes are very different and the data of each class are crowded near the mean of the data.

Fisher Score has been widely used in the microarrays data as the filter method for the reduction in the number of features (Golub et al., 1999; Weston et al., 2000; Furey et al., 2000; Chow et al., 2001; Slonim et al., 2000). Though the expression may differ from (2.1), the essential meaning is pretty similar. A summary of the expressions of Fisher Score used in the literature are given below:

1. Golub et al. (1999); Chow et al. (2001),

$$\frac{(\mu_1 - \mu_2)}{\sigma_1 + \sigma_2} \tag{2.2}$$

2. Furey et al. (2000),

$$|\frac{(\mu_1 - \mu_2)}{\sigma_1 + \sigma_2}| \tag{2.3}$$

3. Slonim et al. (2000),

$$\frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{2.4}$$

In the thesis, we will use (2.1).

2.2 Information Gain

Information gain is a filter method that is based on entropy (information theory). Entropy is a measurement of uncertainty in a system. The entropy of a distribution, x, is given by

$$H(x) = -\sum_x p(x) log_2(p(x)) \tag{2.5}$$

where $p(x)$ is the probability for x to happen.

However, entropy can also be used as a measure of independency. For this purpose, let x be the feature and t be the class. To check the entropy of a joint event when a feature occurs together with the class, the joint entropy is given as

$$H(x,t) = -\sum_x \sum_t p(x,t) log_2(p(x,t)) \tag{2.6}$$

where $p(x,t)$ is the joint probability for (x,t) to occur.

Equations (2.5) and (2.6) are used for computing the information gain of a feature and the class, which is given as

$$Information\ Gain = \sum_{x}\sum_{t} p(x,t) log_2 \frac{(p(x,t))}{p(x)p(t)} \tag{2.7}$$

Equation (2.7) is simply a measure of the reduction in uncertainty about a variable (for example, the feature in this case) due to the knowledge of another variable (the class in this case)(Duda et al., 2001). Thus, it is actually a measure of how much the distributions of the variables (the class and a feature) differ from statistical independency of these two variables.

The value of information gain is always greater than zero. From (2.7), it can be observed that if the class and the feature are independent, the value of mutual information is equal to zero. Hence, the greater the values of information gain, the higher the correlation between a feature and the class.

However, in most cases, the distributions of variables are not known. In order to use information gain, there is a need to discretize the gene expression values. In this project, we employed the Threshold Number of Misclassification (TNoM) method suggested by Ben-Dor et al. (2000) as the discretization method. It is based on the simple rule that uses the value, x,of expression level of a gene. The prediction class, t, is simply $sign(ax+b)$, where $a\epsilon(-1,+1)$. A straightforward approach is to find out the values of a and b that minimize the number of errors. Thus,

$$Err(a,b|x) = \sum_{i} 1\{t_i \neq sign(ax_i + b)\} \tag{2.8}$$

which means if the prediction and the label of an example are different, error is increased by one.

In this case, instead of using the (a,b) which give minimum errors of misclassification, the (a,b) that give the maximum value of information gain over the various possible discretizations is used. Once (a^*,b^*) are found after $2(n+1)$ possible search (where n is the number of possible values of $\mathbf{x}$), information gain

(2.7) can be found too. In short, TNoM (2.8) is used as a binning method before Equation (2.7) is applied.

2.3 Automatic Relevance Determination

We will discuss this feature selection method in Chapter 3, as it is closely related to Gaussian Processes.

Chapter 3

Gaussian Processes

This chapter presents a review on Gaussian Processes. It is first inspired by Neal's work (Neal, 1996) on priors for infinite networks. In spirit, Gaussian Processes models are equivalent to a Bayesian treatment of a certain class of multi-layer perceptron networks in the limit of infinitely large networks (i.e. with an infinite number of hidden nodes). This is shown experimentally by Neal (Neal, 1996). In Bayesian approach to neural networks, a prior on the weights in the network induces a prior distribution over functions. When the network becomes very large, the network weights are not represented explicitly. The priors of these weights are represented a simpler function in Gaussian Processes treatment. The mathematical development of this can be found in Williams (1997). Thus, Gaussian Processes achieves an efficient computation of prediction based on a stochastic process priors over functions.

The idea of having prior distribution over the infinite-dimensional space of possible functions have been known for many years. O'Hagan (O'Hagan, 1978) has used Gaussian priors over functions for his development. Generalized radial basis functions (Poggio and Girosi, 1989), ARMA models (Wahba, 1990) and variable metric kernel methods (Lowe, 1995) are all closely related to Gaussian Processes. The same model has long been used in spatial statistics known as "kriging" (Journel and Huijbregts, 1978; Cressie, 1991).

The work by Neal (Neal, 1996) has motivated examination of Gaussian Pro-

cesses for the high dimensional applications to which neural networks are typically applied, both on regression and classification problems (Williams and Rasmussen, 1996; Williams and Barber, 1998; Gibbs, 1997).

One of the common DNA microarray problems is the classification problem based on gene expressions (i.e. differential expression levels of the gene under different induced conditions). The task is to use the gene expression levels to classify groups to which an example belongs. There are a few classification methods that use Gaussian Processes: Laplace approximation (Williams and Barber, 1998), Monte Carlo method (Neal, 1997), variational techniques (Gibbs, 1997; Seeger, 1999) and mean field approximations (Opper and Winther, 1999). For this thesis, we will mainly focus on Neal's work which uses the technique of Monte Carlo Markov Chain (MCMC). Thus, in the following sections of this chapter, we will discuss the classification model based on MCMC Gaussian Processes.

3.1 Gaussian Processes Model for Classification

We now provide a review of the Gaussian Processes methodology and the associated nomenclature. See Neal (1996) for detailed discussion of this method.

We will use the following notations. x with d dimension is a training example. n is the total number of training examples. Let $\{x\}^n$ denotes the n example of inputs, x. The true label is denoted as $\widetilde{t}$. T denotes n numbers of training data, both inputs ($\{x\}^n$) and outputs (true labels). The Gaussian Processes for classification is based on the regression methodology. In the regression methodology, the predicting function of the regression is $y(x)$, which is also known as latent function. $Y = \{y(x^1), y(x^2), \ldots, y(x^n)\}$ denotes n numbers of latent functions. c_{ij} is the covariance function of input x^i and x^j. C is the covariance function matrix with elements c_{ij}. Let us denote the prediction of the input as $t(x)$, which only take two values (i.e. $+1$ or 0). x^* denotes the testing input. Accordingly, $t(x^*)$ is the predicting label of testing input.

Gaussian Processes is based on Baye's rule, for which a set of probabilistic

models of the data is specified. These models are used to make predictions. Let's denote an element of the set (or one model) by H, with a prior probability, $P(H)$. When the data, T, is observed, the likelihood of H is $P(T|H)$. By Baye's rule, the posterior probability of H is then given by

$$posterior \propto prior \times likelihood \tag{3.1}$$

$$P(H|T) \propto P(H) \times P(T|H) \tag{3.2}$$

The main idea of Gaussian Processes is to predict the output $y(x)$ for a given x. Each model, H, is related to $y(x)$ by $P(y(x)|H)$. Hence, if we have a set of probabilistic models, a combined prediction of the output $y(x)$ is

$$P(y(x)|T) = \sum_{allH} P(y(x)|H) \times P(H|T) \tag{3.3}$$

In the above, $y(x)$ is typically a regression output, i.e., $y(x)$ is a continuous output. This output is also known as latent function. For a classification problem, the above has to be expanded.

In a typical two-class classification problem, we assign a testing input, x^*, to class 1 if

$$P(t(x^*) = +1|T) \tag{3.4}$$

is greater than 0.5 and class 2 (i.e. true label is 0) otherwise.

We can find (3.4) by using a sigmoidal function over the latent function $y(x^*)$, through a transfer function, in the following manner

$$P(t(x^*) = +1|T)$$

$$= \int P(t(x^*) = +1|y(x^*))P(y(x^*)|T)\, dy(x^*) \tag{3.5}$$

where

$$P(t(x^*) = +1|y(x^*)) \tag{3.6}$$

is a sigmoidal function given by

$$(1 + exp(-y(x^*))^{-1} \tag{3.7}$$

From (3.5), to make prediction of the two-class classification problem, we need to find the probability distribution of a predictive function, $y(x^*)$, given all the training data, T, i.e.

$$P(y(x^*)|T) \tag{3.8}$$

In order to find (3.8), let us assume a Gaussian model as the prior of the latent functions, Y, i.e.,

$$P(Y) \propto \frac{1}{det(C)^{\frac{1}{2}}} exp(-\frac{1}{2} Y^t C^{-1} Y) \tag{3.9}$$

Meanwhile, the joint prior probability of $\{Y, y(x^*)\}$ is given by

$$P(y(x^*), Y) \propto \frac{1}{det(C_+)^{\frac{1}{2}}} exp(-\frac{1}{2} Y_+^t C_+^{-1} Y_+) \tag{3.10}$$

where C_+ is the covariance function matrix of $\{\{x\}^n, x^*\}$, and Y_+ is the vector $\{y(x^*), Y\}$. Generally, a covariance function is parameterized by a few parameters. These parameters are known as hyper-parameters, denoted by Θ.

Based on (3.9) and (3.10), the conditional probability $P(y(x^*)|Y, \Theta)$ is normally distributed with mean $k^t C^{-1} Y$ and variance $k_* - k^t C^{-1} k$, where k_* is the covariance function of x^*, while k is the covariance function vector of all inputs and x^*.

With that, $P(y(x^*)|T, \Theta)$ can be written as

$$P(y(x^*)|T, \Theta) = \int P(y(x^*)|Y, \Theta) P(Y, \Theta|T) dY \tag{3.11}$$

Thus if the distribution of Θ is known (3.11) can be re-expressed as

$$P(y(x^*)|T) = \int \int P(y(x^*)|Y, \Theta) P(Y, \Theta|T) dY d\Theta \tag{3.12}$$

The MCMC sampling process will give us iterative values for Y and Θ. Thus,

through these samplings, (3.12) can be re-expressed as

$$P(y(x^*)|T) = \frac{1}{R}\sum_{i=1}^{R} P(y(x^*)|Y_i, \Theta_i) \tag{3.13}$$

where R is the number of sampling iterations, while Y_i and Θ_i are the values of latent functions and hyper-parameters of ith iteration.

With that, (3.5) can be simplified as $P(t(x^*) = +1|T) =$

$$\frac{1}{R}\sum_{i=1}^{R} P(t(x^*) = +1|y^*(x^*))P(y(x^*)|Y_i, \Theta_i) \tag{3.14}$$

(3.14) will give the posterior probability of a testing input, x^*, to be class 1. We will use this posterior probability to make prediction in the two class classification.

From above, to obtain the possible values, a full MCMC treatment is applied over $P(Y, \Theta|T)$, which can be done over a two stage process, first by fixing Θ and finding $P(Y, \Theta|T)$ through sampling and followed by fixing Y and finding $P(Y, \Theta|T)$.

We will discuss the theory of MCMC in a more detail manner in Section 3.3 . Let us discuss how we apply MCMC first.

For fixed Θ, each of the n individual functions, $y(x)$, are updated sequently using Gibbs sampling, which we will discuss more about Gibbs sampling in section 3.3.1. The Gibbs sampling is done for a few scans as it takes a much shorter time to do such sampling. Also, the conditional probability is readily found. After employing Gibbs sampling, candidates of latent functions, let's say Y_i,can be obtained.

Secondly, for fixed Y, the hyper-parameters are updated using the Hybrid Monte Carlo method (HMC). We will discuss HMC in more details in Section 3.3.2.

This full Bayesian treatment with MCMC approach of defining a prior over the functions as well as parameters is a powerful approach. One need not be concerned about inaccuracy due to point estimation. Bayesian treatment will

consider as much location as possible, (theoretically, all locations) in the function space. Also, MCMC random walking (sampling) in the hyper-parameter space will be able to overcome the problem faced due to integration in the formulation (we will discuss this point in Section 3.3). Applying MCMC will be able to search the function as well as hyper-parameter spaces numerically. Most importantly, the two MCMC methods that are employed, i.e. Gibbs Sampling and HMC, are able to work under high dimensional function and hyper-parameter space respectively, unlike other sampling such as important sampling, which become difficult if the dimensions are too high.

3.2 Automatic Relevance Determination

In equations (3.13) and (3.14) we have grouped all parameters of the covariance function as Θ. It is possible to include feature selection ability through this parameterization. Suppose the covariance function is chosen as linear covariance function (which we use for all the applications and testing in this thesis), given by

$$c(x^i, x^j) = \sum_{l=1}^{d} \sigma_l^2 x_l^i x_l^j + \sigma_c^2 \tag{3.15}$$

where σ_l and σ_c are the Θ.

From (3.15), σ_l and σ_c, where l is from 1 to d (which are corresponding to 1 to d features), are the vectors required to parameterize covariance function. When the expected value of the σ_l is small, the model will ignore the input dimension. This implies that the feature corresponding to the σ_l is irrelevant. For relevant input dimension, corresponding σ_l will be big in value. In this case, the model will be hugely depending on the input dimension.

Since this type of build-in relation is found in Gaussian Processes in an "automated" form, and also, it is a measurement of relevance of feature, it is called as Automatic Relevance Determination (ARD)(Mackay, 1993; Neal, 1996).

Mathematically, the expected values of the hyper-parameters can be written

as

$$E(\Theta|T) = \int P(\Theta|T)\Theta d\Theta \tag{3.16}$$

which we can find by using

$$E(\Theta|T) = \int\int P(Y,\Theta|T)\Theta d\Theta dY \tag{3.17}$$

The process for obtaining (3.14) is actually obtaining (3.17) at the same time. By employing MCMC to sample on $P(Y,\Theta|T)$, the equation (3.17) becomes

$$E(\Theta|T) = \frac{1}{R}\sum_{i=1}^{R}\Theta_i \tag{3.18}$$

This shows that the ARD values , i.e. Θ, which obtained based on MCMC method, is statistically meaningful. The two stage process is able to obtain the statistical expected value of these hyper-parameters.

3.3 Monte Carlo Markov Chain

Let us discuss Monte Carlo Markov Chain (MCMC) now as both the predictions of Gaussian Processes for Classification, (3.12) and ARD, (3.18) require the use of MCMC. We will only discuss it briefly. For details of MCMC, the theorems and proofs can be found in (Gilks et al., 1996; Neal, 1993; Tanner, 1996).

The two main uses of Monte Carlo, as a sampling method, are to generate samples from a given distribution $P(x)$ or to estimate expectation of a function under a given distribution $P(x)$ (Mackay, 1998). Suppose that we have a set of random variables $\theta_1, \theta_2, ..., \theta_n$, which may describe the characteristic of a model, taking the values as $q_1, q_2, ..., q_n$, the expectation of a function, f$(\theta_1, \theta_2, ..., \theta_n)$ with respect to the distribution over Θ space is

$$E(f) = \sum_{q_1} ... \sum_{q_n} f(q_1, q_2, ...q_n)P(\theta_1 = q_1, \theta_2 = q_2, ..., \theta_n = q_n) \tag{3.19}$$

which can be approximated as

$$E(f) \approx \frac{1}{R}\sum_{i=1}^{R} f(q_1^i, q_2^i, ...q_n^i) \tag{3.20}$$

where i is the i-th point (sampling) in the sample. The sampling approach is what has been applied in Section 3.1 (equation (3.14)) and 3.2 (equation (3.18)) .

As for Markov chain, it is specified by an initial probability distribution, $P^0(\Theta)$ and a transition probability $T(\Theta'; \Theta)$. The probability distribution of the parameter at i-th iteration of Markov chain can be expressed as

$$P^{i+1}(\Theta') = \int T(\Theta'; \Theta) P^i(\Theta) d\Theta \tag{3.21}$$

Note that Markov Chain is an iteration sampling process, as shown in (3.21). The iteration effect will be clearly seen when we evaluate posterior probability (3.14) and expected value of hyper-parameters (3.18), when we discuss some implementation issues in Chapter 5.

During the construction of the chain, two things have to be assured. Firstly, the desired distribution has to be the invariant distribution of the chain. A distribution $\pi(\Theta)$ is invariant if it satisfies the following relation

$$\pi(\Theta') = \int T(\Theta'; \Theta) \pi(\Theta) d\Theta \tag{3.22}$$

Secondly, the probability distribution, $P^i(\Theta)$, regardless the initial conditions, is ergodic, i.e.

$$\lim_{i \to \infty} P^i(\Theta) = \pi(\Theta) \tag{3.23}$$

Normally, it is more convenient and easier to construct transition probabilities which satisfy the detailed balance condition

$$T(\Theta'; \Theta)\pi(\Theta) = T(\Theta; \Theta')\pi(\Theta') \;\; \forall \Theta \, and \Theta' \tag{3.24}$$

If (3.24) holds, then it is sufficient to prove that a Markov chain simulation will converge to the desired distribution.

To generate points drawn from the distribution with reasonable efficiency, the sampling method must search for relevant regions, without bias. The combination of Markov chain and Monte Carlo, i.e. MCMC, has the ability to search a distribution where parameter of the framework will come to a correct distribution, (provided that the probability distribution satisfies Equation (3.24)), and being generated in the limit as the length of chain grows, a fundamental of Markov chain as explained above (Equation (3.23)). Nevertheless, it is not practical to generate an infinite length of iterations. We need to find out a suitable way to identify the on-set of the equilibrium state where the invariant distribution is reached. We will discuss this in Chapter 5.

Two methods in MCMC are used in this thesis, which are Gibbs sampling and Hybrid Monte Carlo. We will discuss these two sampling methods in the following sections.

3.3.1 Gibbs Sampling

Neal has employed Gibbs Sampling to find the candidates of function, $y(x)$. This sampling uses a conditional distribution to find the next candidate. Thus, if the distribution, from which samples are needed to be solved, having conditional distributions that can be easily formulated, or the values of the parameters can be easily sampled from, Gibbs sampling is prefered.

In Gaussian Processes, the conditional probability of the problem is to sample the function from the Gaussian conditional distribution. The conditional distribution for $y(x)$ given the other functions, is

$$p(y(x_i)|y(x_{\mathbf{n}-i})) \propto exp(-\frac{1}{2}Y^t C^{-1} Y) \tag{3.25}$$

where $y(x_{\mathbf{n}-i})$ denotes all the functions exclusive of function $y(x_i)$.

With (3.25), the selection of the next candidate is quite straight forward. In

a case of a problem where there are n functions, a single scan involves sampling one function at a time:

$$y(x_1)^{t+1} \sim p(y(x_1)|y(x_2)^t, y(x_3)^t, \ldots, y(x_n)^t) \tag{3.26}$$

$$y(x_2)^{t+1} \sim p(y(x_2)|y(x_1)^{t+1}, y(x_3)^t, \ldots, y(x_n)^t) \tag{3.27}$$

$$\vdots$$

$$y(x_n)^{t+1} \sim p(y(x_n)|y(x_2)^{t+1}, y(x_3)^{t+1}, \ldots, y(x_{n-1})^{t+1}) \tag{3.28}$$

It can be proven that a single scan with conditional probability is a transitional probability that satisfies detailed balance (3.24) (Neal, 1993).

By sampling based on (3.26) to (3.28), candidates of Y_i can be obtained.

3.3.2 Hybrid Monte Carlo

The Hybrid Monte Carlo (HMC) method is a MCMC method that reduces random walk behavior (Duan et al., 1987) by making use of the gradient information of the particular distribution. In Gaussian Processes, the gradient of the evidence for hyper-parameter (Θ) can be found as shown in (3.41). Thus, if the gradient of the distribution being investigated is available, HMC can be employed to speed up searching of the hyper-parameter space.

For many systems, the probability distribution can be written in the form

$$P(\Theta) = \frac{1}{Z} exp(-E(\Theta)) \tag{3.29}$$

where $E(\Theta)$ is the potential energy, Θ in this case is the displacements and Z is the normalization factor.

In HMC, an independent extra set of momentum variables, $\mathbf{p} = p_1, p_2, \ldots, p_{d+2}$, with independent and identically distributed (i.i.d.) standard Gaussian distribution is introduced. With the momentum variables, we can add a kinetic term

$K(\mathbf{p})$ to the potential term $E(\Theta)$ to produce a full Hamiltonian energy function on a phase-space

$$H(\Theta, \mathbf{p}) = E(\Theta) + K(\mathbf{p}) \tag{3.30}$$

where $K(\mathbf{p})$ is the kinetic energy given as

$$K(\mathbf{p}) = \frac{1}{2}\sum_{i}^{d+1} p_i^2 \tag{3.31}$$

Using linear covariance function (3.15), the number of p is $d+1$, where there are d of σ_ls as well as the α_c.

The invariant distribution in MCMC for this Hamiltonian is, thus,

$$\pi(\Theta, \mathbf{p}) \propto exp(-H(\Theta, \mathbf{p})) \tag{3.32}$$

The proof that (3.32) is an invariant distribution can be found in (Neal, 1993). The reason to introduce an extra set of kinetics term, though is burdensome due to enlarging dimensions, is to find a candidate that is far away from the current state by following the dynamics path in phase-space.

Another important ingredient for HMC is the step of leapfrog discretization, employed to simulate the Hamiltonian dynamics. Leapfrog discretization updates θ_i and p_i in three steps.

Firstly, it takes a half step for the momentum,

$$p_i^{(t+\frac{\epsilon}{2})} = p_i^{(t)} + \frac{\epsilon}{2}\frac{\delta}{\delta\theta_i}(E(\theta_i^{(t)})) \tag{3.33}$$

where ϵ is the time step of a leapfrog discretization.

Then, it takes a full step for the position

$$\theta_i^{(t+\partial)} = \theta_i^{(t)} + \partial p_i^{(t+\frac{\partial}{2})} \tag{3.34}$$

and lastly another half step for the momentum

$$p_i^{(t+\partial)} = p_i^{(t+\frac{\partial}{2})} + \frac{\partial}{2}\frac{\delta}{\delta\theta_i}(E(\theta_i^{(t+\partial)})) \tag{3.35}$$

With the leapfrog, we will be able to find a set of hyper-parameters candidates. The candidate will be accepted based on the probability of acceptance

$$min\{1, exp[-H(\Theta^{(t+\epsilon)}, \mathbf{p}^{(t+\epsilon))} - H(\Theta^{(t)}, \mathbf{p}^{(t)})]\} \tag{3.36}$$

In summary, the HMC algorithm operates as follows:

1. Randomly choose a direction, λ (either $+1$ or -1).
2. Starting the state, $(\Theta^0, \mathbf{p}^0)$, perform L leapfrog steps with step size $\lambda\epsilon$, resulting in the state $(\Theta^*, \mathbf{p}^*)$
3. Based on (3.36), decide the new state $(\Theta^0, \mathbf{p}^0)$ or $(\Theta^*, \mathbf{p}^*)$

There are a few variations of HMC (Horowitz, 1991; Neal, 1994). However, the main ideas are similar to the one described above. The difference is that those methods are to further speed up the process of searching in the phase-state.

From above, in order to employ HMC, we need to obtain the gradient of $P(Y, \Theta|T)$ with respect to Θ.

Let

$$\Psi(Y, \Theta) = \lg\left(P(Y, \Theta|T)\right) \tag{3.37}$$

Since $P(T|Y, \Theta) = P(T|Y)$ as T is independent of Θ given Y. This gives

$$\Psi(Y, \Theta) = \lg\frac{(P(T|Y)P(Y|\Theta)P(\Theta))}{P(T)} \tag{3.38}$$

By substituting (3.7), and (3.10) with some manipulations, we will get

$$\Psi(Y, \Theta) = T^t Y - \sum_{i=1}^{\mathbf{n}}(1+e^{y(x_i)}) - \frac{1}{2}Y^t C^{-1} Y - \frac{1}{2}\lg\|C\| - \frac{\mathbf{n}}{2}\lg 2\pi + \lg P(\Theta) - \lg P(T) \tag{3.39}$$

Now, reasoned by (Neal, 1996), we can assume a Gamma model as the prior of the hyper-parameters, which is

$$P(\Theta) \propto \Theta^{(a-1)} e^{(-b*\Theta)} \tag{3.40}$$

where a and b are two parameters to be tuned. We will discuss these two parameters in Chapter 5.

We need the derivative of (3.39) with respect to Θ in applying HMC. The derivative can be written as follows:

$$\frac{\partial \Psi(Y,\Theta)}{\partial \theta_i} = -\frac{1}{2} Y^t C^{-1} \frac{\partial C}{\partial \theta_i} C^{-1} Y - \frac{1}{2} tr(C^{-1} \frac{\partial C}{\partial \theta_i}) + \frac{\partial lgP(\Theta)}{\partial \theta_i} \tag{3.41}$$

where θ_i is one of the variables in Θ.

Comparing the two MCMC methods (Gibbs sampling and HMC method) that is to be used in the Gaussian Processes for a full MCMC treatment, a complete Gibbs sampling scan takes a shorter time than the HMC updating of the hyper-parameter (Neal, 1997). This is mainly because the sampling based on condition probability is readily available. However, HMC is required to solve an inverse matrix problem (refer to 3.41), which takes much longer time to solve. Therefore, it is more sensible to have a few Gibbs samplings scans for each HMC sampling in the program, as this probably makes the Markov chain mix faster.

Another issue is why we need two stages instead of one. Firstly, as mentioned above, a two-stage process is faster than one-stage process, if we use purely HMC sampling. This is mainly due to the matrix inversion. Secondly, the condition probability of all latent functions and hyper-parameter is not available. Thus, we are not able to use Gibbs sampling only. This explains why a two-stage process is more favorable.

Chapter 4

Microarrays

Recently developed methods for monitoring mRNA expression changes involve Microarray technologies. The technologies are powerful as they allow us to quickly observe the changes at the differential expression levels of the entire complement of the genome (cDNA) under different induced conditions. It is believed that under these different conditions, how a gene or a set of genes expressed will provide important information and clues to their biological functions.

Due to the large amount of data being produced, it is difficult to analyze the data manually. Recognizing this problem, researchers have applied machine learning techniques to help them interpret the data (Alizadeh et al., 2000; Alon et al., 1999; Brown et al., 2000; Golub et al., 1999; Hvidsten et al., 2001).

In this chapter, we will discuss the microarray datasets used in the experiments. The background of the microarray technologies will be described briefly, and followed by the details of the datasets.

4.1 DNA Microarrays

A gene is a segment of the deoxyribonucleic acid (DNA) and it codes for a particular protein. A DNA molecule is a double stranded polymer made up of nucleotides. Each nucleotide comprises a phosphate group, a deoxyribose sugar, and one of the four nitrogen bases. The four different bases are adenine (A), guanine (G), cy-

tosine (C), and thymine (T). The two stranded polymers are held by hydrogen bond between nitrogen bases. The bases occur in pairs, with G pairing with C, and A pairing with T. The particular order of the bases specify the exact genetics instructions required to create organism.

Within each DNA molecule contains many genes, which carry the information required for constructing proteins. The protein-coding instruction from the genes are transmitted indirectly through messenger ribonucleic acid (mRNA), a transient intermediary molecule that has a single stranded complementary copy of the base sequence in the DNA molecule, with the base uracil (U) replacing thymine. The process during which DNA is transcribed to mRNA is transcription. Then, it will be translated to protein through translation process. To date, attention is mainly on expression level at mRNA level. Microarray uses complementary DNA (cDNA), which is produced from mRNA through reverse-transcription, to understand the expression level of genes. Hybridization between these nucleic acids provides a core capability of molecular biology (Southern et al., 1999).

Base-pairing (that is, A-T and G-C for DNA while A-U and G-C for RNA as mentioned above) or hybridization is the underlining principle of DNA microarray.

A microarray is an orderly arrangement of samples. It provides a medium for matching known and unknown cDNA samples based on base-pairing rules and automating the process of identifying the unknowns. An microarray experiment can make use of common assay systems (e.i. Gene Chip) and can be created by hand or by robots that deposit the sample. In general, sample spot sizes in microarray are typically less than 200 microns in diameter and each microarray usually contains thousands of spots. Microarrays require specialized robots and imaging equipment that generally are not commercially available as a complete system.

DNA microarray, or DNA chips are fabricated by high-speed robots, generally on glass but sometimes on nylon substrates, for which probes with known identity are used to determine complementary binding, thus allowing massively parallel gene expression and gene discovery studies. An experiment with a single DNA

chip can provide researchers information on thousands of genes simultaneously.

There are different types of microarray technologies. We will discuss two of them (which are cDNA microarrays (DeRisi et al., 1997; Duggan et al., 1999; Schena et al., 1995) and high density oligonucleotide arrays (Lipshutz et al., 1999; Loackhart et al., 1996)), from which datasets used in the experiments are produced.

4.1.1 cDNA Microarrays

Fabrication of microarrays begins with choosing the probes[1] to be printed on the microarrays. The specific genes of interest will be obtained and amplified. Then, cDNA microarray are produced by spotting the amplified products onto the matrix. Following purification and quality control to remove unwanted salts and detergents, aliquots in term of nanoliters of purified products are printed on coated glass microscope slides (chips) . The printing is carried out by a computer controlled, high speed robot, with a "spotter" which is essentially a capillary tube.

The targets for the microarrays are obtained from reverse transcription of mRNA pools. However, a labelling is done on the total RNA to maximize the amount of message that can be obtained from a given amount of tissues. Frequently, Cye3-dUTP (green) and Cye5-dUTP (red) are used for this fluorescent labels, as they have relatively high incorporation efficiency with reverse transcriptase and widely separated in their excitation and emission spectra. For example, test targets are labelled with Cye5 and reference targets are labelled with Cye3.

The fluorescent targets, both test and reference, are pooled and allowed to competitively hybridize to the probes on the chips, under stringent conditions. During hybridization, the targets will interact with probes. If there is an interaction, single strands from targets and the single strand of the cDNA will combine and the targets will stick onto the immobilized probes, binding the targets to the microarrays. In other words, such binding means that the gene represented by the

[1] a "probe" is the tethered nucleic acid with known sequence, whereas a "target" is the free nucleic acid sample whose identity/abundance is being detected(Phimister, 1999)

probe is active, or expressed, in the sample. (This is why the final results (images) are actually the expression levels of the genes).

After hybridization process, the remaining solution that contains the targets will be discarded and microarrays will be gently rinsed. The chips will then be placed in a scanning confocal laser microscope. Laser excitation of the targets will give an emission with a characteristic spectra. Genes expressed in common by both test and reference targets will fluoresce with both colors, represented as yellow. Those represented only in the test targets fluoresce red, and those represented in the reference targets fluoresce green. The fluorescence intensity is reflecting the cDNA expression level from both targets. In this case, if green color is observed, we can claim that there is a reduction in expression and if red color is observed, it means there is an increase of expression. So, to show the relevant individual gene expression level, the ratio of test fluorescence intensity to reference (Cye5/Cye3) can be used.

With that, we will be able to obtain images from the scanner. These image data are then used for further analysis.

4.1.2 High Density Oligonucleotide Microarrays

High Density Oligonucleotide Microarrays synthesize oligonucleotides in situ as its probes, instead of obtaining the probes from natural organism like cDNA microarrays. That is the main difference between these two methods.

We will focus on the GeneChip®, a microarray chip produced by Affymetrix. The focus of the synthesis is light-directed. This involves two robust and unique techniques which are photolithography and solid phase DNA synthesis. The fabrication of the Genechip®is shown in Figure 4.1.

Photolithography allows the construction of arrays on rigid glass. Light is directed through a mask to de-protect and activate selected sites, and protected nucleotides couple to the activated sites. The process is repeated, activating different sets of sites and coupling different bases. In Figure 4.1, T is first nucleotides

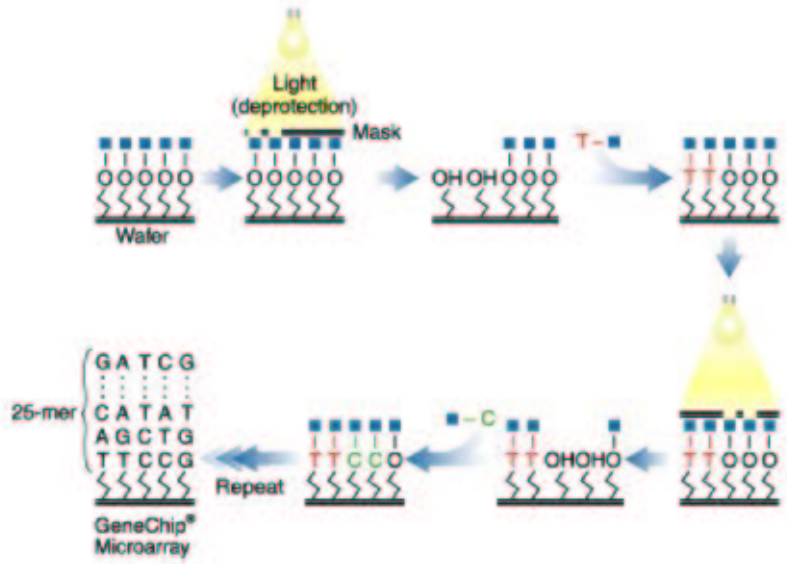

Figure 4.1: A unique combination of photolithography and combinatorial chemistry to manufacture GeneChip®microarrays in Affymetrix. Adapt from http://www.affymetrix.com/ technology/manufacturing/index.affx

introduced to the particular spot, and C is the second ones. In the end, this combination of methods will be able to synthesize a type of complementary probe, which consists of thousands of oligonucleotide probes for a single gene, at each spot on the chip.

The population of cells that one wishes to analyze will be treated similarly in that of cDNA microarray. The mRNA will be acquired from cells to be investigated. By reverse transcription, cDNA will be obtained. And, again, dye like Cye3-dUTP and Cye5-dUTP can be used to label the test targets and reference targets. These targets will be pooled and washed over the microarray. Whenever the probes interact with cDNA strand through hybridization, the cDNA will bind to the spot.

The microarrays is then scanned optically. And similar to cDNA microarrays, different fluorescent colors are the indications of the expression levels of genes. The relative expression levels between the test and reference targets can be found using the ratio of red to green.

With that, images can be found and further analysis can be based on the images obtained.

4.2 Normalization

Microarrays often consist of thousands of probes, and there is a chance that some of the probes will be poor representation of genes. Worse, these probes often play a major part in the increment or decrement of expression levels in microarrays. Take note that the poor representation is more severe in cDNA microarrays technology as High Density Oligonucleotide microarrays have much higher quality control during the fabrication of the chips. This contributes to the major cause of noise in the data. Besides, different technological factors (such as machine setting and saturation settings) and different experiment procedures (such as time taken for aliquot quantities) as well as human errors may cause variability in the data.

Much research and effort have been done to reduce the variability. Much of the literature addressing microarray normalization concerns cDNA microarray data, whereas only a few examples can be found for oligonucleotide microarrays. Workman et al. (2002) have done a short review on microarray normalization techniques.

Basically, the normalization methods are microarrays techniques dependent. Since our focus is on the classification and feature selection, we do not address normalization problem. However, it is observed that by applying normalization in two directions, i.e. normalization on all the gene expressions of each example, and then, another normalization process in done along the genes, gives good classification results. The reason is, though, not known to us.

4.3 Datasets

It is believed that gene expressions are essential for obtaining comprehensive pictures of the functioning of cell. Understanding the relative changes among the thousands of genes will be impossible without the careful use of some genome analysis. Microarrays have provided a tool to record the activities of these genes in parallel. It is hope that through this tool that the key players are able to be

recognized. For example, a design to identify new disease classes may also reveal clues about the basic biology of disorders from the microarrays data, and may suggest novel drug targets. This has motivated researchers to collect and study the microarrays data, based on this idea (Alizadeh et al., 2000; Alon et al., 1999; Golub et al., 1999; Perou et al., 1999; Pollack et al., 1999; Ross et al., 2000).

In this section, we will discuss four microarrays datasets that are used in this project.

4.3.1 Breast Cancer Dataset

This datasets was obtained by West et al. (2001). The dataset is based on breast tumor samples from the Duke Breast Cancer SPORE frozen tissue. There are two problems that are solved by (West et al., 2001). They are the classification of estrogen receptor (ER+/-ER) and classification of lymph node (LN+/LN-). In this project, we only focus on classifying between ER+ and ER-.

There are originally 49 tumor cases. However, West et al. (2001) have observed that in five tumor sample cases, the classification results are inconsistent. Therefore, we only used 44 examples in our experiment, consisting of 23 ER+ cases and 21 ER- cases. The number of gene representing each example is 7129. There is no missing data in this dataset.

The dataset is available at $http://mgm.duke.edu/genome/dna\_micro/work/$.

This dataset is obtained using Affymetrix GeneChip®. The targets and chips were hybridized at 45^0C for 16 hours. After that, the chips were scanned with GeneChip®scanner to obtained the signal.

4.3.2 Colon Cancer Dataset

Alon et al. (1999) have used Affymetrix GeneChip® to find the expression level of genes from colon adenocarcinoma specimens. From some of patients whom these specimens are found, normal colon tissue was also obtained. mRNA was then extracted from these tissue and hybridized to the GeneChip®. Images (i.e. data)

are obtained after the scanning. Thus, the task is to identify tumor from normal tissues.

The data consist of 22 normal tissues and 40 cancer tissues. For each tissue (example), there are around 6000 genes originally. Alon et al. (1999) has selected 2000 genes, where some genes are non-human genes. The dataset is available at $http://lara.enm.bris.ac.uk/colin/$.

4.3.3 Leukaemia Dataset

Leukaemia dataset from Golub et al. (1999) is to distinguish acute myeloid leukaemia (AML) from acute lymphoblastic leukaemia (ALL). The original datasets consist of both training and testing data, 27 ALL and 11 AML samples for training data and 20 ALL and 14 AML for testing data. In this project, we combine both data as the learning samples are not sufficient. In all samples, there are 7129 genes.

The dataset can be obtained from $http://lara.enm.bris.ac.uk/colin/$.

This dataset is obtained through High Density Oligonucleotides microarrays (Affymetrix). And the targets are prepared from bones marrow samples from patients with acute leukaemia.

4.3.4 Ovarian Cancer Dataset

Schena et al. (1995) have used cDNA microarrays techniques to obtained Ovarian cancer. Ovarian cancer dataset consists of 24 normal and 30 cancer tissues. There are 1536 genes for each sample.

Chapter 5

Implementation Issues of Gaussian Processes

Gaussian Processes are used as classifiers and feature selection tools in this thesis. Before applying on Microarray data, experiments were conducted to gain a better understanding on the method and its implementation.

In this chapter, we discuss the experiments done to identify the onset of equilibrium state for the MCMC sampling. After that, we will discuss the effect of gamma distribution of Θ (3.40).

5.1 Understanding on Gaussian Processes

5.1.1 Banana Dataset

To gain more insight into the implementation issues of Gaussian Processes, a simple dataset is used. For this purpose, we use the banana dataset. The dataset can be obtained at $http : //mlg.anu.edu.au/ \sim raetsch/data/$. This dataset contains 400 training and 4900 testing examples. There are only 2 features, with 1 output for each example. The output is binary while the input is continuous, with zero mean and standard deviation of one. There is no missing data in this dataset.

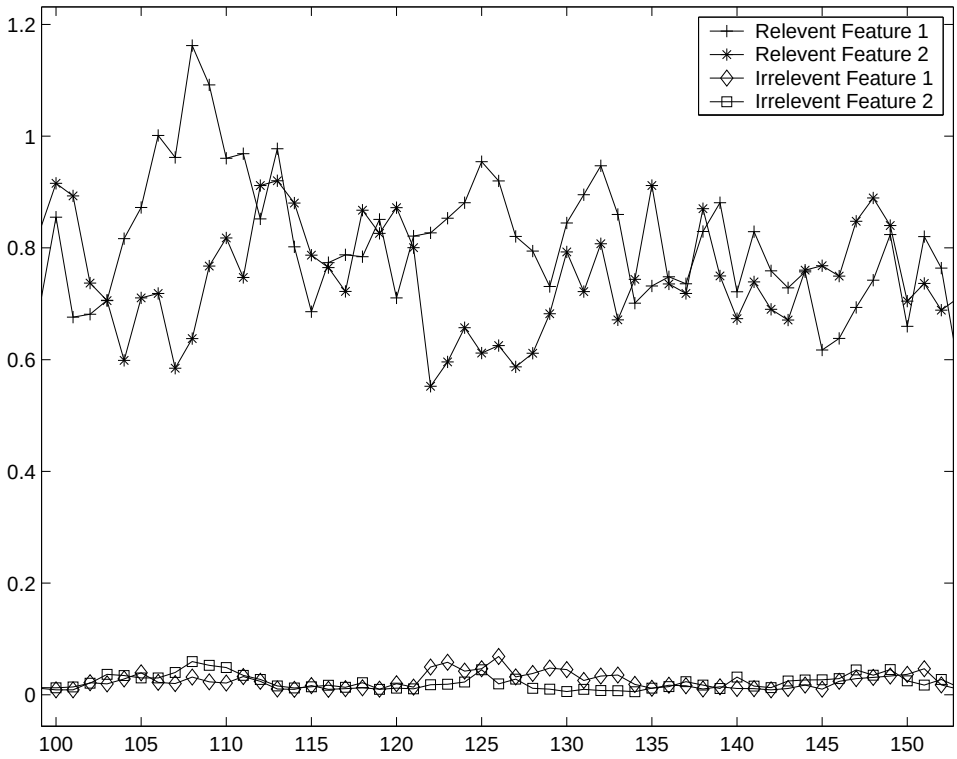

Figure 5.1: Values of Θ for original Banana datasets. Values on the vertical axis is the values of Θ while the horizontal axis is the iteration number of the values of Θ.

5.1.2 ARD at work

We created two irrelevant features using data from a zero mean Gaussian distribution with standard deviation of one. For the first run, we used the original training data with these 4 features for training and testing.

From Figure 5.1, we observe that the ARD values (which is the Θ variable) remained at a certain level. This shows that the ARD is able to identify the relevant features from irrelevant features, through iteration of samplings. Another point, observed from the results, is that there is a strong evidence of the state of invariant distribution. We will not be able to observe such a systematic characteristics if the invariant distribution is not reached.

Having observed the effect of irrelevant features, let us discuss the redundant features. In addition to the irrelevant features created above, we created two additional features that are exact duplicate of the original two features. These two features are therefore redundant. In total, six features are present for each

example.

For our study, we group the 400 training examples and 4900 testing examples into one dataset. Data from this dataset is then divided into three categories, based on the variance of the $P(t(x^*) = +1|T)$ over MCMC samples. The first category consists of examples that are easy to be classified. For each example in the first category, $P(t(x^*) = +1|T)$ over the MCMC samples is small. This means that the examples of the first category are clearly from one class (i.e. the $P(t(x^*) = +1|T)$ is close to 1 or 0). The second category consists of examples that are moderate in terms of the difficulty of classification and last category consists of examples that are difficult to classify. Similarly, the way to decide the difficulty depends on the variance of the $P(t(x^*) = +1|T)$ over the MCMC samples.100 examples are chosen for the dataset to form the training data. In this 100 training examples, equal number for each categories were present.

In Figure 5.2, the entire dataset is shown using the original two features. In Figure 5.3, 100 training examples are shown with "□" being of class 1 and "◇" being of class 2. From the figure, we can see the training examples chosen are representative of the whole dataset.

In addition to the 100 training examples, some examples are also selected for original dataset for detailed study. The location of these selected data are shown as "o" and "*" in Figure 5.4. Again, these selected examples have varying level of difficulties.

During the training process, the typical variation of Θ values over the MCMC samples are shown in Figure 5.5. We can observe that the values of the features, in particular, the relevant feature and its redundant feature interchange in values. For example, in Figure 5.6 (which is a zoom-in figure of Figure 5.5 around 170 to 190 iterations), before iteration 183, "redundant 2" feature is identified as the important feature while "relevant 2" feature is not. After iteration 183, "relevant 2" feature is identified as the important feature while "redundant 2" feature is not. This is expected. The Θ values of the features indicated the importance of the features in the evidence space. When Θ values of the relevant feature is

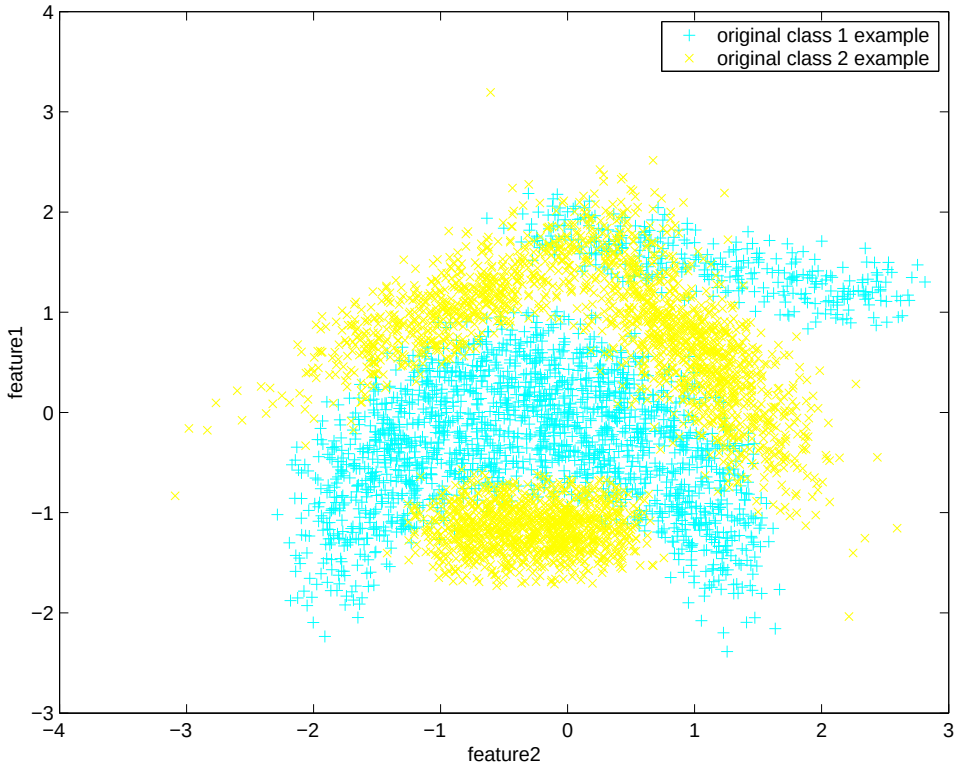

Figure 5.2: Location of all the original examples in the feature space.

high, Θ values of the redundant feature goes low and vice verse. As a result, the value of Θ interchange periodically, depending on the samples locations chosen by the MCMC procedure. In this case, the distribution of the hyper-parameter is like having four humps in the hyper space. This provides a strong evidence that equilibrium state is reached.

5.1.3 Equilibrium State

For six features for our banana dataset, equilibrium states can be monitored by examining pairs of Θ along the iteration. However, it will be almost impossible to observe such pattern with the data have than 10 features, let alone thousands of features as in Microarray datasets. The values of Θ are hard to monitor. Interchanging of the values of Θ is even harder to observe in such a high dimensional feature space. Therefore, we would like to propose another way to detect the onset of equilibrium state, based on the posterior probability, $P(t(x^*) = +1|T)$, as given by (3.14). Naturally, under the Bayesian framework, $P(t(x^*) = +1|T)$ will not be

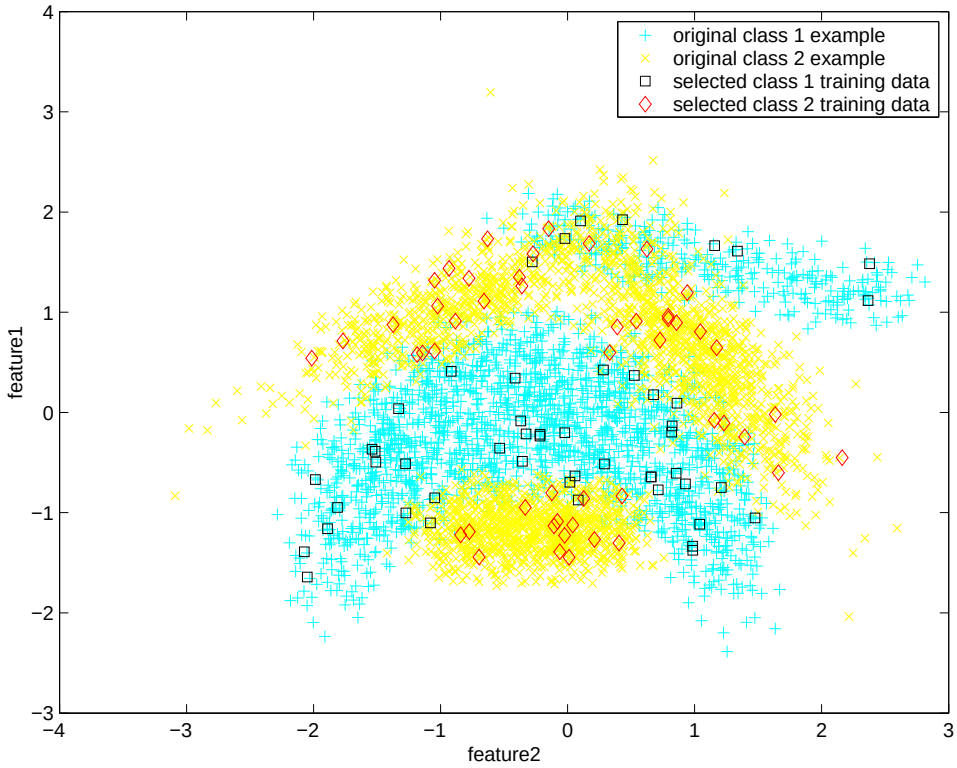

Figure 5.3: Location of all the training examples in the feature space.

the same over every iteration of the MCMC samples. However, it is expected that the median and/or the mean of $P(t(x^*) = +1|T)$, based on some fixed number of iterations, will be consistent once equilibrium state is reached. Example of this fact is shown in Figures 5.7 and 5.8, where the median of $P(t(x^*) = +1|T)$ over 25 MCMC samples (box plot) is drawn. Box plot is a box, which has lines at the lower quartile, median, and upper quartile values.

From Figure 5.5, we observe that the ARD values shows that equilibrium state is reached after about 80 iterations. And from Figure 5.7 and Figure 5.8, we also notice that the median of $P(t(x^*) = +1|T)$ becomes consistent from the 4th box onward (which is 75th iteration onward).

The box plots gives in Figure 5.7 and Figure 5.8 are drawn from examples that are easy to classify (referring to Figure 5.4). These two examples are situated in the midst of examples from the same class.

However, from Figure 5.9 and Figure 5.10 respectively, the examples lie at the boundary of the two classes. In this case, the median of 25 iterations of $P(t(x^*) =$

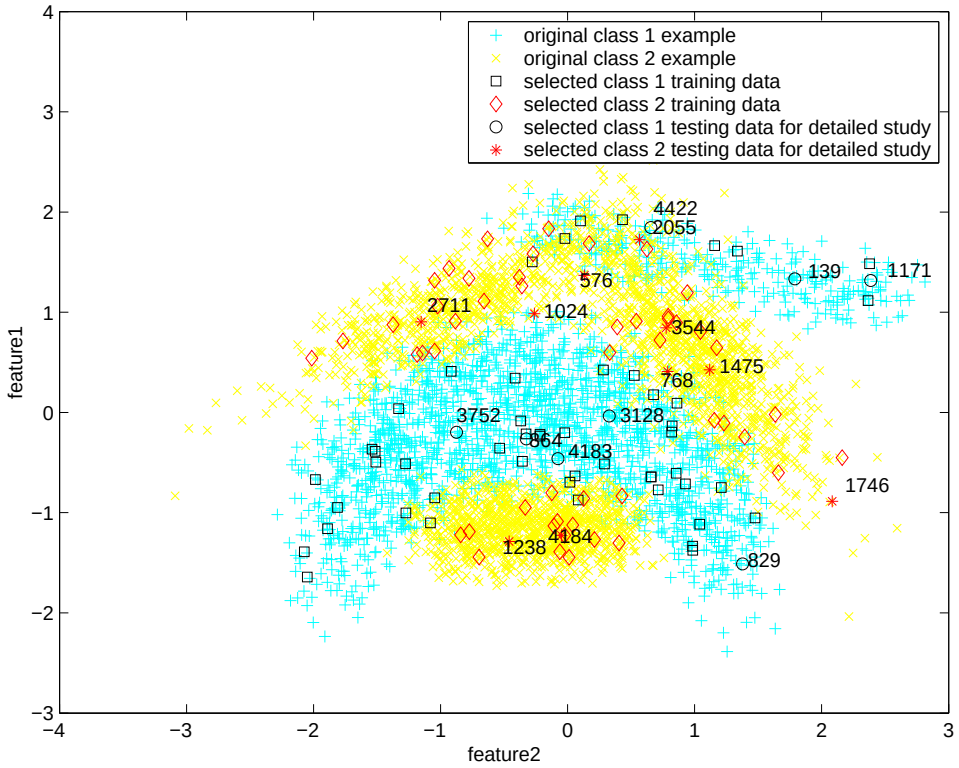

Figure 5.4: Location of all training and testing examples in the feature space.

$+1|T)$ is pretty consistent for the 4th box onwards, but the variance of the median is much higher. This is again expected under the Bayesian framework. The values of $P(t(x^*) = +1|T)$ vary to a greater extent when the example concerned is near the boundary of the two classes.

Thus, with the observations above, we would use the median of $P(t(x^*) = +1|T)$ over 25 samples as the criterion to determine the onset of equilibrium state. The main reason, as mentioned, is that the values of Θ will be hard to monitor and may take a longer time to observe equilibrium state due to its high dimensionality. Also, we will not know the interaction (correlation) of the features (genes) most of the time. The clear interaction which we created purposely in Banana dataset may not be seen. Moreover, there are too many ARD values to monitor.

Now, by using the median of iterations of $P(t(x^*) = +1|T)$, we will be able to identify the equilibrium state. However, there is still one question to answer. How many iterations should be used for the final prediction of $P(t(x^*) = +1|T)$ as given in (3.14)? From the experiment above, it seems to be safe to use the same

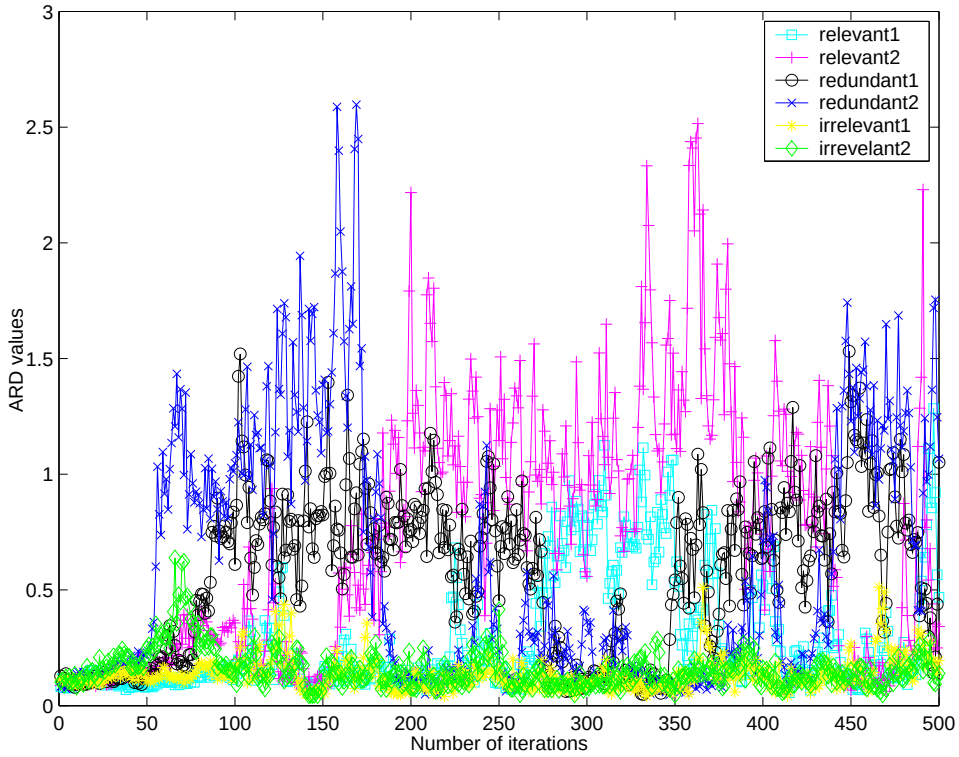

Figure 5.5: Values of Θ for Banana datasets, with redundant features.

number of iterations of which the samplings reach the equilibrium state. We can lengthen the run to increase the "safety factor".

All the box plots of the 18 testing examples can be found in Appendix A.

5.1.4 Effect of Gamma Distribution

From equation (3.40), gamma distribution is used as the prior distribution for each hyper-parameter (3.40). However, we have faced some problems in deciding the values of these parameters, i.e. a and b (3.40).

Based on our experiments, the "wrong" choice of prior distribution can stop the MCMC sampling prematurely. Till now, we are still not sure of the main cause of this problem. However, it is observed that some choices of the prior distribution will cause the HMC to reject many candidates, resulting in the large values of Θ.

The "wrong" prior distribution may also fail to find values of Θ giving a sign of equilibrium state (refer to Figure 5.11). There is no clear crossing between relevant

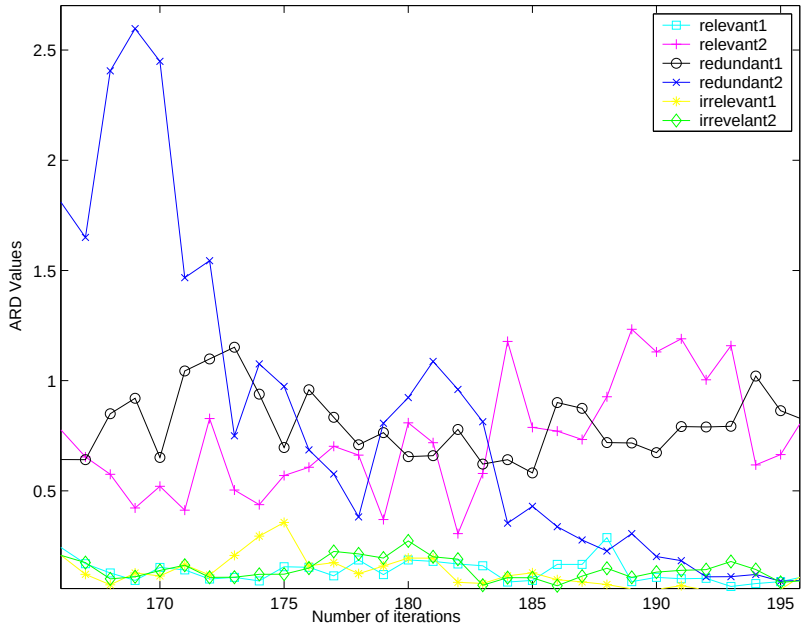

Figure 5.6: Values of Θ for Banana datasets, with redundant features.

and redundant features. When we repeat the banana experiment, for some prior distributions, we can't observe the presence of equilibrium state even after 10000 iterations. Also, none of the median of $P(t(x^*) = +1|T)$ has stabilized.

This is something that is still not being understood fully. What we have done to overcome the problem is simply to identify a prior distribution which avoid such scenario.

5.1.5 Summary

We have proposed a way to determine the onset of equilibrium state in MCMC samplings. This criterion is used for all other experiments in the subsequent data of this thesis.

The median of the posterior probability of an arbitrary iteration will be found for each example, using box plot. The point at which the median of the predicting probability remain consistent (± 0.15) will be identified as the point where the samplings reach equilibrium state.

Besides, it is also a must to make sure that majority of the examples gives a

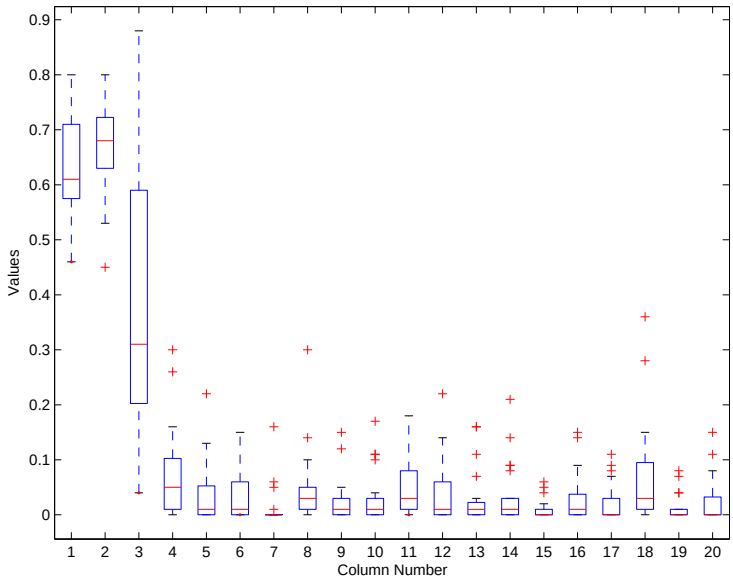

Figure 5.7: Box plot of testing example 4184 along iteration of MCMC samplings. Values on the vertical axis is the values of Θ while each column represents 25 iterations of the values of Θ.

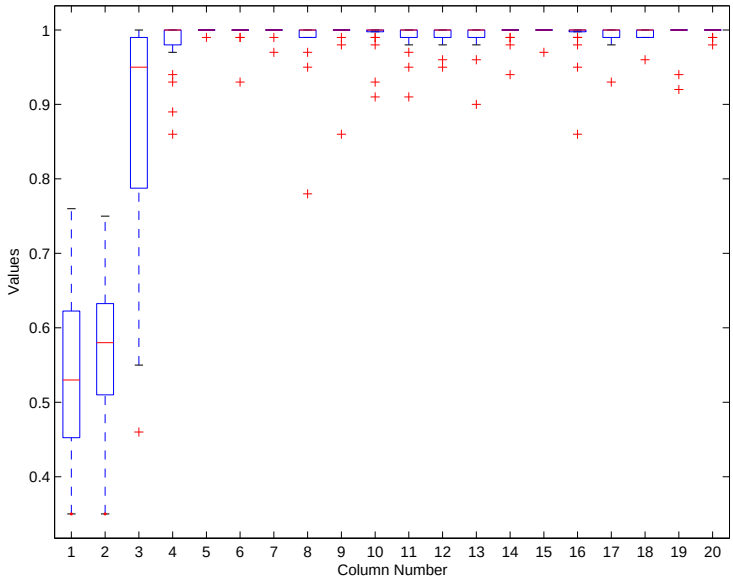

Figure 5.8: Box plot of testing example 864 along iteration of MCMC samplings. Values on the vertical axis is the values of Θ while each column represents 25 iterations of the values of Θ.

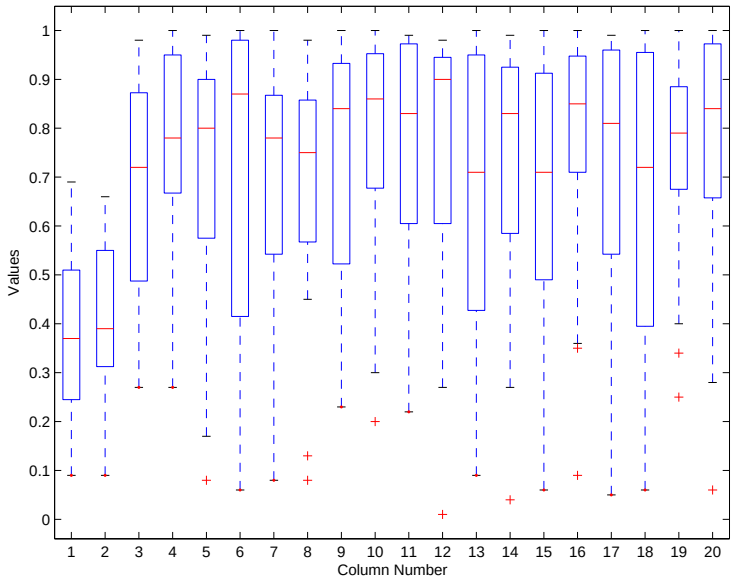

Figure 5.9: Box plot of testing example 2055 along iteration of MCMC samplings. Values on the vertical axis is the values of Θ while each column represents 25 iterations of the values of Θ.

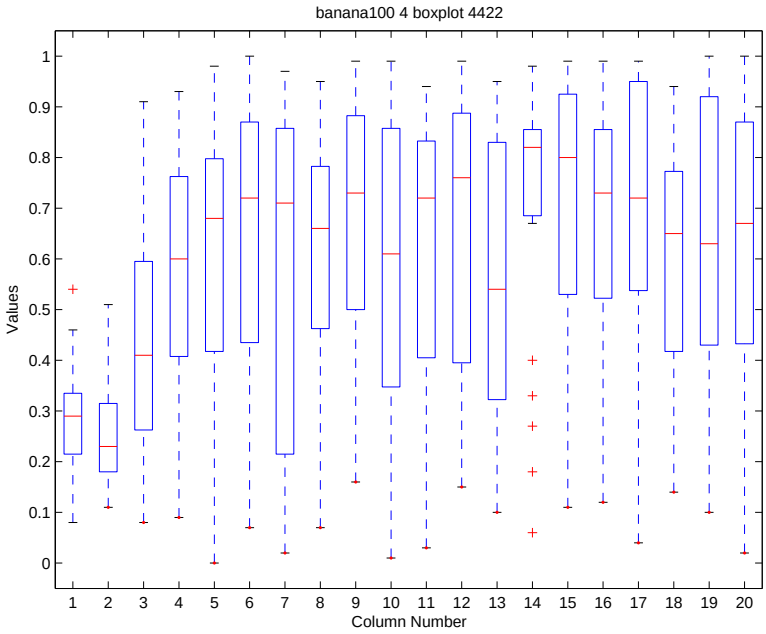

Figure 5.10: Box plot of testing example 4422 along iteration of MCMC samplings. Values on the vertical axis is the values of Θ while each column represents 25 iterations of the values of Θ.

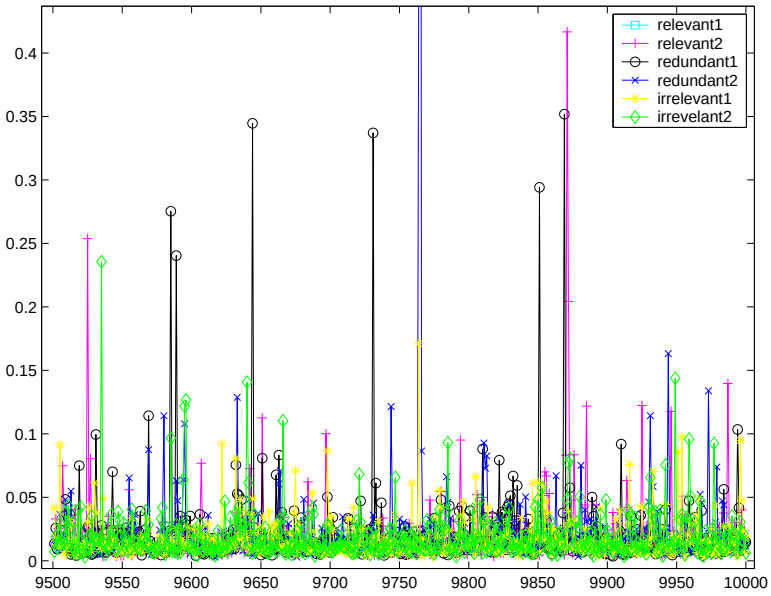

Figure 5.11: Values of Θ for Banana datasets, with prior distribution that fails to work. Only last 500 is shown here.

confident prediction (more than 0.7 or less than 0.3). This is to avoid the problem of using a "wrong" prior distribution of hyper-parameter.

Besides the implementation issues, we also tried to explore the possibility of applying Principal Component Analysis on the values of Θ. However, no concrete finding was obtained. The discussion of this study is in Appendix C.

Chapter 6

Methodology using Gaussian Processes

In this chapter, we will discuss the methodology for feature selection and then testing of the accuracy of the feature selection process. The testing is used for the evaluation of Gaussian Processes as a classifier for DNA Microarray datasets while ARD is the feature selection method.

We have made use of k-fold cross validation to evaluate the algorithm of the classifier. Basically, this re-sampling method splits the data into k folds. During each training process, $k-1$ folds will be used as training data while the remaining one fold will be used as testing data. The training process repeats k times so that all the folds have become testing data exactly once. The testing accuracy given by each training process will be the validation accuracy.

6.1 Feature Selection

For the purpose of feature selection, we use a routine, which we call Procedure Cross Validation, PCV(D,l), where D is the data used as the input to the routine and l is the number of feature returned as the output of the routine.

The PVC(D,l) routine does the following:

1. Stratify the data, D, into m folds. Each time one fold, i, is used as the

testing data while the remaining, D_{-i}, is used as the training dataset.

2. Apply filter method, either Information Gain and Fisher Score on D_{-i}. Based on the measure used for the filter method, the top 100 features are chosen.

3. D_{-i} with only these top 100 features are used as training data in Gaussian Processes. Compute $E(\Theta|D_{-i})$ (as in (3.18)) and come up with a ranking of the genes ordered by their respective Θ values.

4. Let $j = 1$. Select the top j genes based on the ranking of Θ obtained from step 3. Compute the number of correctly classified data as n_i^j for all data points in fold i (the testing data). Here, each data point in fold i is only represented by the top j genes.

5. Repeat step 4 by increasing j by 1 till $j = 100$.

6. Repeat step 2 to 5 over the m folds. In so doing, obtain the number of correctly classified examples with j gene, i.e. $n^j = \sum_{i=1}^{m} n_i^j$ for $j = 1, \ldots, 100$.

7. The number of genes with the best cross-validation accuracy is denoted by l, i.e. $l = \arg\max A^j$, where $A^j = \frac{n^j}{\|D\|}$.

With that, we have completed the m-fold cross validation process (let's call it internal cross validation). From the test accuracy of the m-fold cross validation, we will be able to know the optimal features to be used.

6.2 Unbiased Test Accuracy

In view of the small number of training data, it is important to assume that bias is not introduced . Based on the discussion of Ambroise and McLachlan (2002), we have designed a procedure to find the unbiased test accuracy of Gaussian Processes as a classifier.

The following external cross-validation procedure is used:

1. Stratify the data into p folds. Each time one fold, k, is used as the testing data while the remaining, D_{-k} is used as training dataset.

2. Invoke PCV(D_{-k},l).

3. Using the number l returned from PCV(D_{-k},l)from step 2, compute the testing accuracy on the k fold (the testing fold).

4. Repeat steps 2 and 3 for $k = 1, \ldots, p$.

5. Unbiased testing accuracy will be obtained based on the results from steps 1 to 4.

To avoid long search, we use p=3 and m=3 for all the experiments. If time is allowable, p=10 and m=10 is advisable. A three-fold cross validation uses at least five times shorter the time than a ten-fold cross validation will take.

6.3 Performance Measure

In step 4 of the PCV(D,l) routine, the performance of the Gaussian Processes is measured by n^j, where n^j is the number of correctly classified data in the testing data. Other measures exist to infer the performance of Gaussian Processes. Using the number of correctly classified data does not capture the confidence of a particular prediction. In this section, we introduce two other measures of performance for the purpose. Table 6.1 gives the three measures of performance.

Table 6.1: Three Measure of Performance

PM1	Test Accuracy
PM2	Negative Logarithm Likelihood
PM3	Negative Logarithm Likelihood (iteration)

Firstly, we will use the mean of the posterior probability (3.14). When the true label is 1, and if the mean of posterior probability is more than 0.5, the example is correctly classified. Then the summation is added by 1. When the true label is 0, and if the mean of posterior probability is less than 0.5, the example is again

correctly classified. Then, the summation is added by 1. However, if the true label is 1 and the mean of posterior probability is less than 0.5, the example is wrongly classified. Then there is no addition. Similarly for the case when the true label is 0 and the mean of posterior probability is more than 0.5. Mathematically, this is shown in (6.1).

$$Test\ accuracy = 100 \times \frac{1}{N} \sum_{i=1}^{N} \max\langle sign[(P(t(x^i) = +1|T) - 0.5) \times \widetilde{t^i}], 0\rangle \quad (6.1)$$

where $\widetilde{t^i}$ is the true label, taking values 1 or 0, and N is the number of testing examples.

We also use negative logarithm likelihood as measure. Assuming that all the testing examples are independent, we can find the likelihood using (6.2). This is to make use of the probabilistic modelling of Gaussian Processes. In this case, mean of the posterior probability is used. By using negative logarithm likelihood, we have include more information, especially the confident level of the prediction. However, unlike test accuracy, the value of equation (6.2) is used for comparison. The value by itself does not carry much meaning. For example, we will know how many examples are predicted correctly by checking equation (6.1), but we are not able to do so by using equation (6.2).

$$\textit{Negative Logarithm Likelihood}$$

$$= \sum_{i=1}^{N} \log\{(P(t(x^i) = +1|T)^{\widetilde{t^i}}(1 - (P(t(x^i) = +1|T))^{1-\widetilde{t^i}}\} \quad (6.2)$$

The last method is based on all the predicting probabilities of all iterations that are sampled from equilibrium distribution. And for each iteration, we will find the negative logarithm likelihood as (6.3). So, we can take (6.3) as the expected negative logarithm likelihood. Like (6.2), (6.3) is used for comparison among all the other (6.3). The value by itself does not carry much meaning.

Negative Logarithm Likelihood (*iteration*)

$$= \frac{1}{R} \sum_{j=1}^{R} \sum_{i=1}^{N} \log\{P_j(t(x^i) = +1|T)^{\widetilde{t^i}}(1 - P_j(t(x^i) = +1|T)^{1-\widetilde{t^i}}\} \qquad (6.3)$$

where R is the number of iterations used, and j is the j-th iteration.

Chapter 7

Results and Discussions

This chapter discusses the results obtained based on the methodology mentioned in section 6. They include discussion on the performance of Gaussian Processes as the classifier and then the genes selected by ARD. The notations used follow the previous chapter if not re-defined.

7.1 Unbiased Test Accuracy

This section presents the results based on the methodology mentioned in Section 6.2. The performance measurement is based on the discussion in Section 6.3. For notation, the definition used follows Section 6.2 if it is not defined in this section.

The results of applying the unbiased test accuracy methodology are given in Table 7.1 to Table 7.8 for the four datasets which are Colon Cancer, Breast Cancer, Leukaemia and Ovarian. They also differ in terms of the initial feature selection method. Only the optimal number of features (as in l in routine PVC(D,l)) in the internal cross validation is presented here. Based on the optimal number of features in the third column, we obtain the average unbiased test accuracy, given in the fourth column. Note that PM1, PM2 and PM3 are used for the PCV routine. The external cross-validation process has only PM1.

From Table 7.1, Table 7.3 and Table 7.4, values of PM1 of the three internal cross validation show some variation. One possible reason for this could be the

Table 7.1: Results of the unbiased test accuracy methodology for Breast Cancer Dataset-Fisher Score and ARD

	Internal Cross Validation Run	Number of Features	Performance	Overall Unbiased Test Accuracy
PM1	R1	16	68.97%	
	R2	11	89.66%	
	R3	7	76.67%	
PM1				84.09%
PM2	R1	17	16.05	
	R2	11	10.23	
	R3	6	11.70	
PM1				79.55%
PM3	R1	11	22.42	
	R2	11	17.83	
	R3	6	18.42	
PM1				81.82%

Table 7.2: Results of the unbiased test accuracy methodology for Breast Cancer Dataset-Information Gain and ARD

	Internal Cross Validation Run	Number of Features	Performance	Overall Unbiased Test Accuracy
PM1	R1	10	93.10%	
	R2	12	89.66%	
	R3	18	83.33%	
PM1				84.09%
PM2	R1	10	12.22	
	R2	15	10.96	
	R3	18	12.52	
PM1				84.09%
PM3	R1	6	18.63	
	R2	19	16.69	
	R3	15	19.92	
PM1				68.18%

Table 7.3: Results of the unbiased test accuracy methodology for Colon Cancer Dataset-Fisher Score and ARD

	Internal Cross Validation Run	Number of Features	Performance	Overall Unbiased Test Accuracy
PM1	R1	6	97.56%	
	R2	8	70.73%	
	R3	12	76.19%	
PM1				80.65%
PM2	R1	7	4.34	
	R2	20	25.69	
	R3	16	21.33	
PM1				82.26%
PM3	R1	7	6.11	
	R2	9	41.44	
	R3	6	31.35	
PM1				81.82%

Table 7.4: Results of the unbiased test accuracy methodology for Colon Cancer Dataset-Information Gain and ARD

	Internal Cross Validation Run	Number of Features	Performance	Overall Unbiased Test Accuracy
PM1	R1	7	85.37%	
	R2	12	87.81%	
	R3	5	78.57%	
PM1				87.10%
PM2	R1	12	19.18	
	R2	15	14.78	
	R3	17	24.07	
PM1				87.10%
PM3	R1	20	25.02	
	R2	10	21.30	
	R3	7	32.33	
PM1				87.10%

Table 7.5: Results of the unbiased test accuracy methodology for Leukaemia Dataset-Fisher Score and ARD

	Internal Cross Validation Run	Number of Features	Performance	Overall Unbiased Test Accuracy
PM1	R1	7	75%	
	R2	16	70.83%	
	R3	16	79.17%	
PM1				77.78%
PM2	R1	5	26.337	
	R2	18	28.02	
	R3	7	26.25	
PM1				77.78%
PM3	R1	5	34.15	
	R2	10	44.99	
	R3	5	39.06	
PM1				75.00%

Table 7.6: Results of the unbiased test accuracy methodology for Leukaemia Dataset-Information Gain and ARD

	Internal Cross Validation Run	Number of Features	Performance	Overall Unbiased Test Accuracy
PM1	R1	7	79.17%	
	R2	20	83.33%	
	R3	7	79.17%	
PM1				68.00%
PM2	R1	9	22.25	
	R2	10	22.265	
	R3	9	24.44	
PM1				65.28%
PM3	R1	7	30.48	
	R2	7	34.24	
	R3	6	32.15	
PM1				61.11%

Table 7.7: Results of the unbiased test accuracy methodology for Ovarian Cancer Dataset-Fisher Score and ARD

	Internal Cross Validation Run	Number of Features	Performance	Overall Unbiased Test Accuracy
PM1	R1	9	80.56%	
	R2	11	83.33%	
	R3	9	77.78%	
PM1				70.37%
PM2	R1	18	19.36	
	R2	11	13.17	
	R3	9	17.98	
PM1				70.37%
PM3	R1	5	25.85	
	R2	6	17.65	
	R3	9	22.54	
PM1				66.67%

Table 7.8: Results of the unbiased test accuracy methodology for Ovarian Cancer Dataset-Information Gain and ARD

	Internal Cross Validation Run	Number of Features	Performance	Overall Unbiased Test Accuracy
PM1	R1	18	83.33%	
	R2	13	80.56%	
	R3	8	83.33%	
PM1				79.63%
PM2	R1	7	14.53	
	R2	19	18.21	
	R3	12	17.13	
PM1				81.48%
PM3	R1	7	20.73	
	R2	13	23.12	
	R3	5	21.56	
PM1				75.93%

use of small number of folds in the cross validation process.

We can also observe that the average test accuracy (fourth columns of tables) differs from each internal cross validation (third columns of tables). Again, this may be due to the small number of folds and examples available. There is a high chance that the data is split into the folds non-uniformly. As a result, training process of the classifier during internal cross validation fails to learn such cases. The problem can be overcome if we use ten-fold cross validation or leave-one-out cross validation.

Generally, PM1 performs well compared to the other two measurements. Thus, for feature selection and other comparisons, we will mainly use the results based on PM1 as performance measurement.

We will compare the average unbiased test accuracy mainly with Shevade and Keerthi (2002) as it also used average test accuracy for unbiased estimation of the classifier performance.

Table 7.9: Comparison for different dataset with the Sparse Logistic Regression method of Shevade and Keerthi (2002)

Data set	Sparse Logistic Regression	Fisher Score and ARD	Information and ARD
Breast Cancer	81.2%	84.09%	84.09%
Colon Cancer	82.3%	80.65%	87.10%
Leukaemia	93.1%	77.78%	68.00%
Ovarian Cancer	83.3%	70.37%	79.63%

From Table 7.9, ARD with Information Gain performs competitively in Breast Cancer, Colon Cancer and Ovarian Cancer datasets, while ARD with Fisher Score works well in Breast Cancer, and Colon Cancer dataset.

We also compare the methodology with Long and Vega (2003) who used Boosting method to obtain cross validation estimates. There are eight algorithms used by Long and Vega (2003). Three of six datasets, i.e. Breast Cancer, Colon Cancer and Leukaemia, used by Long and Vega (2003) are compared. Since Long and Vega (2003) set gene limit to ten and hundred, we will do the same using our methodology. In step 4 in PCV routine, j is fixed at 10 (i.e. $j = 10$).

Table 7.10: Comparison for different dataset with Long and Vega (2003) with gene limited at 10

Algorithm	Breast Cancer	Colon Cancer	Leukaemia
Fisher Score with ARD	77.3%	83.9%	79.2%
Information Gain with ARD	79.5%	87.1%	62.5%
Adaboost	80.1%	74.7%	93.8%
Adaboost-VC	81.9%	75.6%	96.1%
Adaboost-NR	80.5%	74.9%	94.0%
Adaboost-PL	79.4%	76.6%	93.0%
Arc-x4-RW	80.2%	75.0%	91.8%
Arc-x4-RW-NR	82.2%	75.3%	96.7%
SVM-RFE	79.1%	80.8%	86.6%
Wilcoxon/SVM	76.8%	75.7%	93.6%

As we have observed previously, the methodology we use performs poorly for Leukaemia dataset compared with the other algorithms, especially Information Gain with ARD. However, for the other two datasets, the methodology shows very competitive results. For Colon dataset in particular, the average accuracy is larger than that of other algorithms.

Another point that needs clarification is that we expect that the results of our methodology shown in Table 7.9 should be similar to, if not better than, that of Table 7.10. The reason is because we consistently look for the optimal number of features in the former experiment. However, for some of the cases, it is not so. Again, we think that this is due to the three-fold cross validation and the low number of features available. The problem will most probably be solved if ten-fold cross validation is used.

From the discussion above, we can see that the methodology using filter methods with ARD is competitive and giving comparable results with other algorithms. If time is allowable, ten-fold cross validation can be used for better confident level as well as more "stable" results.

7.2 Feature Selection

From the results in section 7.1, we make use of two results which we apply in feature selection process.

1. The measurement for performance will be based on the test accuracy (that is PM1).

2. Certain combinations of feature selection methods are applied on different datasets, as shown in Table 7.11.

Table 7.11: Feature selection method used in different dataset

Dataset	Feature Selection Method
Breast Cancer	Both Combinations
Colon Cancer	Information Gain with ARD
Leukaemia	Fisher Score with ARD
Ovarian Cancer	Information Gain with ARD

And, Table 7.12 shows the number of genes that give best performance based on the feature selection process discussed in section 6.1.

Table 7.12: Optimal number of features on different dataset

Dataset	Optimal Number of Feature
Breast Cancer (Fisher Score with ARD)	12
Breast Cancer (Information Gain with ARD)	15
Colon Cancer	7
Leukaemia	13
Ovarian Cancer	12

Table 7.13: Selected genes for the breast cancer based on fisher score with ARD

ID	Gene Annotation
6484	H.sapiens mRNA for cathepsin C
1376	Guanosine 5' -Monophosphate Synthase
709	Human mRNA for KIAA0182 gene, partial cds
1488	Human transforming growth factor-beta 3 (TGF-beta3) mRNA, complete cds
4234	Human splicing factor SRp40-3 (SRp40) mRNA, complete cds
2159	Homo sapiens (clone zap128) mRNA, 3'end of cds
4	Human mRNA for semaphorin E, complete cds
6917	H.sapiens kinectin gene
1095	Tyrosine Kinase
822	Human mRNA for KIAA0232 gene, complete cds
1512	Y box binding protein-1 (YB-1) mRNA
1505	Human microtubule-associated protein tau mRNA, complete cds

Comparing the results of Shevade and Keerthi (2002), all the top six genes that were selected by Shevade and Keerthi (2002) appear in Table 7.13, while three out

Table 7.14: Selected genes for the breast cancer dataset based on Information Gain with ARD

ID	Gene Annotation
6484	H.sapiens mRNA for cathepsin C
1881	Human high mobility group protein (HMG-I(Y)) gene exons 1-8, complete cds
1376	Guanosine 5' -Monophosphate Synthase
4234	Human splicing factor SRp40-3 (SRp40) mRNA, complete cds
2159	Homo sapiens (clone zap128) mRNA, 3'end of cds
3464	L-arginine:glycine amidinotransferase [human, kidney carcinoma cells, mRNA, 2330 nt]
2779	Human omega light chain protein 14.1 (Ig lambda chain related) gene
1512	Y box binding protein-1 (YB-1) mRNA
4629	Human leukotriene C4 synthase (LTC4S) gene, complete cds
4169	Human p16INK4/MTS1 mRNA, complete cds
1025	Major Intrinsic Protein
3998	Human TFIID subunit TAFII55 (TAFII55) mRNA, complete cds
1999	Human autoantigen pericentriol material 1 (PCM-1) mRNA, complete cds
5642	Human CD14 mRNA for myelid cell-specific leucine-rich glycoprotein
362	Human mRNA for platelet-type phosphofructokinase, complete cds

of the six genes appeared in Table 7.14 (which are gene ID 6484, 4234 and 1512). In West et al. (2001), there are 40 genes listed. Only 2 (gene ID 6484 and 1505) and 1 (gene ID 6484) genes found in Table 7.13 and Table 7.14 respectively.

Table 7.15: Selected genes for the colon cancer dataset based on Information Gain with ARD

ID	Gene Annotation
377	H.sapiens mRNA for GCAP-II/uroguanylin precursor
493	MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)
1423	MYOSIN REGULATORY LIGHT CHAIN 2, SMOOTH MUSCLE ISOFORM (HUMAN); contains element TAR1 repetitive element
1325	S-100P PROTEIN (HUMAN)
14	MYOSIN LIGHT CHAIN ALKALI, SMOOTH-MUSCLE ISOFORM (HUMAN)
765	Human cysteine-rich protein (CRP) gene, exons 5 and 6
627	H.sapiens mRNA for protein tyrosine phosphatase

Comparing Table 7.15 and results from Shevade and Keerthi (2002), there are only two genes in common, which are gene ID 377 and 493. These two genes appear to be the top two genes given by our methodology. Same two genes are also mentioned in Li et al. (2001b). Of the 50 genes listed in Bo and Jonassen (2002) who evaluate genes in pairs, all the top 7 genes are included. However, none of the gene is in common with the 7 genes selected by Guyon et al. (2002).

Table 7.16: Selected genes for the leukaemia dataset based on fisher score with ARD

ID	Gene Annotation
4781	Gp25L2 protein
2238	GATA2 GATA-binding protein 2
4142	CD37 CD37 antigen
2402	Azurocidin gene
2641	Fetal Alz-50-reactive clone 1 (FAC1) mRNA
5352	GB DEF = Nonmuscle myosin heavy chain-B (MYH10) mRNA, partial cds
5402	E2F5 E2F transcription factor 5, p130-binding
6134	GB DEF = Clone 110298 map Xq28, mRNA sequence
2242	PEPTIDYL-PROLYL CIS-TRANS ISOMERASE, MITOCHONDRIAL PRECURSOR
2626	MSH2 DNA repair protein MSH2
2056	NFKB1 Nuclear factor of kappa light polypeptide gene enhancer in B-cells 1 (p105)
5418	DNA-dependent protein kinase catalytic subunit (DNA-PKcs) mRNA
3258	Phosphotyrosine independent ligand p62 for the Lck SH2 domain mRNA

The genes given in Table 7.16 do no coincide with that of Shevade and Keerthi (2002), Guyon et al. (2002), Li et al. (2001b) nor Szabo et al. (2002).

Table 7.17: Selected genes for the ovarian cancer dataset based on fisher score with ARD

ID	687	1526	1491	1028	93	1419	658	1387	1117	1512	596	47

Of the 12 genes mentioned in Table 7.17, gene ID 1526 and 93 are considered important features of algorithm 1 in Li et al. (2001b) and gene ID 1526 and 1491 are considered important features of algorithm 2 in Li et al. (2001b). Comparing with Shevade and Keerthi (2002), gene ID 1526 and 1387 are the common features.

From the results of the feature selection process, we can observe that gene selected by the methodology does give some common genes, except Leukaemia dataset.

Chapter 8

Conclusion

In this thesis, we have used ARD as the feature selection tool for DNA microarray data. In the probabilistic framework of Gaussian processes along with ARD parameters, we apply the external cross validation methodology. Based on the design, we have observed competitive results. Except the Leukaemia dataset, the results of the other three datasets show that the methodology perform closely with the results of Shevade and Keerthi (2002). Also, we compare the genes selected under the methodology with the rest of the literature. Again, we can see some of the genes selected are in common (particularly Breast Cancer Dataset). It is the aim of this project to highlight a small number of some genes (from thousands of them) which the classifier (Gaussian Processes) and the methodology (Information Gain with ARD and Fisher Score with ARD) have identified. We hope that this will help the biologists shorten their time to find out the culprit of certain disease. With the acquired knowledge gain, they may be able to develop procedures and drugs to prevent diseases.

During the project, we also have gained better understanding on Gaussian Processes. We have proposed the use of the median of posterior probabilities to check whether the equilibrium distribution is reached. This is mainly due to a lack of other methods to judge whether the samplings are done on equilibrium distribution.

There are also many other ways to speed up the training process. In the soft-

ware, there are different kinds of Monte Carlo Markov Chain methods to be used. There are even a few versions of Hybrid Monte Carlo (HMC) which may improve the sampling. For simplicity, the HMC that used in the project is mainly based on Duan et al. (1987). However, different versions that speed up the attainment of equilibrium state are available. Theoretically, any version used will still bring us to invariant distribution. Further study could investigate which version of MCMC sampling methods are more suitable for DNA Microarray datasets.

Another main question which remained unsolved is the choice of the prior of hyper-parameters. From the observation and discussion in Section 5.1.4, it is important to choose the right parameter for the Gamma Distribution. We believe that these parameters are data dependent, especially the spreading of the values of the features. However, further study is required to gain better understanding in this matter.

Also, there are a few issues on the design of the methodology that we would like to highlight. Firstly, we only use three-fold cross validation. However, in our view, it is advisable to use ten-fold cross validation for both internal and external cross validation. This will give results with higher confidence and accuracy. In view of the small number of examples available, if the computation time and power is not an issue, leave-one-out cross validation can be used too. Another concern of using three-fold cross validation is the reproducibility of the results obtained in this thesis. Again, this can be overcome by using a larger number of k-fold cross validation.

We have employed Information Gain and Fisher Score as filter methods to reduce the number of features largely. 100 features in this case was arbitrarily set. Such a crude cut over the number of the features may be not optimal. However, from the results (average test accuracy), the number seems to be a reasonable. Moreover, the filter method used in this case does not seem to affect the final results significantly. Nevertheless, in the future, if there is an indication that more genes are required, we can use 200 features or more. The increase in the number of features does require more time to do a MCMC samplings in Gaussian Processes.

References

A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J.C. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson, L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisenburger, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, and L. M. Staudt. Different types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 511:403–503, 2000.

U. Alon, N. Barkai, D. Nottermau, K. Gish, S. Ybarra, D. Mack, and A. Levine. Broad patterns of gene expression revealed by clustering analysis of tumour and normal colon cancer tissues probed by oligonucleotide arrays. *Cell Biology*, 96: 6745–6750, 1999.

C. Ambroise and G. J. McLachlan. Selection bias in geneextraction on the basis of microarray gene expression data. *Preedings of the National Academy of Sciences, USA*, 99:668–674, 2002.

A. Ben-Dor, N. Friedman, and Z. Yakhini. Scoring genes for relevance. Technical Report AGL-2000-13, Agilent Laboratories, 2000.

A. Bendor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini. Tissue classification with gene expression profiles. *Proceedings of the 4th Annual International Conference on Computational Molecular Biology (RECOMB)*, 2000.

T. H. Bo and I. Jonassen. New feature subset selection procedures for classification of expression profiles. *Genome Biology*, 3(4):1–11, 2002.

A. Bronzma and J. Vilo. Gene expression data analysis. *Microbes and Infection*, 3:823–829, 2001.

M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machine. *Proceedings Natl. Acad. Sci.*, 97:262–267, 2000.

M. L. Chow, E. J. Moler, and I. S. Mian. Identifying marker genes in transcription profiling data using a mixture of feature relevance experts. *Physiol Genomics*, 5:99–111, 2001.

N. A. C. Cressie. *Statistics for Spatial Data*. Wiley Interscience, 1991.

J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–685, 1997.

S. Duan, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195:216–222, 1987.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Recognition*. John Wiley Sons Inc, 2nd edition, 2001.

J. Dudoit, S. Fridlyand and T. P. Speed. Comparison of discrimination methods for the classification of tumor using gene expression data. Technical report, University of California, Berkeley, 2000.

J. D. Duggan, M. Bittner, Y. Chen, and J. M. Trent. Expression profiling using cdna microarrays. *Nature Genetics*, 21:10–14, January 1999.

T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler. Suport vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

Nature Genetics. The chipping forecast. *Nature Genetics Supplement*, 21, 1999.

M. N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. Phd thesis, Cambridge University, 1997.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.

T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, Downing J. R., M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

I. Guyon, J Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machine. *Machine Learning*, 46:389–422, 2002.

A. M. Horowitz. A generalized guided monte carlo algorithm. *Physics Letters B*, 268:247–252, 1991.

T. R. Hvidsten, J. Komorowski, A. K. Sandvik, and A Laegreid. Predicting gene function from gene expressions and ontologies. *Proceedings of the Pacific Symposium on Biocomputing*, 2001.

A. G. Journel and C. J. Huijbregts. *Mining Geostatisitcs*. Academic Press, 1978.

A. D. Keller, M. Schummer, L. Hood, and W. L. Ruzzo. Bayesian classification of dna array expression data. Technical Report UW-CSE-2000-08-01, University of Washington, 2000.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *AIJ Special Issue on Relevance*, 1996.

L. Li, C. R. Weinberg, Thomas A. Darden, and L. G. Pedersen. Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the ga/knn method. *Bioinformatics*, 17(12):1131–1142, 2001a.

Y. Li, C. Campbell, and M. Tipping. Bayesian automatic relevance determination algorithms for classifying gene expression data. 2001b.

R. J. Lipshutz, S. Fodor, and D. Gingeras, T.and Lockhart. High density synthetic oligonucleotide arrays. *Nature Genetic*, 21:20–24, 1999.

D. J. Loackhart, H. L. Dong, Byrne M. C., M. T. Follettie, M. V. Gallo, M. S. CHee, M. Mittmann, C. Wang, M. Kobayashi, and H. Horton. Expression monitoring by hribridization to high density oligonucleotide arrays. *Nature Biotechnology*, 14:1675–1680, 1996.

P. M. Long and V. B. Vega. Boosting and microarray data. *Kluwer Academic Publishers*, January 2003.

D. G. Lowe. Similarity metric learning for a variable kernel classifier. *Neural Computation*, 7:72–85, 1995.

B. M. Luscombe, D. Greenbaum, and M. Gerstein. What is bioinformatics? a proposed definition and overview of the field. *Methods Inf Med*, 40(4):346–58, 2001.

D. J. C. Mackay. *Bayesian Methods for Backpropagation Networks*. Springer, models of neural networks ii edition, 1993.

D. J. C. Mackay. *Introduction to Monte Carlo Methods*, pages 175–204. NATO Science Series. Kluwer Academic Press, 1998.

A. H. Mark. *Correlation-based Feature Selection for Machine Learning*. Doctor of philosophy thesis, The University of Waikato, 1999.

R. M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

R. M. Neal. An improved acceptance procedure for the hybrid monte carlo algorithm. *Journal of Computational Physics*, 1994.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer Verlag, 1996. Revised version of Ph. D. thesis from Graduate Department of Computer Science, University of Toronto.

R. M. Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification. Technical Report 9702, Department of Statistics, University of Toronto, http://www.cs.toronto.edu/ radford/., 1997.

D. V. Nguyen and D. M. Rocke. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, 18(1):39–50, 2002.

A. O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society B*, 40:1–42, 1978.

M. Opper and O. Winther. Gp classification and svm: Mean field results and leave-one-out estimator. *In Advances in Large Margin Classifiers. MIT Press*, 1999.

C. M. Perou, Jeffrey S. S., M. van de Rijn, C. A. Rees, M. B. Eisen, Ross D. T., A. Pergamenschikov, C. F. Williams, S. X¿ Zhu, J. C. F . Lee, D. Lashkari, D. Shalon, P. O. Brown, and D. Botstein. Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. *Proceeding Natl. Acad. Sci.*, 96:9212–9217, 1999.

B. Phimister. Going global. *Nature Genetics*, 21:1, 1999.

T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical report a. i., M.I.T., 1989.

J. R. Pollack, C. M. Perou, A. A. Alizadeh, M. B. Eisen, A. Pergamenschikov, C. F. Williams, S. S. Jeffery, D. Botstein, and P. O. Brown. Genome-wide analysis of dna copy-number changes using cdna microarrays. *Nature Genetics*, 23:41–46, 1999.

D. T. Ross, U. Scherf, M. B. Eisen, C. M. Perou, P. Spellman, V. Iyer, S. S. Jeffery, M van de Rijn, M. Waltham, A. Pergamenschikov, J. C. F. Lee, D. Lashkari, D. Shalon, T. G. Myers, J. N. Weinstein, D. Botstein, and P. O. Brown. Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics*, 24:227–234, 2000.

M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270: 467–470, 1995.

M. Seeger. Bayesian model selection for support vector machines, gaussian processes and other kernel classifiers. *In Advances in Neural Information Processing Systems, MIT Press*, 1999.

S. K. Shevade and S. S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. To be appear in Bioinformatics Journal, Nov 2002.

D. K. Slonim, P Tamayo, J. P. Mesirov, T. R. Golub, and E. R. Lander. Class prediction and discovery using gene expression data. *In Proceedings of the 4th Annual International Conference on Computational Molecular Biology (RECOMDB)*, pages 263–272, 2000.

E Southern, K. Mir, and M. Shchepinov. Molecular interactions on microarrays. *Nature Genetics*, 23:5–9, 1999.

A. Szabo, K. Boucher, W. L. Carroll, L. B. Klebanov, and A. D. Tsodikov. Variable selection and pattern recognition with gene expression data generated by the microarray technology. *Mathematical Biosciences*, 176:71–98, 2002.

M. A. Tanner. *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*. Springer Series in Statistics. Springer Verlag, 3rd edition, 1996.

G. Wahba. Spline models for obsevational data. *CBMS-NSF Regional Conference Series in Applied Mathematics*, 1990. Society for Industrial and Applied Mathematics.

M West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Jr. Olson, J. R. Marks, and J. R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences*, 98:11462–11467, 2001.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. *Advances in Neural Information Processing Systems 13*, 2000.

C. K. I. Williams. Prediction with gaussian processes: From linear regression to linear prediction and beyond. Technical Report NCRG/97/012, Neural Computing Research Group, 1997.

C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing Systems 8*, 1996.

C. K. L. Williams and D. Barber. Bayesian classification with gaussian processes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

C Workman, L. J. Jensen, H Jarmer, R Berka, L Gautier, H. B. Nielser, H. Saxild, C Nielsen, S. Brunak, and S Knudsen. A new non-linear normalization method for reducing variability in dna microarray experiments. *Genome Biology*, 3(9): 1–16, 2002.

Appendix A

Figures for Banana Experiments

The figures are obtained for gaining a better understanding on Gaussian Process. The detail of the experiments and the discussion can be found in Chapter 5.

The following is the locations the training examples and testing examples in the two relevant features space.

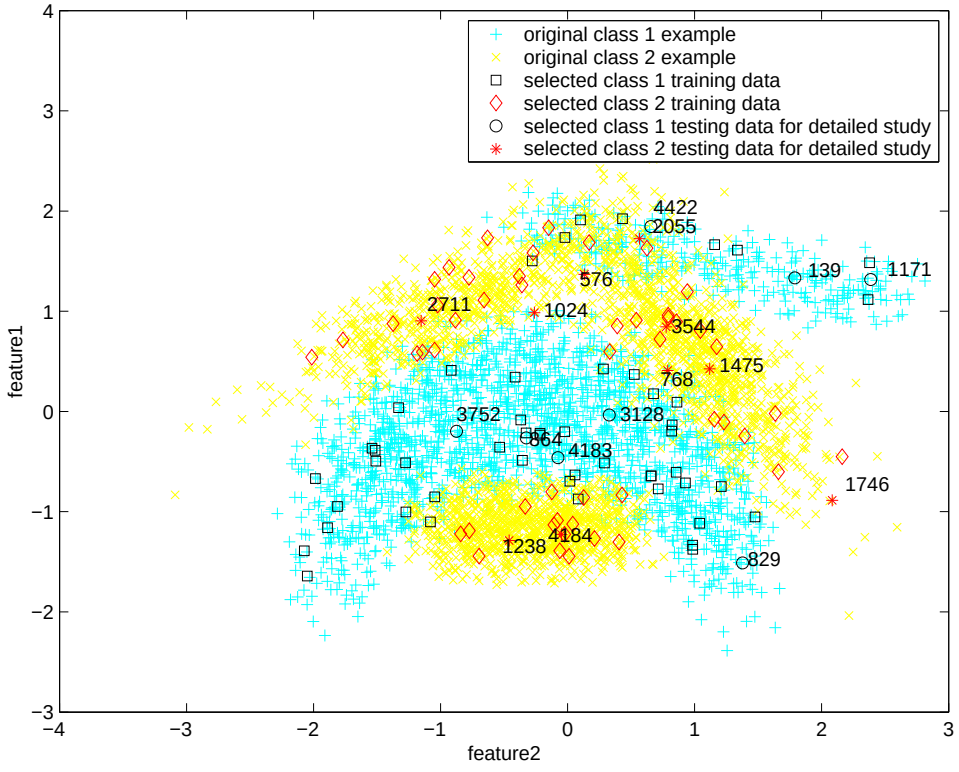

Figure A.1: Location of training and testing examples in the feature space.

The followings are the figures that shows the box plot of the testing examples

of interest. The vertical axis is the predicting probability of class 1. The horizontal axis is the iterations. For each box plot, it represents 25 iterations.

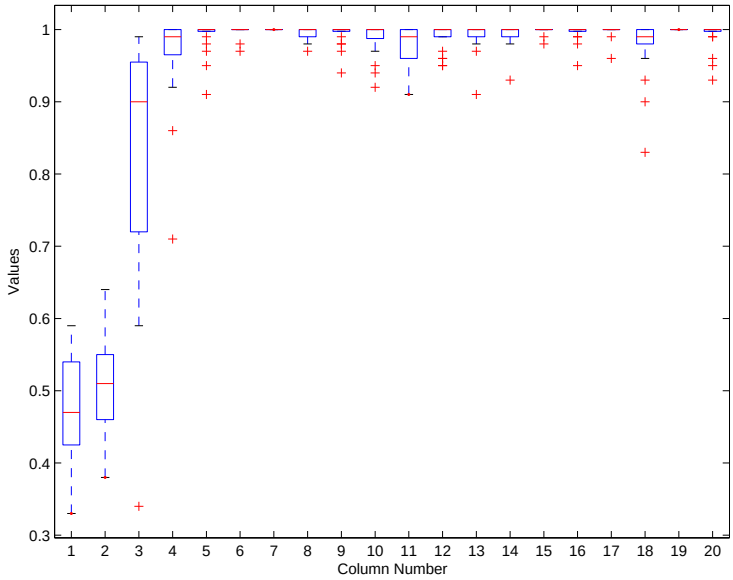

Figure A.2: Box plot of testing example 3128 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

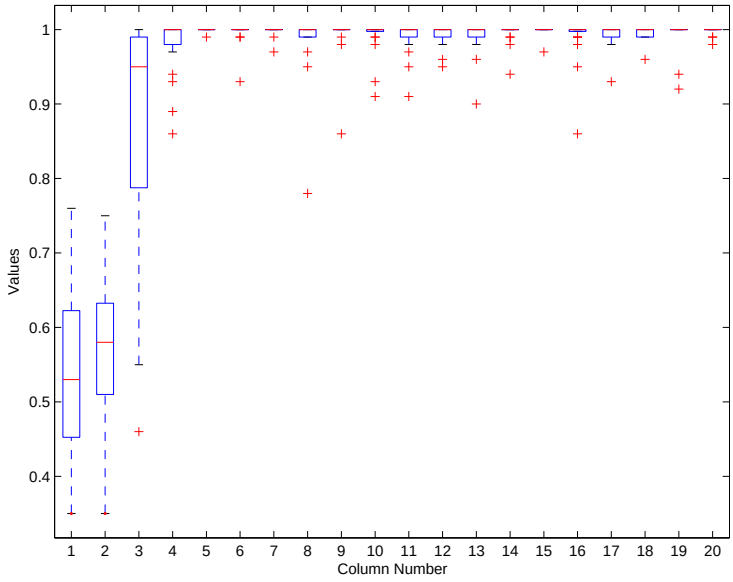

Figure A.3: Box plot of testing example 864 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

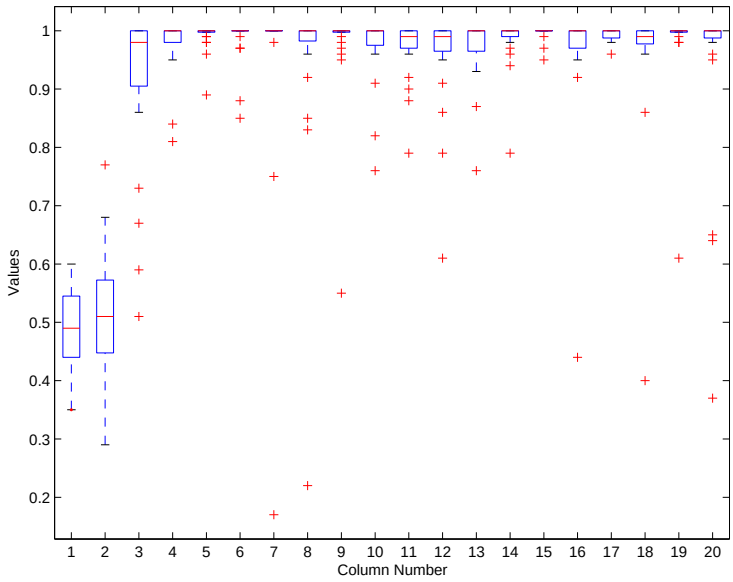

Figure A.4: Box plot of testing example 3752 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

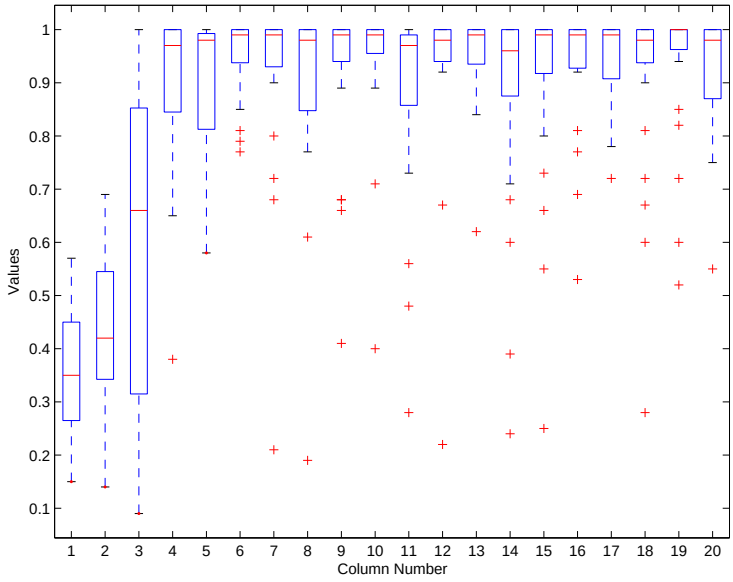

Figure A.5: Box plot of testing example 1171 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

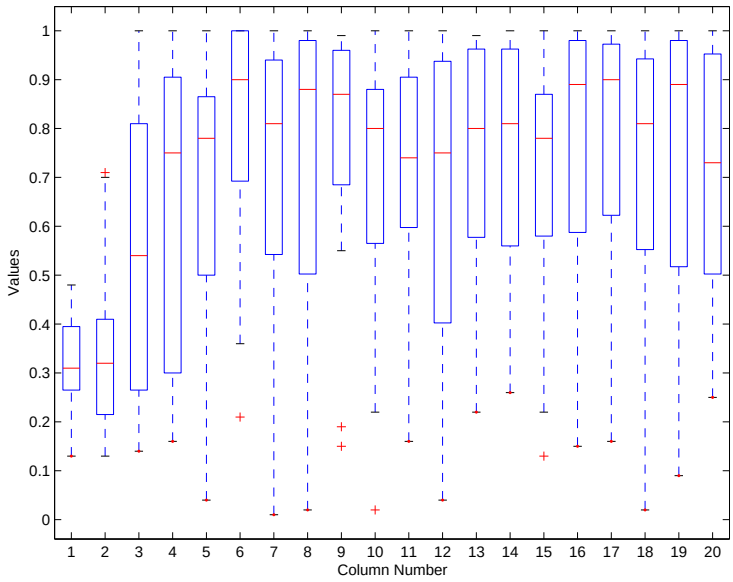

Figure A.6: Box plot of testing example 139 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

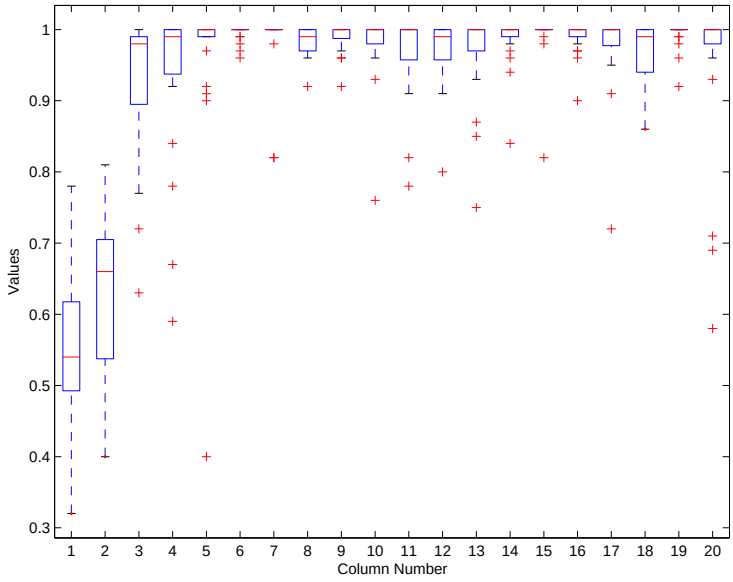

Figure A.7: Box plot of testing example 4183 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

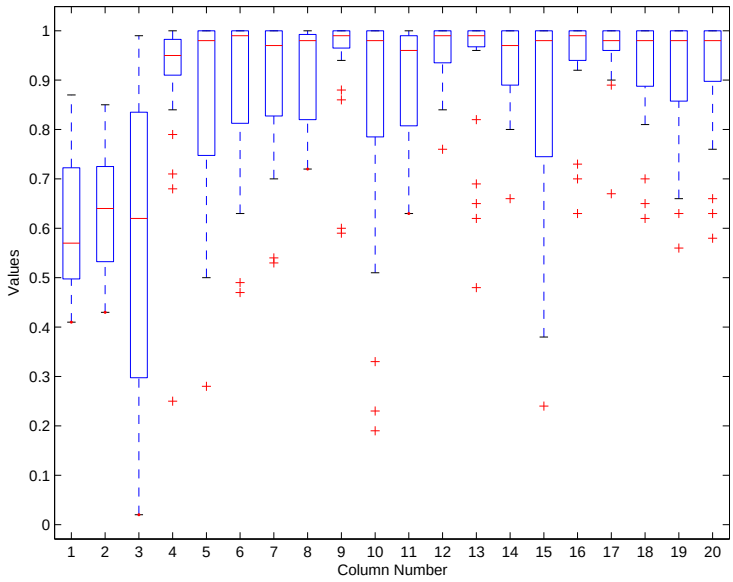

Figure A.8: Box plot of testing example 829 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

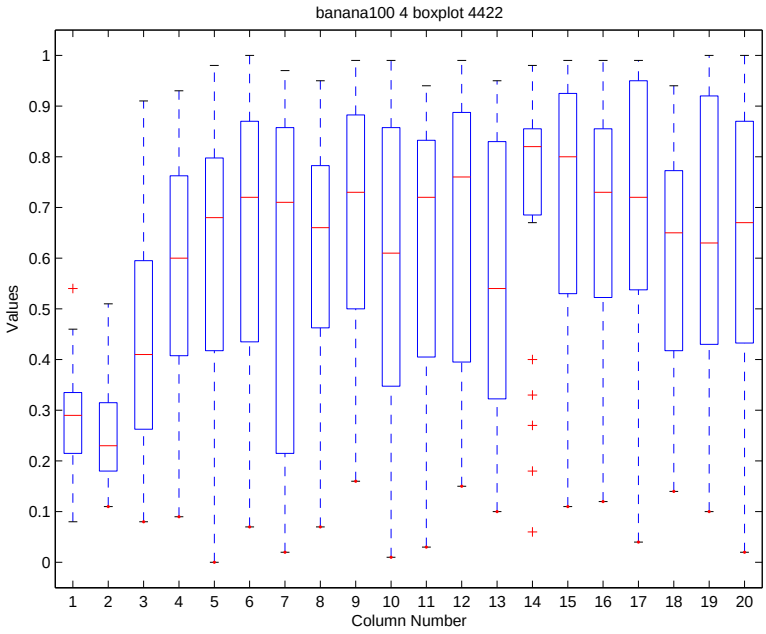

Figure A.9: Box plot of testing example 4422 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

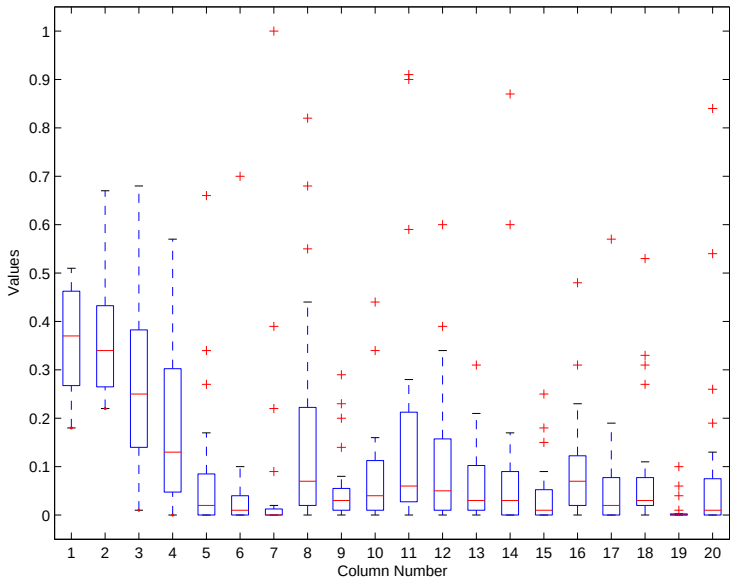

Figure A.10: Box plot of testing example 3544 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

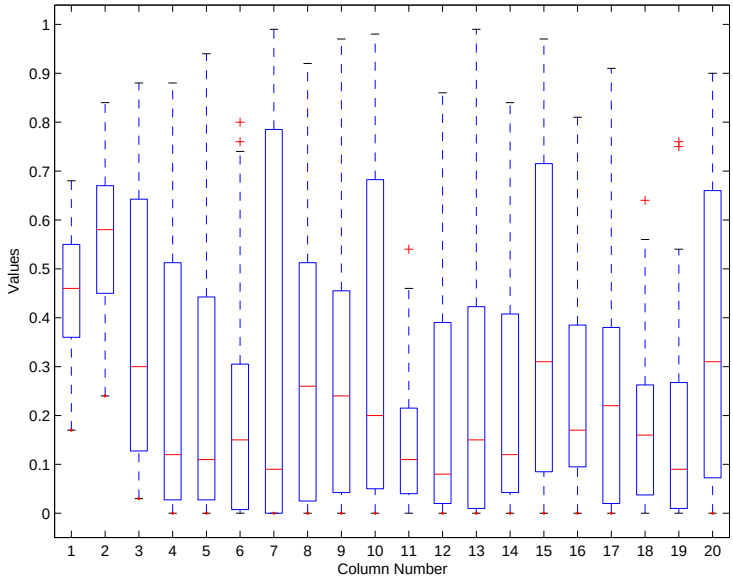

Figure A.11: Box plot of testing example 1475 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

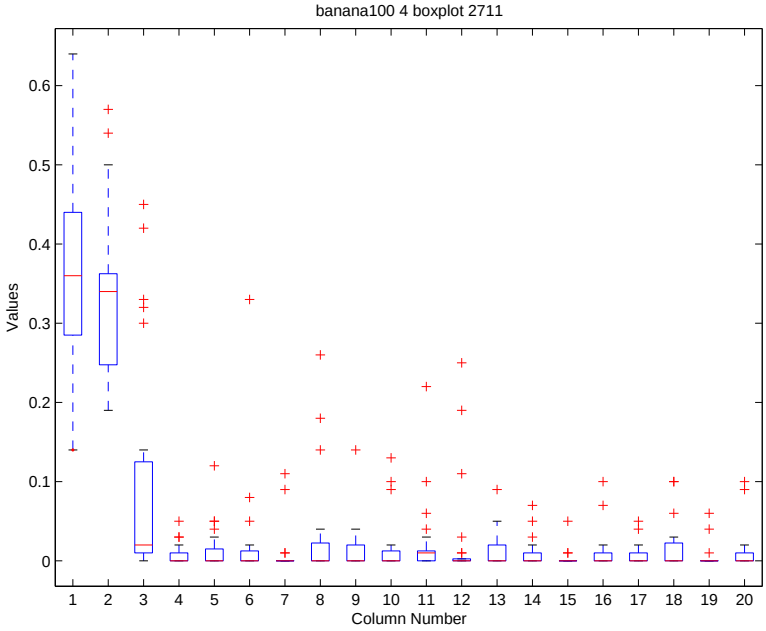

Figure A.12: Box plot of testing example 2711 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

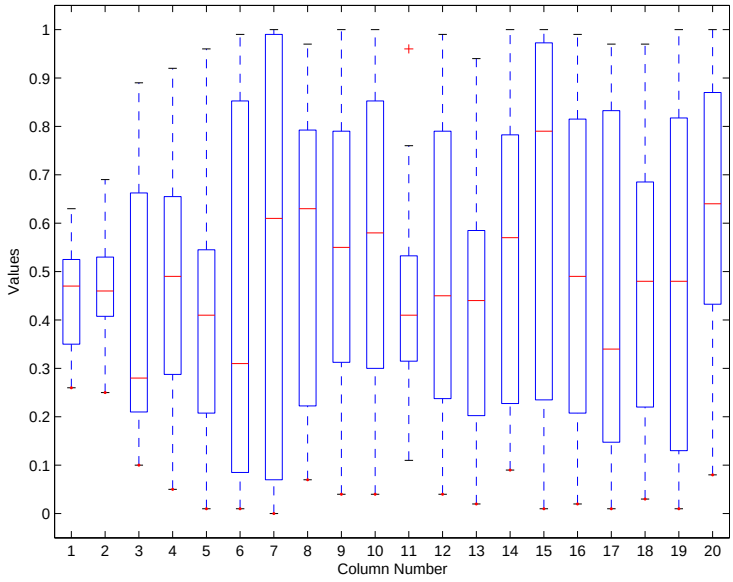

Figure A.13: Box plot of testing example 768 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

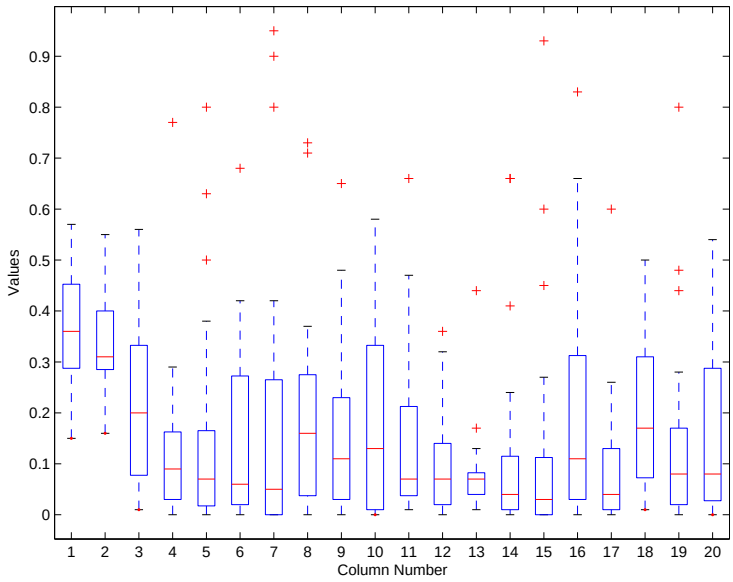

Figure A.14: Box plot of testing example 576 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

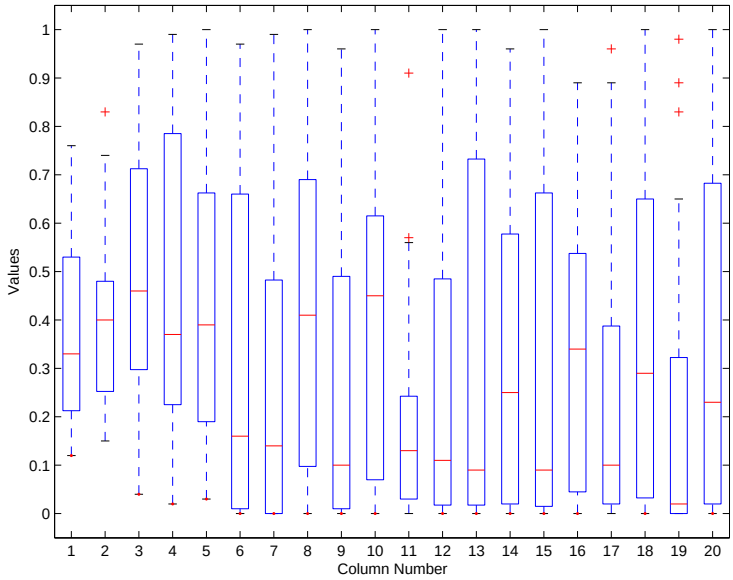

Figure A.15: Box plot of testing example 1024 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

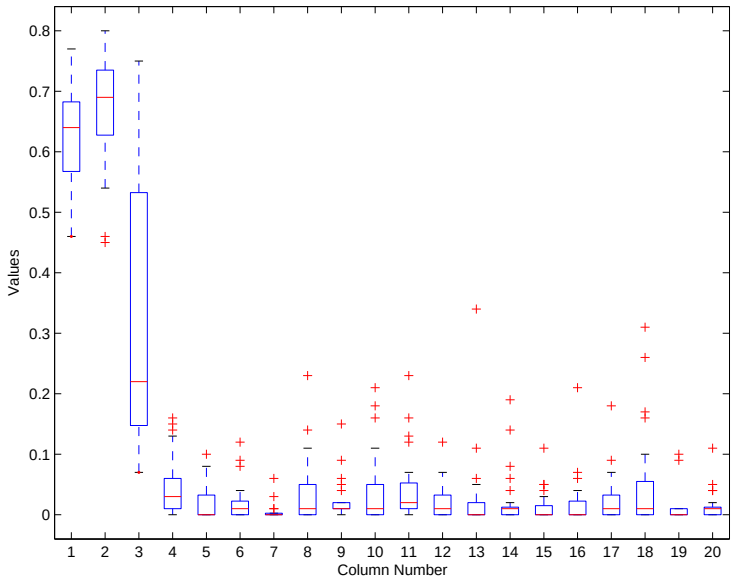

Figure A.16: Box plot of testing example 1238 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

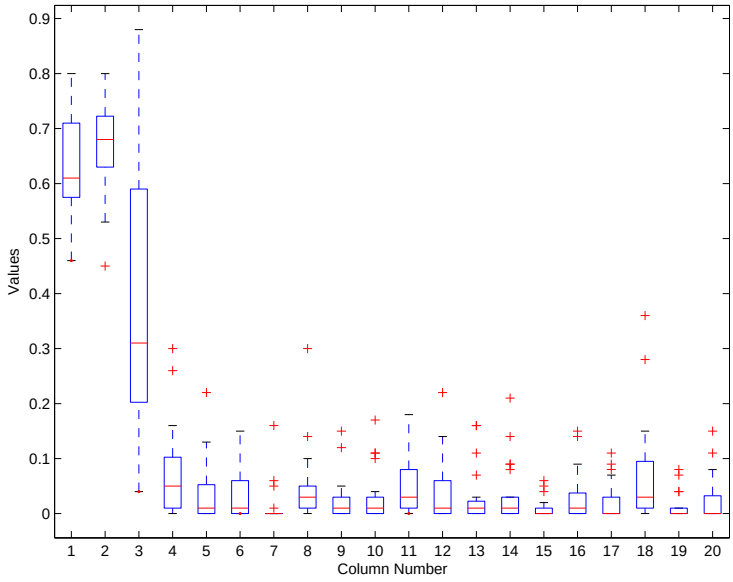

Figure A.17: Box plot of testing example 4184 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

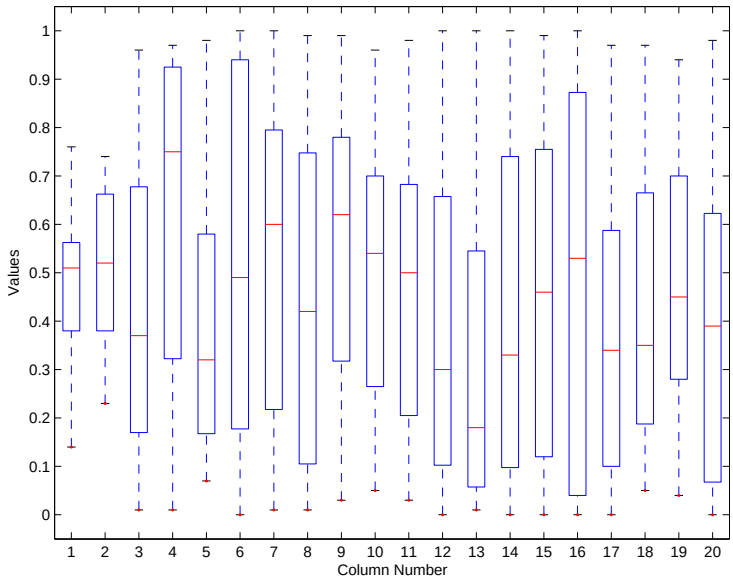

Figure A.18: Box plot of testing example 1746 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

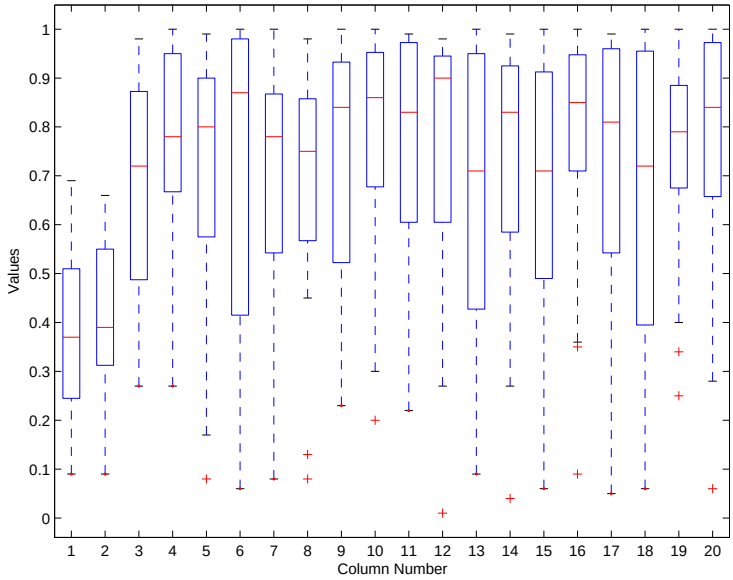

Figure A.19: Box plot of testing example 2055 along iteration of MCMC samplings. Values on the vertical axis is the ARD values while each column represents 25 iterations of ARD values.

Appendix B

A Biased Design

What we have mentioned in section 6 is an unbiased design. However, before coming out with the design, we have another methodology. From the results, we realize that it is actually bias. This problem highlights the significance of selection bias (Ambroise and McLachlan, 2002) in Microarray Dataset or any small dataset where number of examples is small (less than 100 examples). We will briefly discuss this in the following section.

B.1 Biased Test Accuracy

If the notation is not defined, we will follow the definition that used in section 6.2 to discuss this methodology. For this methodology, we are using ten-fold cross validation. The data is split into 10 folders which are K1, K2,..., K10. For first external cross-validation run, R1, we will have K1 as external testing data, and the rest of the data (let us denote it as K_{-1})be the training data.

We apply filter method (information gain and fisher score) on the K_{-1} data to choose 100 features (or a bit more if the score of features are the same). The chosen 100 features will be fed into Gaussian Processes to find the ranking of the features based on ARD values. We will then split K_{-1} to ten folders for internal cross validation. Thus, we will have k1, k2 ..., k10. For first internal cross validation run, r1, we will have k1 as internal testing data and the rest of the data (which

is denoted as k_{-1}) be the internal training data. And based on the ARD ranking, we will have k1 with different number of features, from 5 features to 20 features.

Thus, after 10 internal runs (which are r1, r2, ..., r10) of internal cross validation, we will obtain internal cross validation performance for different number of features. Take note that for this methodology, the features of training and testing data for all internal runs (r1 to r10) of R1 will be similar. In other words, train5 of r1 will have the same features as train5 in r2. This is not the same as the unbiased test accuracy methodology in section 6.2.

After R1, we will obtain an optimal number of feature. Based on this optimal features, we will create testing and training data based on K1 and K_{-1} respectively. This is repeated for different external cross validation runs (i.e. R2, to R10). With the completion of the external cross validation, we will obtain a test accuracy. Take notes that for each external cross validation run, we may have different optimal features.

Table B.1: Results of the biased test accuracy methodology for Breast Cancer Dataset-Fisher Score and ARD

Number of Features	Internal Cross Validation Test Accuracy
19	100
12	100
19	100
13	100
12	100
13	100
11	97.4
16	100
20	100
17	100
	External Cross Validation Test Accuracy
	79.5

Table B.1 to Table B.8 show the results of the bias test accuracy methodology. It can be observed that there are significant difference in all results of the internal cross validation and the external cross validation. All the internal cross validation are having a better accuracy than that of external cross validation. This is due to the selection bias in the internal cross validation.

Table B.2: Results of the biased test accuracy methodology for Breast Cancer Dataset-Information Gain and ARD

Number of Features	Internal Cross Validation Test Accuracy
17	100
12	97.4
18	97.4
10	94.9
11	97.5
13	100
13	95.0
15	97.5
16	100
17	100
	External Cross Validation Test Accuracy
	70.5

Table B.3: Results of the biased test accuracy methodology for Colon Cancer Dataset-Fisher Score and ARD

Number of Features	Internal Cross Validation Test Accuracy
3	89.5
7	96.5
18	93.0
13	91.2
17	93.1
17	91.3
16	93.1
9	89.7
10	94.8
10	93.1
	External Cross Validation Test Accuracy
	81.3

Table B.4: Results of the biased test accuracy methodology for Colon Cancer Dataset-Information Gain and ARD

Number of Features	Internal Cross Validation Test Accuracy
12	93.0
16	94.7
5	91.2
13	91.2
20	93.1
11	93.1
10	89.7
16	93.1
19	93.1
19	89.7
	External Cross Validation Test Accuracy
	80.6

Table B.5: Results of the biased test accuracy methodology for Leukaemia Dataset-Fisher Score and ARD

Number of Features	Internal Cross Validation Test Accuracy
9	95.3
13	93.8
5	93.8
11	89.2
13	93.8
18	95.4
13	89.2
17	92.3
10	93.8
10	89.2
	External Cross Validation Test Accuracy
	66.7

Table B.6: Results of the biased test accuracy methodology for Leukaemia Dataset-Information Gain and ARD

Number of Features	Internal Cross Validation Test Accuracy
13	89.1
13	87.5
16	84.6
20	86.1
19	87.7
17	86.1
15	92.3
17	89.2
16	92.3
19	89.2
	External Cross Validation Test Accuracy
	72.2

Table B.7: Results of the biased test accuracy methodology for Ovarian Cancer Dataset-Fisher Score and ARD

Number of Features	Internal Cross Validation Test Accuracy
15	87.5
15	83.3
15	83.3
7	85.4
12	87.8
18	85.7
13	83.7
14	83.7
16	87.8
14	83.7
	External Cross Validation Test Accuracy
	72.2

Table B.8: Results of the biased test accuracy methodology for Ovarian Cancer Dataset-Information Gain and ARD

Number of Features	Internal Cross Validation Test Accuracy
19	89.6
11	85.4
13	89.6
18	87.5
16	89.8
17	87.8
13	85.7
12	83.7
12	83.7
12	83.7
	External Cross Validation Test Accuracy
	72.2

Let us take a closer look at R1, we have applied filter method and ARD on all the folders (k1 to k10, or K_{-1}) before we create the training and testing data based on the ARD ranking for internal cross validation. This is not a correct way as we have allowed the testing data in the internal cross validation to be learnt by the feature selection methods. Since the data is learnt beforehand, the results of the internal cross validation is better than external cross validation. During R1, K1 is not used for any training in the internal cross validation.

Generally, this problem is not significant if there are a lot of examples. However, in view of the small number of examples to be learnt in the DNA Microarray datasets, the effect of selection bias will be significant. We will end up choosing the wrong number of optimal features due to this bias.

The biased test accuracy methodology is discussed here and not used for further comparison. Yet, it verified that the concern of Ambroise and McLachlan (2002) does valid. This problem is overcome in the unbiased test accuracy.

Appendix C

Applying Principal Component Analysis on ARD values

In Gaussian Processes, ARD values (Θ) are the products of MCMC samplings on the equilibrium distribution. From the discussion in section 3.2, the ARD values contain precious information on the features of the original input data. Besides knowing that it is a measure of the importance of a feature, we try to understand whether there is other information that we can cultivate from the ARD values. For this purpose, we have used the Principle Components Analysis.

The dataset we used is Robotic Arm dataset. The dataset consists of two features as its inputs and one output. There are 200 number of training examples and 200 testing examples. The data can be found in:

$http://www.inference.phy.cam.ac.uk/mackay/Bayes_FAQ.html\sharp Data$

We used a regression problem as the simulation would be much faster than that of a classification problem. A huge data can be obtained instantly for further study. This dataset consists of 2 features (relevant features) originally. We have added two redundant features and two irrelevant features. Different noise are added in for the redundant features.

We will only discuss mainly two datasets that we generated based on Robotic Arm dataset. One is without noise when we generate the redundant features, which means the two features are the exact duplicates of the relevant features.

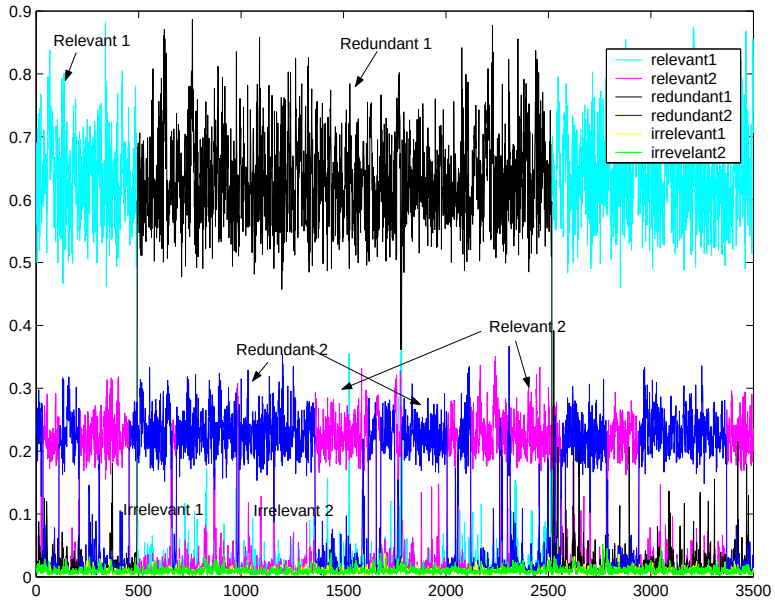

Figure C.1: ARD values for Robotic Arm dataset without noise. Only iterations after equilibrium distribution reached are shown here. Take note that the vertical axis is the ARD values while the horizontal axis is the number of iterations.

Another one is with gaussian noise of 0.04 added when we create the redundant features. The ARD of the two datasets are shown in Figure C.1 and Figure C.2 respectively.

The figures show that equilibrium distribution is established.

Based on the simulation, we apply the PCA on the ARD values. Table C.1 and Table C.2 summarize the output.

Table C.1: Results of PCA based on Robotic Arm without noise

Eigenvalues	Percentage	Eigenvectors					
0.18493	88.2763	−0.70957	0.00278	0.70442	−0.01016	0.01336	0.00276
0.02127	10.1526	−0.01675	0.70169	−0.03068	−0.71046	0.04010	0.00687
0.00260	1.2418	0.70417	0.03911	0.70880	−0.00787	0.01224	0.00277
0.00064	0.3099	0.01941	−0.71139	0.01151	−0.70139	0.03782	0.00495
0.00003	0.0140	−0.00046	0.00215	0.01276	−0.03777	−0.55499	−0.83090
0.00001	0.0053	−0.00065	0.00067	0.01231	−0.04131	−0.82983	0.55634

First columns of the tables are the eigenvalues of the new features. In this case, it basically describes the spreading of the data in the new feature space. The higher the value of the eigenvalue, the wider spread of the data is. Since the

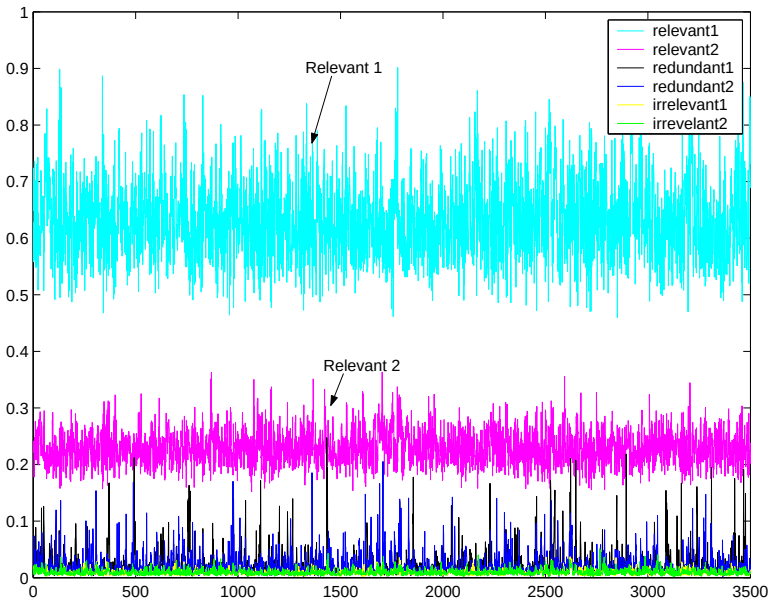

Figure C.2: ARD values for Robotic Arm dataset with noise. Only iterations after equilibrium distribution reached are shown here. Take note that the vertical axis is the ARD values while the horizontal axis is the number of iterations.

Table C.2: Results of PCA based on Robotic Arm with noise

Eigenvalues	Percentage	Eigenvectors					
0.00462	71.8279	−0.99938	−0.03071	0.00010	−0.01538	0.00738	0.00180
0.00090	14.0594	0.02999	−0.98451	−0.16964	0.02613	0.01956	0.00240
0.00054	8.4008	0.01044	−0.16776	0.93415	−0.30982	0.05457	0.01210
0.00033	5.1356	−0.01302	−0.02570	0.30394	0.94733	0.09485	0.01914
0.00002	0.3826	−0.00557	−0.02086	0.05520	0.05157	−0.52767	−0.84581
0.00001	0.1937	−0.00536	−0.02383	0.05624	0.05478	−0.84211	0.53300

spreading is huge, there are more information in this particular space. Meanwhile, the lower value the eigenvalue of the new space, the smaller the spread. It may mean that the data are all cluster near one point.

The second column is the percentage of the particular eigenvalue based on the sum of eigenvalues. Eigenvector is how the feature space is being transpose to the new feature space.

From the table, we can observe that when there are redundant features, the PCA may fail to identify them. This is mainly due to the values of ARD. The ARD values may (with noise) or may not (without noise) totally abandon redundant features.

However, we can observe that irrelevant features can be detected in ARD values. From both tables, eigenvalues of the $5th$ and $6th$ new features are very small. And the two new features are only made up by the two irrelevant features. In short, the two irrelevant features will form a two dimension space which are orthogonal and the values of the two dimensions clustered at a relatively small value. Visually, we can treat it as a thin plate in hyper-plane. It is thin as the variance is small in value.

What we have learnt from here may not be useful in this project. Further study may be required. Also, other unsupervised methods may be used as well for exploring the further understanding of ARD values.