CHAPTER 5

RECENT ADVANCES IN STATISTICAL METHODS FOR MICROARRAY DATA ANALYSIS

5.1 INTRODUCTION

The field of high throughput genetic experimentation is evolving rapidly, with the advent of new technologies. Bayesian methods played a central role to the future of data and knowledge integration in the field of Bioinformatics and microarray technology. Lee et al., (2003) proposed a hierarchical Bayesian model and employed latent variables to specialize the model to a regression setting and applied variable selection to select differentially expressed genes. Bae and Mallick (2004) proposed an alternative method to perform Bayesian gene selection using a two level hierarchical Bayesian model. A comprehensive comparative study on several discriminant methods in cancer classification can be found in Dudoit et al., (2000).

Newton et al.,(2004) used a hierarchical model that consists of two levels. The first level of the model describes the conditional distributions of the observed measurement of gene expression given the means of the control ($\mu_1$) and the treatment ($\mu_2$), where $i$ indexes the gene. The second level describes the distribution of $\mu_1$ and $\mu_2$ jointly by a mixture of three multivariate distributions with constrained parameters of $\mu_1 > \mu_2$, $\mu_1 = \mu_2$, and $\mu_1 < \mu_2$. Genes become linked by virtue of having $\mu_1$ and $\mu_2$ drawn from a common distribution. Parameter estimation was accomplished via the Expectation Maximization (EM) algorithm.

5.2 BAYESIAN APPROACHES

A sophisticated statistic called SAM (statistical analysis of Microarray data) proposed by Tusher et al.,(2001) slightly moderates the t statistics by adding a suitable constant in the denominator of the t statistic. SAM can be thought of as Bayesian t statistic, where the denominator is a sum of a global, prior standard error and gene specific standard error of the genes average log ratio. Baldi and Long (2001) proposed a slightly different empirical Bayes framework model for each channel ($\log 2R$ and $\log 2G$) separately, using normal distributions with conjugate priors (normal and inverse gamma). This results gives two posterior models for each
gene and posterior probability that the two location parameters are equal, \( P(\log_2 R = \log 2G/Data) \).

The framework of Bayesian hierarchical model refers to a generic model building strategy in which unobserved quantities are organized into a small number of discrete levels with logically distinct and scientifically interpretable functions and probabilistic relationships between them that capture inherent features of the data. It is of course important to perform some basic exploration and visualization of the data before formulating complex models. Frigessi et al.,(2005) discussed a Bayesian model for obtaining estimates of concentrations of RNA from two color arrays. Broet et al., (2002) present a full Bayesian hierarchical model which allows the genes to be assigned to one of an unknown number of expression levels, the genes assigned to a given level are modeled to have the same mean and variance. Lonnstedt (2005) proposed mixture models of regulated and unregulated genes, where the former as well as the latter in the second model are allowed gene specific means and variances. This simplifies the numerical evaluation and maintains a low number of hyper parameters.

5.2.1 Priors

Baldi and long (2001) discussed full Bayesian treatment which requires introducing a prior distribution \( P(\mu, \sigma^2) \). The choice of a prior is part of the modeling process and several alternatives are possible (Box and Tiao, 1973; Pratt et al., 1995) which is a sign of the flexibility of the Bayesian approach rather than its arbitrariness. Several kinds of priors for the mean and variance of a normal distribution have been studied in the literature, including the non informative improper prior and the conjugate prior. For microarray data, the conjugate prior seems to be more suitable and flexible, not only because of its convenient form, but also because it incorporates the basic observation that \( \mu \) and \( \sigma^2 \) are typically not independent. When both the prior and the posterior have the same functional form, the prior is said to be a conjugate prior. When estimating the mean alone of a normal model of known variance, the obvious conjugate prior is also a normal distribution. In the case of dye models for biological sequences, the standard conjugate prior is a Dirichlet distribution (Baldi and Brunak, 2001). The form of the likelihood in equation (5.2) in which the measurements of each gene in each situation of treatment
or control is with a normal distribution $N(x; \mu, \sigma^2)$. For each gene and each condition, we have a two parameter model $w = (\mu, \sigma^2)$, and by focusing on one such model we can omit indices identifying the gene or the condition and $c$ denotes the normalizing constant of any distribution.

$$P(D|\mu, \sigma^2) \approx \prod_{i=1}^{n} N(x_i; \mu, \sigma^2)$$  

$$= C(\sigma^2)^{-n/2} e^{-(n(m-\mu)^2 + (n-1)s^2/2\sigma^2)}$$

This equation (5.2) shows that the conjugate prior density must have the form $P(\mu|\sigma^2)P(\sigma^2)$, where the marginal $P(\sigma^2)$ is scaled as inverse gamma and the conditional distribution $P(\mu|\sigma^2)$ is normal. This leads to a hierarchical model with a vector of four hyper parameters for the prior $\alpha = (\mu_0, \lambda_0, \nu_0, \sigma^2_0)$ with the densities:

$$P(\mu|\sigma^2) = N(\mu; \mu_0, \sigma^2/\lambda_0)$$

$$P(\sigma^2) = I(\sigma^2; \nu_0, \sigma^2_0)$$

The expectation of the prior is finite if and only if $\nu_0 > 2$. The prior $P(\mu, \sigma^2|\alpha) = P(\mu, \sigma^2|\alpha)$ is given by

$$C\sigma^{-1}(\sigma^2)^{(\nu_0/2+1)} \exp \left[ -\frac{\nu_0}{2\sigma^2} \sigma_0^2 - \frac{\lambda_0}{2\sigma^2} (\mu_0 - \mu)^2 \right]$$

Notice that it makes perfect sense with array data to assume a priori that $\mu$ and $\sigma^2$ are dependent, as suggested immediately by visual inspection of typical microarray data sets.

### 5.3 Gene Selection Methods

Let us assume that there are $p$ predictor genes, defined as $x_1, x_2, ..., x_p$, where $x_i = (x_{i1}, x_{i2}, ..., x_{in})'$ represents the gene expression profile of gene $i$ across $n$ samples from different conditions. The aim was to select potential predictor genes, which can well discriminate the samples under different conditions. A binary response variable $y$ was introduced where $y_i = 1$, if the sample is from group 1 and $y_i = 0$ indicates the sample is from group 2. A standard normal linear model is used to describe the relationship between the observed dependent variable and the set of all predictors $x_1, x_2, ..., x_p$. In statistical terms
\[ g(y_i = 1/x_i\beta) = \phi(x_i^T \beta) \]  

(5.6)

where \( \beta \) is a \( p \times 1 \) vector of unknown regression coefficients and \( \phi(.) \) denotes the standard normal distribution function. The likelihood is given by

\[
\pi (y/\beta, X) = \prod_{i=1}^{p} \Phi (x_i^T \hat{\beta})^{y_i} \{1-\Phi(x_i^T \hat{\beta}) \}^{1-y_i}
\]

(5.7)

where \( X = [x_1, ..., x_p] \) is an \( n \times p \) matrix, by augmenting the observed data matrix \( \{y_i, x_i\} \) with latent variable \( z_i \). Albert and Chib (1993) discussed that the probit model can be expressed in hierarchical form as, \( y_i = 1 \) if \( z_i > 0 \) and

\[
Z_i \sim N(x_i^T \beta, 1)
\]

(5.8)

Gene selection was performed using the algorithm given by Lee et al., (2003) where a \( p \times 1 \) vector \( \gamma = (\gamma_1, \gamma_2, ..., \gamma_p) \) of indicator variable was introduced where \( \gamma_i = 0 \), if the gene is not selected (\( \beta_i = 0 \)) and \( \gamma_i = 1 \), if the gene is selected (\( \beta_i \neq 0 \)). The Bayesian variable selection was to estimate \( \gamma \) from the posterior distribution of \( p(\gamma|z) \), where \( z = (z_1, z_2, ..., z_n)' \). The Gibbs sampling algorithm for estimating \( \{\alpha, \beta, z\} \) is summarized as follows

\[
S(\gamma, z) = z' - \frac{c}{1+c} X'_\gamma (X'_\gamma X_\gamma)^{-1} X'_\gamma z
\]

(5.9)

where \( X_\gamma \) refers to selected columns of \( X \). By straightforward computation the posterior distribution of \( p(\gamma|z) \) is approximated by

\[
p(\gamma|z) \propto p(z|\gamma) p(\gamma) \propto \exp\{-\frac{1}{2} S(\gamma, z)\} \prod_{i=1}^{p} \pi_\gamma \{1 - \pi_i\}^{1-\gamma_i}
\]

(5.10)

where \( \Pi = p(\gamma=1) \) are the probabilities to select \( i^{th} \) gene. Given \( \gamma \), and let \( \beta_\gamma \) consists of all nonzero elements of \( \beta \). The prior distribution of \( \beta_\gamma \) is \( N(0, c(X'_\gamma X_\gamma)^{-1}) \), where \( c \) is a constant and the value of \( c \) can be decided for specific study, (for example \( c = 100 \)). Then the posterior distribution of \( p(\beta|z) \) is

\[
N(V_\gamma X_\gamma Z_\gamma, V_\gamma)
\]

(5.11)

where

\[
V_\gamma = \frac{c}{1+c} (X'_\gamma X_\gamma)^{-1}
\]

(5.12)
and index $\gamma$ refers to all the elements corresponding to $\gamma_i = 1$. To implement Gibbs sampling, sampling was done iteratively from these full conditional posterior distributions large number of times. An initial burn in samples were discarded and inferences were drawn based on the posterior summaries calculated using the joint posterior distribution.

Lonnstedt and Speed (2002) proposed $B$ statistic for the selection of differentially expressed genes in replicated microarray experiments. $B$ is a log posterior odds of differential expression, based on the log ratios $M_{ij}$, $i = 1, \ldots, N$ and $j = 1, \ldots, n$ for $N$ genes each represented on $n$ replicate microarray slides. The $M_{ij}$ are regarded as normally distributed random variables,

$$M_{ij} | \mu_i, \sigma_i \sim N(\mu_i, \sigma_i^2) \quad (5.13)$$

Given the parameters all genes and replicates are assumed independent. This is of course not true but provides a pragmatic basis for modeling the data. Most genes have $\mu_i = 0$, but a small proportion $p$ of genes have $\mu_i \neq 0$, indicated by $I_i = 1$ as opposed to $I_i = 0$. The log posterior odds for gene $g$ to be differentially expressed is calculated as

$$B_g = \log \left( \frac{\text{Pr}(I_g = 1|M)}{\text{Pr}(I_g = 0|M)} \right) \quad (5.14)$$

By assuming conjugate prior distributions for the variances of the genes [inverse gamma, $n\beta^2/2\sigma_i^2 \Gamma(\alpha, 1)$], as well as their means are not zero $[\mu_i | \sigma_i^2, I_i \neq 0 \sim N(0, c \sigma_i^2)]$, we obtain an explicit formula for $B$

$$B_g = \log \left( \frac{p}{1-p} \cdot \frac{1}{\sqrt{1+nc}} \cdot \frac{\beta + S_g^2 + M_g^2}{\beta + S_g^2 + \frac{M_g^2}{1+nc}} \right)^{\frac{\alpha + n}{2}} \quad (5.15)$$

The use of $B$ provides a ranking among the genes based on high relative expression level and explicability. It rules out the groups of genes that are easily falsely selected if only the average expression or a $t$ statistic was used.

5.4 MARKOV CHAIN MONTE CARLO METHODS

A major limitation towards more wide spread implementation of Bayesian approaches is that obtaining the posterior distribution often requires the integration of high dimensional functions. This is computationally very difficult. So
we use Markov chain Monte Carlo methods which attempts to simulate direct draws
from some complex distribution of interest. Markov Chain Monte Carlo (MCMC)
methods were developed to facilitate the calculation of posterior distributions. The
MCMC approach got its name because one uses the previous sample values to
randomly generate the next sample value, thus producing a Markov chain. Calculation
of the posterior distribution involves evaluation of complex multidimensional
integrals; an efficient way to deal with this integration is by employing Monte Carlo
approximation. The Gibbs sampler is a Bayesian method that replaces the problem of
calculating the posterior distribution with iteratively sampling from the full
conditional distribution (Casella and George, 1997). The most commonly used
algorithm in MCMC application are metropolis hasting algorithm and Gibbs
sampling.

5.4.1 Metropolis Hasting algorithm

Metropolis Hastings algorithm is a Markov Chain Monte Carlo (MCMC)
method for obtaining a sequence of random samples from a probability
distribution for which direct sampling is difficult. This sequence can be used to
approximate the distribution (i.e., to generate a histogram) or to compute an
integral (such as an expected value). One of the major task in Monte Carlo
integration method is obtaining samples from complex probability distribution \( p(x) \).
Attempts to solve this problem are the roots for MCMC methods. Metropolis et al.,
(1953) realized that it was non obvious to calculate ergotic limit for model of
physical system and provide an algorithm constructing a markov chain having a
specified equilibrium distribution. The Metropolilic algorithm is generalized by

Metropolis Hastings and other MCMC algorithms are generally used for
sampling from multi dimensional distributions, especially when the number of
dimensions is high. Suppose to draw samples from some distribution \( p(\theta) \), where
\( p(\theta) = f(\theta)/K \), where the normalizing constant \( K \) may not be known and very difficult
to compute. The Metropolis algorithm (Metropolis and Ulam, 1949; Metropolis et
al., 1953) generates a sequence of draws from this distribution is as follows:
1. Start with any initial value $\theta$ satisfying $f(\theta_0) > 0$.

2. Using current $\theta$ value, sample a candidate point $\theta^*$ from some jumping distribution $q(\theta_1; \theta_2)$, which is the probability of returning a value of $\theta_2$ given a previous value of $\theta_1$. This distribution is also referred to as the proposal or candidate generating distribution. The only restriction on the jump density in the Metropolis algorithm is that it is symmetric, That is $q(\theta_1; \theta_2) = q(\theta_2; \theta_1)$.

3. Given the candidate point $\theta^*$, calculate the ratio of the density at the candidate ($\theta^*$) and current ($\theta_{t-1}$) points. It can be notice that considering the ratio of $p(x)$ under two different values, the normalizing constant $k$ cancel out.

$$\alpha = \frac{p(\theta^*)}{p(\theta_{t-1})}$$  \hspace{1cm} (5.16)

4. If the jump increases the density ($\alpha > 1$) accept the candidate point ($\theta_t = \theta^*$) and return to step 2. If the jump decreases the density ($\alpha < 1$), then the probability $\alpha$ accept the candidate point else reject it and returned to step 2. The metropolis sampling as first computing can be summarize

$$\alpha = \min \left( \frac{f(\theta^*)}{f(\theta_{t-1})}, 1 \right)$$  \hspace{1cm} (5.17)

And accepting a candidate point with probability $\alpha$ (the probability of a move). This generate a markov chain ($\theta_0, \theta_1, \ldots, \theta_k, \ldots$) as the transition probabilities from ($\theta_t$ to $\theta_{t-1}$). Following a sufficient burning period, the chain approaches its stationary distribution, samples from the vector ($\theta_k+1, \ldots, \theta_{k+n}$) are samples from $p(x)$. Hastign(1970) generated the metropolis algorithm by using an arbitrary transition probability function $q(\theta_1, \theta_2) = \Pr(\theta_1 \rightarrow \theta_2)$ and setting acceptance probability for a candidate points as

$$\alpha = \min \left( \frac{f(\theta^*)q(\theta^*, \theta_{t-1})}{f(\theta_{t-1})q(\theta_{t-1}, \theta^*)}, 1 \right)$$  \hspace{1cm} (5.18)

This is the Metropolis Hastings algorithm. Assuming that the proposed distribution is symmetric, That is $1q(x,y) = q(y,x)$ recovers the original metropolis algorithm.

### 5.4.2 Gibbs Sampling

The Gibbs sampler introduced in the context of image processing by Geman in 1984. It is a special case of Metropolis Hastings sampling wherein the random value is always accepted. The task remains to specify how to construct a Markov Chain whose values converge to the target distribution. The key to the Gibbs sampler
is that one only considers univariate conditional distributions the distribution when all of the random variables but one are assigned fixed values. Such conditional distributions are far easier to simulate than complex joint distributions and usually have simple forms (often being normal, inverse $\chi^2$, or other common prior distributions). Thus, one simulates $n$ random variables sequentially from the $n$ univariate conditionals rather than generating a single $n$ dimensional vector in a single pass using the full joint distribution. To introduce the Gibbs sampler, consider a bivariate random variable $(x, y)$ and suppose we wish to compute one or both marginal, $p(x)$ and $p(y)$. The idea behind the sampler is that it is far easier to consider a sequence of conditional distributions, $p(x|y)$ and $p(y|x)$, than it is to obtain the marginal by integration of the joint density $p(x,y)$, e.g., $p(x) = \int p(x,y)dy$. The sampler starts with some initial value $y_0$ for $y$ and obtains $x_0$ by generating a random variable from the conditional distribution $p(x|y = y_0)$. The sampler then uses $x_0$ to generate a new value of $y_1$, drawing from the conditional distribution based on the value $x_0, p(y|x = x_0)$. The sampler proceeds as follows

$$x_i \sim p(x|y = y_{i-1}) \quad (5.19)$$

$$y_i \sim p(y|x = x_i) \quad (5.20)$$

Repeating this process $k$ times, generates a Gibbs sequence of length $k$, where a subset of points $(x_j; y_j)$ for $1 \leq j \leq m < k$ are taken as our simulated draws from the full joint distribution. (One iteration of all the univariate distributions is often called a scan of the sampler. To obtain the desired total of $m$ sample points (here each point on the sampler is a vector of the two parameters), one samples the chain (i) after a sufficient burn in to removal the effects of the initial sampling values and (ii) at set time points (say every $n$ samples) following the burn in. The Gibbs sequence converges to a stationary (equilibrium) distribution that is independent of the starting values and by construction this stationary distribution is the target distribution we are trying to simulate (Tierney, 1994).

When more than two variables are involved, the sampler is extended in the obvious fashion. In particular, the value of the $k^{th}$ variable is drawn from the distribution $p(\theta^{(k)}|\mathcal{O}^{(-k)})$ where $\mathcal{O}^{(-k)}$ denote a vector containing all of the but $k$,
thus t during the i\textsuperscript{th} iteration of the sample, to obtain the value of $\theta_i^k$, draw from distribution
\[
\theta_i^k \sim p(\theta^{(k)}|\theta^{(1)} = \theta_i^{(1)}, \ldots, \theta_i^{(k-1)} = \theta_i^{(k-1)}, \theta_i^{(k+1)} = \theta_i^{(k+1)} \ldots, \theta_i^{(n)} = \theta_i^{(n)}) \tag{5.21}
\]

Gelfand and Smith (1990) illustrated the power of Gibbs sampler to address a wide variety of statistical issues, while Smith and Roberts (1993) showed the importance and application of the gibbs sampler with Bayesian statistics in obtaining posterior distributions. A number of researches described and discussed the application of gibbs sampler in their research papers (Casella and George, 1992; Tanner, 1996; Besag et al., 1995 and Lee, 1997). The Gibbs sampler can be thought of as a stochastic analog to the expectation maximization approaches used to obtain likelihood functions when missing data were present. In that case, the sampler random sampling replaces the expectation and maximization steps.

Hein et al., (2005) present a fully Bayesian hierarchical model, estimated by MCMC, for obtaining gene expression measures for each gene in each experimental condition. If there are replicate samples for a condition (including biological replicates) the model produces an estimate for that condition, rather than separate estimates for each replicate. On the other hand, by using the variability of the probe sets for each gene, the model can be used with a single array for each condition and meaningful comparison between conditions without any replicates can be achieved (Hein and Richardson, 2006). Lee (2006) used a combination of truncated sampling and Markov Chain Monte Carlo (MCMC) based computation techniques to simulate the posterior distribution in microarray data analysis. The Bayesian model is flexible enough to identify the significant genes as well as to perform future predictions. The method is applied to cancer classification via c DNA microarrays (Lee, 2006).

If we have binary responses data $Y_i$, where $Y_i = 1$ indicates the tumor sample $i$ is class 1 and $Y_i = 0$ indicates it is class 2, for $i = 1, \ldots, n$. For each sample, we have a measurement of the expression levels for all the genes, so $X_{ij}$ is the measurement of the expression level of the $j\textsuperscript{th}$ gene for the $i\textsuperscript{th}$ sample where $j = 1, \ldots, p$. Because of the binary nature of the data, couldn’t compute the posterior distribution in explicit form. In such situation Gilks et al., (1996) use a MCMC method specifically Gibbs
sampling (Gelfand and Smith, 1990) to generate the posterior distributions of parameters (Lee et al., 2006).

The MCMC algorithm used in the gene selection and analysis process by Lee et al., (2006) as follows.

Step 1: Start with initial values \([\gamma(0), Z(0), \beta(0)]\)

Step 2: At the \(t^{th}\) iteration, Draw \(\gamma(t)\) from \(p(\gamma|Z(t-1))\).

Step 3: Draw \(Z(t)\) from \(p(Z|\beta(t-1), \gamma(t))\).

Step 4: Draw \(\beta(t)\) from \(p(\beta|Z(t), \gamma(t))\).

Step 5: Let \(t \leftarrow t + 1\), Continue required number of iterations.

For decision making we calculate the relative number of times each gene appeared in the MCMC sample (number of time the corresponding \(\gamma\) is 1). This will give us an estimate of the posterior probability of inclusion of that gene and tell the relative importance of the gene for classification purposes. We can also obtain the predictive classification of a new observation \(Y_{new}\) condition on the expression levels as

\[
P(Y_{new} = 1|X) = \int_{\gamma} \int_{Z} \int_{\beta} P(Y_{new} = 1|X,Z,\beta,\gamma)P(Z,\beta,\gamma|Y) \, dZ \, d\beta \, d\gamma)
\]

(5.22)

and the Monte Carlo estimate of this probability will be

\[
\hat{P}(Y_{new} = 1|X) = \frac{1}{m} \sum_{i=1}^{m} P(Y_{new} = 1|X, Z^{(t)}, \beta^{(t)}, \gamma^{(t)})
\]

(5.23)

Jia and Xu (2007) described the Bayesian clustering method in which, the observed gene expression levels are described by a regression model as done by Gusnanto et al., (2005). For each gene, the irrelevant intercept is removed by a special normalization scheme. The slope of the regression represents the difference of expression under the two conditions. The Bayesian method is implemented via the Markov chain Monte Carlo (MCMC) algorithm. The regression coefficient of each gene is assumed to be sampled from a mixture of three normal distributions with constrained parameters. The three distributions are \(N(\beta_k, \nu_k)\) for \(k = 1, 2, 3\), where \(\beta_1 < 0, \beta_2 = 0,\) and \(\beta_3 > 0\) are the constrained parameters. The proposed method in his study turns the problem of a complicated multivariate mixture distribution of Newton et al., (2004) into that of a univariate mixture distribution.
5.5 HIDDEN MARKOV MODELS

The hidden Markov model (HMM) was first introduced by Baum and his colleagues. It is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved states. HMMs were first applied to speech recognition in the early 1970s. The use of HMMs in speech recognition is reported by Levinson et al., (1983) and Juang and Rabiner (1991). A number of researchers reported the applications of HMMs in the field bioinformatics (Thompson, 1983; Churchill, 1989, 1992; Guttorp, Newton and Abkowitz, 1990; Baldi, 1994 and Krogh 1994). Another area in application of HMMs is financial time series modelling such as the stock market. Ryden (1998) used HMM to model temporal and distributional properties of daily data from speculative markets. Elliott and van der Hoek (1997) applied HMM to asset allocation problems.

5.5.1 Basic HMM Model

A hidden Markov model is a stochastic process in which an underlying stochastic process that is not observable (it is hidden) can only be observed through another stochastic process that produces a sequence of observations. Thus, if \( S = \{ S_n \mid n = 1, 2, \ldots \} \) is a Markov process and \( \Omega = \{ \Omega_k \mid k = 1, 2, \ldots \} \) is a function of \( S \), then \( S \) is a hidden Markov process or hidden Markov model that is observed through \( \Omega \), and we can write \( \Omega_k = f(S_k) \) for some function \( f \). Here \( S \) is the state process that is hidden and \( \Omega \) as the observation process that can be observed. A hidden Markov model is usually defined as 5-tuple \(( S, \Omega, P, \Theta, \pi )\), where \( S = \{ s_1, s_2, \ldots, s_N \} \) is a finite set of \( N \) states, \( \Omega = \{ o_1, o_2, \ldots, o_m \} \) is a finite set of \( m \) possible symbols, \( P = \{ P_{ij} \} \) is a set of state transition probabilities, where \( P_{ij} \) is the probability that the system goes from a state \( S_i \) to state \( S_j \), \( \Theta = \{ \phi_i(o_k) \} \) are the observation probability that the symbol \( o_k \) is emitted when the system is in state \( S_j \), \( \pi = \{ \pi_i \} \) are the initial state probabilities, that is \( \pi_i \) is the probability that the system states starts in the state \( s_i \). The parameter of HMM can be denoted by \( \lambda = ( P, \Theta, \pi ) \). Fundamental problems in hidden Markov Models are evaluation problem, decoding problem and learning problem which are described as follows.
1. The evaluation problem: Given a model \( \lambda = (P, \Theta, \pi) \) and an observation sequence \( O = v_1, v_2, \ldots, v_T \) of length \( T \), where \( v_i \in \Theta \). Then we can evaluate, how efficiently we compute the probability that the model is generating the observation sequence; that is denoted by \( P[O|\lambda] \).

2. The decoding problem: Given a model \( \lambda = (P, \Theta, \pi) \), We can find out the most likely sequence of hidden states that could have generated a given observation sequence. In these case, we have to calculate \( Q^* = \arg \max_Q P[Q,O|\lambda] \), where \( Q \) is the hidden state.

3. The learning problem: Given a set of observation sequences and to find the HMM that best explains the observation sequences, that is to find this value of \( \lambda \) that maximize \( P[O|\lambda] \) or \( \lambda^* = \arg \max_{\lambda} P[O|\lambda] \). In other wise, we can say that the problem is to estimate the most likely HMM parameters for a given observation sequence.

### 5.5.2 Algorithms for solving HMM problems

There are different types of algorithm available for solving HMM problems. The evaluation problem is usually solved by the forward algorithm and the backward algorithm. The decoding problem can be solved by the Viterbi algorithm. And the learning problem is solved by the Baum Welch algorithm.

#### 5.5.2.1 The Forward Algorithm

One important observation in the calculation of \( P[O|\lambda] \) by the direct method is that it requires many redundant calculations that are not saved and reused. The induction step of algorithm can be illustrated in Figure 5.1. A trellis can record the probability of all initial sub paths of the HMM that end in a certain state at a certain time. This allows the probability of longer sub paths to be worked out in terms of shorter sub paths. A forward probability variable \( \alpha_t(i) \) is defined as follows:

\[
\alpha_t(i) = P[o_1, o_2, \ldots, o_t, q_t = s_i|\lambda] \quad t = 1, \ldots, T; \quad i = 1, \ldots, N
\]

That is, \( \alpha_t(i) \) is the probability of being in state \( s_i \) at time \( t \) after having observed the sequence \( \{o_1, o_2, \ldots, o_t\} \).
Figure 5.1: Induction step of the forward algorithm

The forward algorithm works as follows:

1. Initialization: \( \alpha_t(i) = \pi_i \phi(o_1) \) for \( 1 \leq i \leq N \)

2. Induction:
   \[
   \alpha_{t+1}(j) = \sum_{i=1}^{N} P_{ij} \alpha_t(i) \phi_j(o_{t+1})
   \]
   for \( 1 \leq t \leq T-1, 1 \leq j \leq N \)

3. Update time: Set \( t = t+1 \), if \( t < T \), go to step 2, otherwise go to step 4

4. Termination: \( P[O|\lambda] = \sum_1^T \alpha_T(i) = \sum_1^T P(O_t, q_{t+1} = S_i | \lambda) \)

5.5.2.2 The Backward Algorithm

The backward algorithm is a dual method to solve the evaluation problem. It starts by defining a backward probability variable \( \beta_t(i) \) which is as follows

\[
\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T | q_{t+1} = s_t, \lambda)
\]

for \( t = 1, \ldots, T; s_t \in S \)

Where \( \beta_t(i) \) is the conditional probability of the partial observation \( O_{t+1}, O_{t+2}, \ldots, O_T \) given that the model is in state \( S_i \) at time \( t \).
The main steps in backward algorithm

1. Initiation
   \[ \beta_T (i) = 1 \quad \text{where} \quad 1 \leq i \leq N \]

2. Induction:
   \[ \beta_t (i) = \sum_{i=1}^{N} P_{ij} \beta_{t+1} + (j) \phi_j (O_{t+1}) \quad , \quad 1 \leq t \leq T-1 , \quad 1 \leq i \leq N \]

   The induction step is schematically displayed in figure 5.2

3. Update time: Set \( t = t - 1 \). If \( t > 0 \), go to step 2; otherwise go to step 4.

4. Termination:
   \[ P[O|\lambda] = \sum_{i=1}^{N} \beta_t (i) \alpha_1 (i) = \sum_{i=1}^{N} \beta_1 (i) \pi \phi_1 (O_1) \]

   The forward-backward algorithm is obtained from the observation that for any \( t \), \( 1 \leq t \leq T \), it can be shown that
   \[ P[O|\lambda] = \sum_{i=1}^{N} \beta_t (i) \alpha_t (i) \]

5.5.2.3 The Viterbi Algorithm

   The Viterbi algorithm was originally designed for decoding convolutional codes and is now applied in many other areas. In HMMs, it is used to find the most likely state sequence \( Q^* = \{ q_1^*, q_2^*, \ldots, q_T^* \} \) for a given observation sequence \( O = \{ o_1, \ldots, o_T \} \).
Let the function $\arg \max_y \{z\}$ denote the argument $y$ that corresponds to the maximum of the expression $z$. The Viterbi algorithm simultaneously maximizes both the joint probability $P[q,O]$ and the conditional probability $P[q|O]$ due to the fact that

$$\arg \max_Q \{P(Q,O,\lambda)\} = \max_Q \{\frac{P[q,O,\lambda]}{P[q,\lambda]}\} = \arg \max_Q \{P(Q,O,\lambda)\} \quad (5.25)$$

The algorithm defines the variable $\delta_t(i)$ which is the largest probability along a single path that accounts for the first $t$ observations and ends in state $s_i$. Thus, it is the probability of the most likely state path for the partial observation sequence. Another variable $\psi_t(j)$ stores the node of the incoming arc that leads to this most probable path. That is,

1. Initialization:
   
   $$\delta_1(i) = \pi b_i(o_1) \quad \psi_1(i) = 0 \quad 1 \leq i \leq N$$

2. Recursion:
   
   $$\delta_t(j) = \max_{1 \leq i \leq N} \{\delta_{t-1}(i)P_{ij}\} b_j(o_t)$$
   
   $$\psi_t(j) = \arg \max_{1 \leq i \leq N} \{\delta_{t-1}(i)P_{ij}\}, \quad \forall j, 1 \leq j \leq N, \quad 2 \leq t \leq T$$

3. Update time: Set $t = t + 1$. If $t < T$, go to step 2; otherwise go to step 4.

4. Termination
   
   $$P^* = \max_{1 \leq i \leq N} \{\delta_T(i)\}$$
   
   $$q_T^* = \arg \max_{1 \leq i \leq N} \{\delta_T(i)\}$$

5. Path (or state sequence) backtracking:
   
   $$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad t = T-1, T-2, \ldots, 1$$

The backtracking step allows the best state sequence to be found from the back pointers stored in the recursion step.
Note that this step is similar to the induction step of the forward algorithm. The main difference between the two is that the forward algorithm uses summation over previous states while the Viterbi algorithm uses minimization.

5.5.2.4 The Baum Welch Algorithm

The algorithm starts by setting the parameters $P, \Theta,$ and $\pi$ to some initial values that can be chosen from some prior knowledge or from some uniform distribution. Then using the current model, all possible paths for each training set are considered to get new estimates $\hat{p}, \hat{\Theta}$ and $\hat{\pi}$. The procedure is repeated until there are insignificant changes in the parameters of the current model. As a forward backward algorithm the Baum-Welch algorithm uses the same forward probability variable $\alpha_t(i)$ and backward probability variable $\beta_t(i)$ used in the evaluation problem:

$$\alpha_t(i) = P[o_1, o_2, \ldots, o_t, q_t = s_i | \lambda]$$
$$\beta_t(i) = P[o_{t+1}, o_{t+2}, \ldots, o_T | q_t = s_i, \lambda]$$

where $t = 1, \ldots, T$; $i = 1, \ldots, N$. Recall that $\alpha_t(i)$ is the probability of being in state $s_i$ at time $t$ after having observed the sequence $\{o_1, o_2, \ldots, o_t\}$, and $\beta_t(i)$ is the conditional probability of the partial observation $o_{t+1}, o_{t+2}, \ldots, o_T$ given that the model is in state $s_i$ at time $t$. The detailed algorithm can be summarized as follows.

1. Obtain the estimate of the initial state distribution for state $s_i$ as the expected frequency with which state $s_i$ is visited at time $t = 1$; that is $\bar{\pi}_1(i) = \gamma_1(i)$
2. Obtain the estimates $\bar{P}_{ij}$ and $\bar{\phi}_j(k)$.
3. Let the current model be $\lambda = (P, \Theta, \pi)$ that is used to compute the values of $\bar{P}_{ij}$ and $\bar{\phi}_j(k)$. Then the estimated model will be $\tilde{\lambda} = (\bar{P}, \bar{\Theta}, \bar{\pi})$. Using this updated model perform a new iteration until convergence.
4. If $P(O|\tilde{\lambda}) - P(O|\lambda) < \delta$, stop, where $\delta$ is a predefined threshold value. The EM theory states that after each iteration, one of two things can happen:
   a. $\tilde{\lambda}$ is more likely than $\lambda$ in the sense that $P(O|\tilde{\lambda}) > P(O|\lambda)$, or
   b. we have reached a stationary point of the likelihood function at which $\tilde{\lambda} = \lambda$
5.5.3 Types of Hidden Markov Models

HMMs can be classified according to the nature of the distribution of the output probabilities \( \phi_i(\nu_k) \). If the observations \( \nu_k \) are discrete quantities, then \( \phi_i(\nu_k) \) are probability mass functions and the HMM is called a discrete HMM. If the observations are continuous random variables, then the HMM is called a continuous HMM. In this case, \( \phi_i(\nu_k) \) are probability density functions and we have a continuous observation space. Another model is the left to right HMM. A left to right HMM has a left to right transition to the next state as well as a self transition. The self-transition is used to model contiguous features in the same state. It is popularly used to model speech as a time sequence of distinct events that start at an initial state, which is usually labelled begin, and end at a final state, which is usually labelled end. The model is also used in profile HMM. An example of a left to right HMM is can be seen in figure 5.3 where the states labelled B and E denote Begin and End respectively of a sequence.

![Figure 5.3: Example of left to right HMM](image)

5.5.3.1 Hierarchical Hidden Markov Model

Hierarchical HMM (HHMM) was proposed by Fine (1998) to extend the standard HMM in a hierarchical manner to a hierarchy of hidden states. Alternatively, it can be considered a structured multilevel model that makes each hidden state in the standard HHMM as well. This means that each state can emit sequences rather than single symbols. There are two types of states: the normal, HMM states \( S = \{s_1, s_2, \ldots, s_N\} \), which are called production states, and internal states \( I = \{i_1, i_2, \ldots, i_M\} \) that can connect to other states but cannot produce observations. Only the production states can produce observations. There are end states at every level from where
control is returned to the immediate upper level internal state from where the transition to the sub HMM originated. That is, entering an end state causes a sub HMM to terminate, and a transition to an end state could be triggered by some environmental condition.

5.5.3.2 Factorial Hidden Markov Model

Factorial hidden Markov model (FHMM) was proposed by Ghahramani and Jordan (1997). In a regular HMM, information about the past is conveyed through a single discrete variable, which is the hidden state. FHMM permits the state to be factored into multiple state variables and is therefore represented in a distributed manner. Thus, FHMM can be used to represent a combination of multiple signals produced independently where the characteristics of each signal were described by a distinct Markov chain. For example, Kadirkamanathan and Varga (1991) used one chain to represent speech and another chain to represent some dynamic noise source. Similarly, Logan and Moreno (1998) used two chains to represent two underlying concurrent sub processes governing the realization of an observation vector in speech processing. Jacobs (2002) developed a generalized back fitting algorithm that computes customized error signals for each hidden Markov chain of an FHMM and then trains each chain one at a time using conventional techniques.

5.5.3.3 Hidden Semi Markov Models

The hidden Semi Markov process attempts to generalize the Markov process by permitting a generally distributed holding time at each state instead of the exponential or geometric holding time, the hidden semi markov model (HSMM) is an HMM in which the number of symbols emitted when the process is at a given state before it moves to a new state is a random variable. Thus, each state can emit a sequence of observations. In Ferguson (1980) and Levinson (1986), HSMM is called HMM with variable duration, while in Mitchell, (1995) it is called HMM with explicit duration. The model was first investigated by Ferguson (1980).

A good graphical representation of the model, which is given by Murphy (2002), is shown in figure 5.4. The states $s_k$ are the regular states that emit symbols, and the states $d_k$ are used to capture the remaining duration of the process in state $s_k$. When the process enters state $s_k$, the value of the duration in $d_k$ is chosen according to
the probability distribution associated with $s_k$. When the time in $d_k$ counts down to zero, the state is free to change. Details of the model was given by Murphy (2002). Note that the $\Omega_k$ are strings of symbols; that is, $\Omega = \{v_1, \ldots, v_m\}$. Because a string of symbols can be emitted from a hidden state, one of the problems that needs to be solved in addition to the standard HMM problems is to calculate the duration distribution of a given state.

![Diagram](image)

**Figure 5.4: Example of HSMM**

### 5.5.3.4 Profile HMMs for Biological Sequence Analysis

The architecture of Hidden Markov Model was introduced by Krogh (1994). Profile Hidden Markov model have wide application in bioinformatics. The use of computers has enabled efficient sequence alignment methods that are now commonly used in bioinformatics and molecular biology. Early research in molecular biology and bioinformatics was motivated by protein sequence analysis. However, due to human genome project and other high throughput projects, there is a dramatic increase in many types of biological data available. This has extended the scope of bioinformatics research to include such topics as protein classification, RNA analysis, structural and functional predictions, and gene prediction and in all these research fields, HMM play a vital role. Deoxyribonucleic acid (DNA), ribonucleic acid (RNA), and proteins are the fundamental building blocks of life. DNA is composed of four bases: adenine (A), cytosine (C), guanine (G), and thymine (T). Similarly, RNA has four bases: adenine (A), cytosine (C), guanine (G), and uracil (U). Thus, one major difference between DNA and RNA is that RNA has uracil instead of thymine. Proteins are more diverse in structure and function than the other kinds of molecules.
and are built from an alphabet of 20 smaller molecules known as amino acids whose single letter representations are A, V, L, I, F, P, M, D, E, K, R, S, T, C, N, Q, H, Y, W, and G. The molecules are usually connected in a linear sequence such that a DNA molecule, RNA molecule, or protein molecule is represented as a sequence of letters. Such sequences are called biological sequences. This simple sequence representation of the molecules enables them to be compared in a simple way. Thus, it is possible to match or align two sequences letter by letter to see how they pair up. One of the reasons for making such a comparison is to find the evolutionary relation between species on a molecular level. Hidden Markov models have become one of the most statistically powerful methods used to model sequence alignment. A special type of left-to-right HMMs called profile HMM (PHMM) is commonly used to model multiple alignments. PHMM is well suited to the popular profile methods for searching databases using multiple sequence alignments instead of single query sequences. It has three types of states: match states that are represented by squares labelled \( m \), insert states that are represented by diamonds labelled \( i \), and delete states that are represented by circles labelled \( d \).

**Figure 5.5: Architecture of profile HMM**

HMM training is the estimation of the emission and transition probabilities. For PHMM, these parameters are obtained from multiple alignment sequences in a protein, DNA, or RNA sequence family. If there is any sequence whose components are known, then it can be used for the training. In general the emission probabilities are the maximum likelihood estimates of the letters in each column. Similarly, the transition probabilities are obtained by counting the number of times each transition would be taken. Multiple alignment means taking a group of three or more sequences and identifying the amino acids that are homologous (or structurally and functionally
similar). Proteins and nucleic acid sequences and their interrelationships can be demonstrated by multiple alignments of the sequences. Suppose we have DNA sequence that are of different lengths, the first step is to introduce gaps to make them of the same length, after that we can calculate emission and transition probabilities (Oliver C, 1947).

The probability of a sequence is used to calculate a score for the sequence. Because multiplication of fractions is computationally expensive and prone to floating point errors such as underflow, the calculation is simplified by taking the logarithm of the score, thereby replacing multiplication by addition. The resulting number is the log score of a sequence. Because a score measures the probability that a sequence belongs to a given family, a high score implies that the sequence of interest is probably a member of the class while a low score implies it is probably not a member.

Bockhorst et al., (2003) proposed a method with a Bayesian network to link many kinds of observations such as spacer sizes, expression profiles, and codon usage. Tjaden et al., (2002) used a hidden Markov model (HMM) to predict transcription boundaries, a hidden Markov model. HMM is a doubly stochastic Markov chain in which a sequence of multinomial state variables \( (v_1, v_2, \ldots, v_T) \) are linked via a state transition matrix, and each element \( y_t \) in a sequence of observations \( (y_1, y_2, \ldots, y_T) \) is drawn independently of the other observations conditional on \( v_t \) (Rabiner, 1989). This is essentially a dynamic variant of a finite mixture model, in which there is one mixture component corresponding to each value of the multinomial state. Note that the HMM involves not a single mixture model, but rather a set of mixture models: one for each value of the current state. That is, the current state \( v_t \) indexes a specific row of the transition matrix, with the probabilities in this row serving as the mixing proportions for the choice of the next state \( v_{t+1} \). Given the next state \( v_{t+1} \), the observation \( y_{t+1} \) is drawn from the mixture component indexed by \( v_{t+1} \).

Beal and Krishnamurthy’s (2012) paper discusses and compared hierarchical Dirichlet process hidden Markov model (HDP-HMM) to the standard HMM, referred to as finite HMM here after. For the finite HMM, we ran experiments with 7 different seed values and averaged over the various scores (explained below), in order to minimize the effects of initialization in EM. We define the (probabilistic measure of) dissimilarity between two genes’ time courses for finite HMM as the \((c, d)^{th}\)
element of a matrix $P$ (size $517 \times 517$), which is the probability that the two time courses of each gene have identical hidden state trajectories. This can be computed straightforwardly after an E step in the Baum-Welch algorithm. Denoting the posterior over the hidden state at time $t$ of the $c^{th}$ gene sequence by eqn. 5.26 below.

$$P (v_t^{(c)} | y_{1:T}^{(c)}, \Theta)$$

(5.26)

Where $\Theta$ are the current parameters of the HMM, then $\log P_{cd}$ is straightforwardly given by

$$\sum_{t=1}^{T} \log \sum_{r=1}^{k} P(v_t^{(c)} = r | y_{1:T}^{(c)}, \Theta) P(v_t^{(d)} = r | y_{1:T}^{(d)}, \Theta)$$

(5.27)

And therefore $P_{cd} (= P_{dc})$ measures the probability of two genes $c$ and $d$ having transversed similar entire hidden trajectories. Log $P_{cd}$ is a measure of divergence between gene $c$ and $d$^2.

In an infinite model, the posterior distribution over hidden state trajectories is represented as a set of samples and the above quantity can be calculated simply from an empirical computation over the samples of trajectories taken over very long MCMC runs. Since the posterior samples always consist of represented hidden states, we do not suffer from the countably infinite state space (Ramussen, 2000; Wild et al., 2002).

Classic applications of HMMs in different areas includes computational biology, modelling of protein families (Krogh et al., 1994; Bateman et al., 2002), gene finding (Krogh, Mian and Haussler, 1994; Burge and Karlin, 1997; Henderson, Salzberg and Fasman, 1997; Lukashin and Borodovsky, 1998; Salzberg et al., 1999), predicting transmembrane helices (Krogh et al. 2001) and tertiary structure prediction. More recent applications include modelling of epigenetic signals, copy number variations and molecular evolution (Garnier and Munson, 1997; Bystroff, Thorsson and Baker, 2000).  

5.6 APPLICATION TO M. TUBERCULOSIS DATA

The Bayesian mixture hieracheral model were used to model gene expression data of mycobacterium tuberculosis and simulation were done using MCMC Gibbs sampling methods.
5.7 RESULTS

5.7.1: Bayesian mixture model with Gibbs sampling

After few simulation, we can visualize that the parametric mean become stable and MC error minimizing. All the iteration results can be visualized in the posterior density plot, autocorrelation plot, trace plot etc. After 4000 iteration, the mean of lambda [1] become 1.743 with MC error of lambda 0.0030 and credible interval (1.471, 2.014). After 6000 iteration the mean of lambda [1] become 1.742 with MC error 0.00238 with credible interval (1.475, 2.008).After 12000 iteration, the mean of parameters λ₁, λ₂, tau and theta were 1.742, 5.39,4.30 and 3.658 respectively .The detailed iteration results of all parameters can be showed in the following table (5.1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mean</th>
<th>SD</th>
<th>MC error</th>
<th>Credible interval</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda [1]</td>
<td>1.743</td>
<td>0.136</td>
<td>0.0030</td>
<td>1.471 , 2.014</td>
<td>4000</td>
</tr>
<tr>
<td>lambda [2]</td>
<td>5.389</td>
<td>0.251</td>
<td>0.0055</td>
<td>4.921 , 5.848</td>
<td>4000</td>
</tr>
<tr>
<td>Tau</td>
<td>4.290</td>
<td>1.480</td>
<td>0.0293</td>
<td>1.978 , 7.778</td>
<td>4000</td>
</tr>
<tr>
<td>Theta</td>
<td>3.646</td>
<td>0.2922</td>
<td>0.0070</td>
<td>3.11 , 4.17</td>
<td>4000</td>
</tr>
<tr>
<td>lambda [1]</td>
<td>1.742</td>
<td>0.1351</td>
<td>0.0023</td>
<td>1.475 , 2.008</td>
<td>6000</td>
</tr>
<tr>
<td>lambda [2]</td>
<td>5.395</td>
<td>0.2446</td>
<td>0.0042</td>
<td>4.92 , 5.864</td>
<td>6000</td>
</tr>
<tr>
<td>Tau</td>
<td>4.302</td>
<td>1.456</td>
<td>0.023</td>
<td>1.996 , 7.678</td>
<td>6000</td>
</tr>
<tr>
<td>Theta</td>
<td>3.654</td>
<td>0.2843</td>
<td>0.0054</td>
<td>3.117 , 4.184</td>
<td>6000</td>
</tr>
<tr>
<td>lambda [1]</td>
<td>1.744</td>
<td>0.135</td>
<td>0.0021</td>
<td>1.478 , 2.013</td>
<td>8000</td>
</tr>
<tr>
<td>lambda [2]</td>
<td>5.397</td>
<td>0.242</td>
<td>0.0035</td>
<td>4.933 , 5.852</td>
<td>8000</td>
</tr>
<tr>
<td>Tau</td>
<td>4.307</td>
<td>1.449</td>
<td>0.0207</td>
<td>2.00 , 7.654</td>
<td>8000</td>
</tr>
<tr>
<td>Theta</td>
<td>3.652</td>
<td>0.2819</td>
<td>0.0047</td>
<td>3.116 , 4.184</td>
<td>8000</td>
</tr>
<tr>
<td>lambda[1]</td>
<td>1.742</td>
<td>0.1346</td>
<td>0.00163</td>
<td>1.473 , 2.007</td>
<td>12000</td>
</tr>
<tr>
<td>lambda[2]</td>
<td>5.399</td>
<td>0.2381</td>
<td>0.00259</td>
<td>4.936 , 5.853</td>
<td>12000</td>
</tr>
<tr>
<td>Tau</td>
<td>4.304</td>
<td>1.451</td>
<td>0.01585</td>
<td>1.964 , 7.654</td>
<td>12000</td>
</tr>
<tr>
<td>Theta</td>
<td>3.658</td>
<td>0.2775</td>
<td>0.00358</td>
<td>3.118 , 4.187</td>
<td>12000</td>
</tr>
</tbody>
</table>
Figure 5.6: Posterior density curve (Initial iteration) of lambda 1 and 2

Figure 5.7: Initial Posterior density curve for tau and theta

Figure 5.8: Posterior density curve of lambda 1 and lambda2

Figure 5.9: Initial Posterior density curve for tau and theta
Figure 5.10: Dynamic Trace Plot of all parameters

Figure 5.11: Quantiles plot of all parameters
Figure 5.12: Trace history plot of all parameters
5.7.2 Gene prediction using HMM

Nucleotide sequence of mycobacterium tuberculosis strain H37Rv with genomic reference No 448814763 with accession number NC_000962.3 was selected for gene prediction using HMM method. It was a complete genome sequence data of H37Rv in the FASTA format. A part of complete genome sequence were selected for gene prediction analysis.
5.7.2.1 H37Rv Sequence

ttgaccgatgaccccggttcaggcttcaccacagtgtggaacgcggtcgtctccgaacttaacggcgacccttaacacgacgcaaggtcgc gcacgctctcagccgccgactcggacatcagaactcggcagagtctgttgccgagaagccgacgatccggggtttgctctgttatccgtgccgagcagctttgtc caaaacgaaatcgagcgccatctgcgggccccgattac

cgacgctctcagccgccgactcggacatcagaactcggcagagtctgttgccgagaagccgacgatccggggtttgctctgttatccgtgccgagcagctttgtc caaaacgaaatcgagcgccatctgcgggccccgattac
5.7.2.2 Gene prediction output

From the given nucleotide sequence, two protein sequence were predicted with probability greater than 0.01. From the given sequence, two exon sequence were predicted with probability 0.91 and 0.99.
5.8. SUMMARY

The Bayesian model is flexible enough to identify the significant genes as well as to perform future prediction. Bayesian inference from the data modeled by mixture distribution can feasibly be performed via Monte Carlo simulation. This Monte Carlo simulation technique based on gibbs sampling procedure can be used to solve Bayesian prediction problem for mixture model. So we can use this simulation technique to analyze complex microarray data sets. Hidden markov model is one of the most powerful techniques, using this method we can predict introns and exons in a sequence. Based on the genomic sequence, HMM model can be used to find the probability of predicting corresponding amino acid codon. In addition to that, HMM can be used to predict protein families and structures also. When we apply the classical statistical techniques to the microarray data with few samples and large variables, it may violate the assumptions. Bayesian statistics with MCMC methods can be solved with this problem and will get better estimates.