BAYESIAN CLASSIFICATION AND REGRESSION WITH HIGH DIMENSIONAL FEATURES

Longhai Li

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Statistics
University of Toronto

Bayesian Classification and Regression with High Dimensional Features

Longhai Li

Submitted for the Degree of Doctor of Philosophy

August 2007

Abstract

This thesis responds to the challenges of using a large number, such as thousands, of features in regression and classification problems.

There are two situations where such high dimensional features arise. One is when high dimensional measurements are available, for example, gene expression data produced by microarray techniques. For computational or other reasons, people may select only a small subset of features when modelling such data, by looking at how relevant the features are to predicting the response, based on some measure such as correlation with the response in the training data. Although it is used very commonly, this procedure will make the response appear more predictable than it actually is. In Chapter 2, we propose a Bayesian method to avoid this selection bias, with application to naive Bayes models and mixture models.

High dimensional features also arise when we consider high-order interactions. The number of parameters will increase exponentially with the order considered. In Chapter 3, we propose a method for compressing a group of parameters into a single one, by exploiting the fact that many predictor variables derived from high-order interactions have the same values for all the training cases. The number of compressed parameters may have converged before considering the highest possible order. We apply this compression method to logistic sequence prediction models and logistic classification models.

We use both simulated data and real data to test our methods in both chapters.

Acknowledgements

I can never overstate my gratitude to my supervisor Professor Radford Neal who guided me throughout the whole PhD training period. Without his inspiration, confidence, and insightful criticism, I would have lost in the course of pursuing this degree. It has been my so far most valuable academic experience to learn from him how to ponder problems, how to investigate them, how to work on them, and how to present the results.

I would like to thank my PhD advisory and exam committee members — Professors Lawrence Brunner, Radu Craiu, Mike Evans, Keith Knight, Jeffrey Rosenthal and Fang Yao. Their comments enhance this final presentation. Most of the aforementioned professors, and in addition Professors Andrey Feuerverger, Nancy Reid, Muni Srivastava, and Lei Sun, have taught me in various graduate courses. Much knowledge from them has become part of this thesis silently.

I wish to specially thank my external appraiser, Professor Andrew Gelman. Many of his comments have greatly improved the previous draft of this thesis.

I am grateful to the support provided by the statistics department staff — Laura Kerr, Andrea Carter, Dermot Wheland and Ram Mohabir. They have made the student life in this department so smooth and enjoyable.

I am indebted to my many student colleagues, who accompanied and shared knowledge with me. Special thanks go to Shelley Cao, Meng Du, Ana-Maria Staicu, Shuying Sun, Tao Wang, Jianguo Zhang, Sophia Lee, Babak Shahbab, Jennifer Listgarten, and many others.

I would like to thank my wife, Yehua Zhang. It would have been impossible to finish this thesis without her support and love.

I wish to thank my sister, Meiwen Li. She provided me with much support at the most difficult time to me.

The last and most important thanks go to my parents, Baoqun Jie and Yansheng Li. They bore me, raised me and supported me. To them I dedicate this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Classification and Regression

Methods for predicting a response variable y given a set of features $\boldsymbol{x} = (x_1, \dots, x_p)$ are needed in numerous scientific and industrial fields. A doctor wants to diagnose whether a patient has a certain kind of disease from some laboratory measurements on this patient; a post office wants to use a machine to recognize the digits and characters on envelopes; a librarian wants to classify documents using a pre-specified list of topics; a businessman wants to know how likely a person is to be interested in a new product according to this person's expenditure history; people want to know the temperature tomorrow given the meteorologic data in the past; etc. Many such problems can be summarized as finding a *predictive function* C linking the features $\boldsymbol{x}$ to a prediction for y:

$$\hat{y} = C(\boldsymbol{x}) \tag{1.1}$$

The choice of function C depends also on the choice of loss function one wishes to use in making a decision. In scientific discussion, we focus on finding a probabilistic predictive distribution:

$$P(y \mid \boldsymbol{x}) \tag{1.2}$$

Here, $P(y \mid \boldsymbol{x})$ could be either a probability density function for continuous y (a regression model), or a probability mass function for discrete or categorical y (a classification model). Given a loss function, one can derive the predictive function C from the predictive distribution $P(y \mid \boldsymbol{x})$ by minimizing the average loss in the future. For example, when y is continuous, if we use a squared loss function $L(\hat{y}, y) = (\hat{y} - y)^2$, the best guess of y is the mean of $P(y \mid \boldsymbol{x})$; if we use an absolute loss function $L(\hat{y}, y) = |\hat{y} - y|$, the best guess is the median of $P(y \mid \boldsymbol{x})$; and when y is discrete, if we use $0 - 1$ loss function $L(\hat{y}, y) = I(\hat{y} \neq y)$, the best guess is the mode of $P(y \mid \boldsymbol{x})$.

One approach to finding $P(y \mid \boldsymbol{x})$ is to learn from empirical data — data on a number of subjects that have known values of the response and values of features, denoted by $\{(y^{(1)}, \boldsymbol{x}^{(1)}), \ldots, (y^{(n)}, \boldsymbol{x}^{(n)})\}$, or collectively by $(y^{\text{train}}, \boldsymbol{x}^{\text{train}})$. This is often called "training" data, and the subjects are called "training" cases, as we are going to use these data to "train" an initially "unskilled" predictive model, as discussed later. In contrast, a subject whose response and features are denoted by $(y^*, \boldsymbol{x}^*)$, for which we need to predict the response, is called a "test" case, because we can use the prediction result to test how good a predictive model is if we are later given the true y^*.

There are many methods to learn from the training data (Hastie, Tibshirani and Friedman 2001 and Bishop 2006). One may estimate $P(y^* \mid \boldsymbol{x}^*)$ using the empirical distribution of the responses in the neighbourhood of $\boldsymbol{x}^*$ in some metric, as in the k-nearest-neighbourhood method. Such methods are called nonparametric methods. In this thesis, we consider parametric methods, in which we use a closed-form function with unknown parameters to model the data. Once the parameters are inferred from the training data we can discard the training data because we only need the parameters of the "trained" model for making predictions on test cases.

One class of parametric methods, called conditional modelling methods, start by defining $P(y \mid \boldsymbol{x})$ as a function involving some unknown parameters, denoted by $\boldsymbol{\theta}$. These parameters will be inferred from training data. For continuous y, the simplest and most commonly used form for $P(y \mid \boldsymbol{x})$ is a Gaussian model:

$$P(y \mid \boldsymbol{x}, \boldsymbol{\beta}, \sigma) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y - f(\boldsymbol{x}, \boldsymbol{\beta}))^2}{2\,\sigma^2}\right) \tag{1.3}$$

For a discrete y that takes K possible values $0, \ldots, K-1$, one may use a logistic form for $P(y \mid \boldsymbol{x})$:

$$P(y = k \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{\exp(f_k(\boldsymbol{x}, \boldsymbol{\beta}_k))}{\sum_{j=0}^{K-1} \exp(f_j(\boldsymbol{x}, \boldsymbol{\beta}_j))} \tag{1.4}$$

The function $f(\boldsymbol{x}, \boldsymbol{\beta})$ or functions $f_j(\boldsymbol{x}, \boldsymbol{\beta}_j)$ link $\boldsymbol{x}$ to y. They are often linear functions of $\boldsymbol{x}$, but may be also nonlinear functions of $\boldsymbol{x}$ defined, for example, by multilayer perceptron networks. Our work in Chapter 3 uses linear logistic models.

Another class of methods model the joint distribution of y and $\boldsymbol{x}$ by some formula with unknown parameters $\boldsymbol{\theta}$, written as $P(y, \boldsymbol{x} \mid \boldsymbol{\theta})$. The conditional probability $P(y \mid \boldsymbol{x}, \boldsymbol{\theta})$ can be found by:

$$P(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{P(y, \boldsymbol{x} \mid \boldsymbol{\theta})}{P(\boldsymbol{x} \mid \boldsymbol{\theta})} \tag{1.5}$$

Examples of such $P(y, \boldsymbol{x}, \boldsymbol{\theta})$ include naive Bayes models, mixture models, Bayesian networks, and Markov random fields, etc., all of which use conditional independency in specifying $P(y, \boldsymbol{x} \mid \boldsymbol{\theta})$. For example naive Bayes models assume all features $\boldsymbol{x}$ are independent given y. Our work in Chapter 2 uses naive Bayes models and mixture models.

There are two generally applicable approaches for inferring $\boldsymbol{\theta}$ from the training data. One is to estimate $\boldsymbol{\theta}$ using a single value, $\hat{\boldsymbol{\theta}}$, that maximizes the likelihood function or a penalized likelihood function, i.e., the value that best fits the training data subject to some constraint. This single estimate will be plugged in to $P(y \mid \boldsymbol{x}, \boldsymbol{\theta})$ or $P(y, \boldsymbol{x} \mid \boldsymbol{\theta})$ to obtain the predictive distribution $P(y^* \mid \boldsymbol{x}^*, \hat{\boldsymbol{\theta}})$ for a test case.

Alternatively, we can use a Bayesian approach, in which we first define a prior distribution, $P(\boldsymbol{\theta})$, for $\boldsymbol{\theta}$, which reflects our "rough" knowledge about $\boldsymbol{\theta}$ before seeing the data, and then update our knowledge about $\boldsymbol{\theta}$ after we see the data, still expressed with a probability distribution, using Bayes formula:

$$P(\boldsymbol{\theta} \mid y^{\text{train}}, \boldsymbol{x}^{\text{train}}) = \frac{P(y^{\text{train}}, \boldsymbol{x}^{\text{train}} \mid \boldsymbol{\theta})\, P(\boldsymbol{\theta})}{P(y^{\text{train}}, \boldsymbol{x}^{\text{train}})} \tag{1.6}$$

$P(\boldsymbol{\theta} \mid y^{\text{train}}, \boldsymbol{x}^{\text{train}})$ is called the posterior distribution of $\boldsymbol{\theta}$. The joint distribution of a test case $(y^*, \boldsymbol{x}^*)$ given the training data $(y^{\text{train}}, \boldsymbol{x}^{\text{train}})$ is found by integrating over $\boldsymbol{\theta}$ with respect to the posterior distribution:

$$P(y^*, \boldsymbol{x}^* \mid y^{\text{train}}, \boldsymbol{x}^{\text{train}}) = \int P(y^*, \boldsymbol{x}^* \mid y^{\text{train}}, \boldsymbol{x}^{\text{train}}, \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid y^{\text{train}}, \boldsymbol{x}^{\text{train}})\, d\boldsymbol{\theta} \tag{1.7}$$

The predictive distribution can then be found as $P(y^*, \boldsymbol{x}^* \mid y^{\text{train}}, \boldsymbol{x}^{\text{train}}) / P(\boldsymbol{x}^* \mid y^{\text{train}}, \boldsymbol{x}^{\text{train}})$, which will be used to make predictions on test cases in conjunction with our loss function.

1.2 Challenges of Using High Dimensional Features

In many regression and classification problems, a large number of features are available for possible use. DNA microarray techniques can simultaneously measure the expression levels of thousands of genes (Alon et.al. 1999, Khan et.al. 2001); the HIRIS instrument for the Earth Observing System generates image data in 192 spectral bands simultaneously (Lee and Landgrebe et.al. 1993); one may consider numerous high-order interactions of discrete features; etc.

There are several non-statistical difficulties in using high-dimensional features, depending on the purpose the data is used for. The primary one is computation time. Models for high dimensional data will require high dimensional parameters. Consequently, the time for training the model and making predictions on test cases may be intolerable. For example, a speech recognition program or data compression program must be able to give out the prediction very quickly to be practically useful. Also, in some cases, measuring high dimensional features takes substantially more time or money.

Serious statistical problems also arise with high dimensional features. When the number of features is larger than the number of training cases, the usual estimate of the covariance matrix of features is singular, and therefore can not be used to compute the density function. Regularization methods that shrink the estimation to a diagonal matrix have been proposed in the literature (Friedman 1998, Tadjudin and Landgrebe 1998, 1999). Such methods usually need to adjust some parameters that control the degree of shrinkage to a diagonal matrix, which may be difficult to determine. Another aspect of this problem is that even a simple model, such as a linear model, will overfit data with high dimensional features. Linear logistic models with the coefficients estimated by the maximum likelihood method will have some coefficients equal to ∞; the solution is also not unique. This is because the training cases can be divided by some hyperplanes in the space of features into groups such that all the cases with the same response are in a group; indeed, there are infinitely many such hyperplanes. The resulting classification rule works perfectly on the training data but may perform poorly on the test data. Overfitting problems usually arises because one uses more complex models than the data can support. For example, when one uses a polynomial function of degree n to fit the relationship between x and y in n data points $(y^{(i)}, x^{(i)})$, there are infinitely many such polynomial functions that go exactly through each of these n points.

A sophisticated Bayesian method can overcome the overfitting problem by using a prior that favours simpler models. But unless one can analytically integrate with respect to the posterior distribution, implementating such a Bayesian method by Markov chain sampling is difficult. With more parameters, a Markov chain sampler will take longer for each iteration and require more memory. It may need more iterations to converge, or get trapped more easily in local modes. Also, with high dimensional features, it is harder to come up with a prior that reflects all of our knowledge of the problem.

1.3 Two Problems Addressed in this Thesis

For the above reasons, people often use some methods to reduce the dimension of features before applying regression or classification methods. However, a simple implementation of such a "preprocessing" procedure may be invalid. For example, we may first select a small set of features that are most correlated with the response in the training data, then use these features to construct a predictive distribution. This procedure will make the response variable appear more predictable than it actually is. This overconfidence may be more pronounced when there are more features available, as more actually useless features will by chance pass the selection process, especially when very few useful features exist. In Chapter 2, we propose a method to avoid this problem with feature selection in a Bayesian framework. In constructing the posterior distribution of parameters, we condition not only on the retained features, but also on the information that a number of features are discarded because of their weak correlations with the response. The key point in our solution is that we need only calculate the probability that one feature is discarded, then raise it to the power of the number of discarded features. We therefore can save much computation time by selecting only a very small number of features for use, and at the same time make well-calibrated predictions for test cases. We apply this method to naive Bayes models and mixture models for binary data.

A huge number of parameters will arise when we consider very high order interactions of discrete features. But many interaction patterns are expressed by the same training cases. In Chapter 3, we use this fact to reduce the number of parameters by effectively compressing a group of parameters into a single one. After compressing the parameters, there are many fewer parameters involved in the Markov chain sampling. The original parameters can later be recovered efficiently by sampling from a splitting distribution. We can therefore consider very high order interactions in a reasonable amount of time. We apply this compression method to logistic sequence prediction models and logistic classification models.

1.4 Comments on the Bayesian Approach

The Bayesian approach is sometimes criticized for its use of prior distributions. Many people view the choice of prior as arbitrary because it is subjective. The prior is the distribution of $\boldsymbol{\theta}$ that generates, through a defined sampling distribution, the class of data sets that will enter our analysis. Thus, there is only one prior that accurately defines the characteristics of the class of data sets, which may be described in another way, such as in words. Different individuals may define different classes of data sets. The choice of prior is therefore subjective, but not arbitrary, since we may indeed decide that a prior distribution is wrong if the data sets it generates contradict our beliefs. Typically we choose a diffuse prior to include a wide class of data sets, but de-emphasize some data sets we believe less likely to appear in our analysis, for example a data set generated by a linear logistic model with coefficient equal to 10000 for a binary feature. This distribution is therefore also phrased as expressing our prior belief, or our "rough" knowledge about which $\boldsymbol{\theta}$ may have generated our data set.

There is usually useful prior information available for a problem before seeing any data set, such as relationships between the parameters (or data). For example, a set of body features of a human should be closer to those of a monkey than to other animals. A sophisticated prior distribution can be used to capture such relationships. For example, we can assign the two groups of parameters, which are used to define the distribution of body features of a human and a monkey, a joint prior distribution in which they are positively correlated (Gelman, Bois and Jiang 1996). We usually construct such joint distributions by introducing some extra parameters that are shared by a group of parameters, which may also have meaningful interpretations. One way is to define the priors of the parameters of likelihood function in terms of some unknown hyperparameters, which is again given a higher level distribution. For example, in Automatic Relevance Determination (ARD) priors for neural network regression (Neal 1996), all the coefficients related to a feature are controlled by a common standard deviation. Such priors enable the models to decide whether a feature is useful automatically, through adjusting the posterior distribution of the common standard deviation. Similarly, in the priors for the models in Chapter 2 we use a parameter α to control the overall degree of relationship between the features and response. Our method for avoiding the bias from feature selection has the effect of adjusting the posterior distribution of α to be closer to the right one (as would be obtained using the complete data), by conditioning on *all information known to us*, both the retained features and information about the feature selection process. Another way of introducing dependency is to express a group of parameters as the functions of a group of "brick" parameters. For example, in Chapter 3, the regression coefficients for the highest order interaction patterns are expressed as sums of parameters representing the effects of lower order interaction patterns. Such priors enable the models to choose the orders automatically.

Once we have assigned an appropriate prior distribution for a problem, all forms of inference for unknown quantities, including the unknown parameters, can be carried out very straightforwardly in theory using only the rules of probability, since the result of inference is also expressed by a probability distribution. These predictions are found by averaging over all sets of values of $\boldsymbol{\theta}$ that are plausible in light of the training data. Compared with non-Bayesian methods, which use only a single set of parameters, the Bayesian approach has the following advantages from a practical viewpoint.

First, the prediction is automatically accompanied by information on its uncertainty in making predictions, since the prediction is expressed by a probability distribution.

Second, Bayesian prediction may be better than prediction based on only a single set of parameters. If the set of parameters that best explains the training data, such the MLE, is not the true set of parameters that generates the training data, we still have the chance to make good predictions, since the true set of parameters should be plausible given the training data and therefore will be considered as well in Bayesian prediction.

Third, sophisticated Bayesian models, as described earlier, will self-adjust the complexity of a model in light of the data. We can define a model through a diffuse prior that can cover a wide class of data sets, from those with a low level of complexity to those with a high level of complexity. If the training data does not favour the high complexity, the posterior distribution will choose to use the simple model. In theory we do not need to change the complexity of a model according to the properties of the data, such as the number of observations. The overfitting problem in applying a complex model to a data set of small size is therefore overcome in Bayesian framework. Although more complex models may make the computation harder, Bayesian methods are, at least, much less sensitive to the choice of model complexity level than non-Bayesian methods.

Bayesian inference, however, is difficult to carry out, primarily for computational reasons. The posterior distribution is often on a high dimensional space, often takes a very complicated form, and may have a lot of isolated modes. Markov chain Monte Carlo (MCMC) methods (Neal 1993, Liu 2001 and the references therein) are so far the only feasible methods to draw samples from a posterior distribution (Tierney 1994). In the next section, we will briefly introduce these methods. However, for naive Bayes models in Chapter 2 we do not use MCMC, due to the simplicity of naive Bayes models.

1.5 Markov Chain Monte Carlo Methods

We can simulate a Markov chain governed by a transition distribution $T(\boldsymbol{\theta}' \mid \boldsymbol{\theta})$ to draw samples from a distribution $\pi(\boldsymbol{\theta})$, where $\boldsymbol{\theta} \in S$, if T leaves π invariant:

$$\int_S \pi(\boldsymbol{\theta}) T(\boldsymbol{\theta}' \mid \boldsymbol{\theta})\, d\boldsymbol{\theta} = \pi(\boldsymbol{\theta}') \tag{1.8}$$

and satisfies the following conditions: the Markov chain should be aperiodic, i.e., it does not explore the space in a cyclic way, and the Markov chain should be irreducible, i.e., the Markov chain can explore the whole space starting from any point. Given these conditions, it can be shown there is only one distribution π satisfying the invariance condition (1.8) for a Markov chain transition T if there is one. (The condition of aperiodicity is not actually required for Monte Carlo estimation, but it is convenient in practice if a Markov chain is aperiodic, since we have more freedom in choosing the iterations for making Monte Carlo estimation. And it is obviously required to ensure that the result in (1.9) is true.)

Let us denote a Markov chain by $\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \ldots$. (Roberts and Rosenthal 2004) shows that if a Markov chain transition T satisfies all the above conditions with respect to π, then starting from any point $\boldsymbol{\theta}_0$ for $\boldsymbol{\theta}^{(0)}$, the distribution of $\boldsymbol{\theta}^{(n)}$ will converge to π:

$$\lim_{n->\infty} P(\boldsymbol{\theta}^{(n)} = \boldsymbol{\theta} \mid \boldsymbol{\theta}^{(0)} = \boldsymbol{\theta}_0) = \pi(\boldsymbol{\theta}), \qquad \text{for any } \ \boldsymbol{\theta}, \boldsymbol{\theta}_0 \in S \tag{1.9}$$

In words, after we run a Markov chain sufficiently long, the distribution of $\boldsymbol{\theta}^{(n)}$ (regardless the starting point) will be close to the target distribution $\pi(\boldsymbol{\theta})$ in some metric (Rosenthal 1995 and the references therein). We can therefore use the states afterward as samples from $\pi(\boldsymbol{\theta})$ (though correlated) for making Monte Carlo estimations. It is tremendously difficult to determine in advance how long we should run for an arbitrary Markov chain, though we can do this for some types of Markov chains (Rosenthal 1995). In practice we check the convergence by running multiple chains starting from different points and see whether they have mixed at a certain time (see for example Cowles and Carlin 1996, and the references therein).

It is usually not difficult to construct a Markov chain transition T that satisfies the invariance condition for a desired distribution π and the other two conditions as well, based on the following facts. First, one can show that a Markov chain transition T leaves π invariant if it is reversible with respect to π:

$$\pi(\boldsymbol{\theta})\, T(\boldsymbol{\theta}' \mid \boldsymbol{\theta}) = \pi(\boldsymbol{\theta}')\, T(\boldsymbol{\theta} \mid \boldsymbol{\theta}'), \qquad \text{for any} \ \ \boldsymbol{\theta}', \boldsymbol{\theta} \in S \tag{1.10}$$

It therefore suffices to devise a Markov chain that is reversible with respect to π. Second, applying a series of Markov chain transition T_i that have been shown to leave π invariant will also leave π invariant. Also, applying a series of appropriate Markov chain transition T_i that explores only a subset of S can explore the whole space, S.

Gibbs sampling method (Geman and Geman 1984, and Gelfand and Smith 1990) and the Metropolis-Hastings method (Metropolis et. al. 1953, and Hastings 1970) are two basic methods to devise a Markov chain transition that leaves π invariant. We usually use a combination of them to devise a Markov chain transition satisfying the above conditions for a complicated target distribution π.

Let us write $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$. Gibbs sampling defines the transition from $\boldsymbol{\theta}^{(t-1)}$ to $\boldsymbol{\theta}^{(t)}$ as follows:

$$
\begin{aligned}
&\text{Draw } \theta_1^{(t)} \text{ from } \pi(\theta_1 \mid \theta_2^{(t-1)}, \ldots, \theta_p^{(t-1)}) \\
&\text{Draw } \theta_2^{(t)} \text{ from } \pi(\theta_2 \mid \theta_1^{(t)}, \theta_3^{(t-1)}, \ldots, \theta_p^{(t-1)}) \\
&\vdots \\
&\text{Draw } \theta_i^{(t)} \text{ from } \pi(\theta_i \mid \theta_1^{(t)}, \ldots, \theta_i^{(t)}, \theta_{i+1}^{(t-1)}, \ldots, \theta_p^{(t-1)}) \\
&\vdots \\
&\text{Draw } \theta_p^{(t)} \text{ from } \pi(\theta_p \mid \theta_1^{(t)}, \ldots, \theta_{p-1}^{(t)})
\end{aligned}
$$

The order of updating θ_i can be any permutation of $1, \ldots, p$. One can show each updating of θ_i is reversible with respect to $\pi(\boldsymbol{\theta})$, and a complete updating of all θ_i therefore leaves $\pi(\boldsymbol{\theta})$ invariant. Sampling from the conditional distribution for θ_i can also be replaced with any transition that leaves the conditional distribution invariant, for example, a Metropolis-Hastings transition as described next.

The Metropolis-Hastings method first samples from a proposal distribution $\hat{T}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(t-1)})$ to propose a candidate $\boldsymbol{\theta}^*$, then draws a random number U from the uniform distribution over $(0, 1)$. If

$$
U < \min\left(1,\ \frac{\pi(\boldsymbol{\theta}^*)\,\hat{T}(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^{(t-1)})\,\hat{T}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(t-1)})}\right), \tag{1.11}
$$

we let $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^*$, otherwise we let $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$. One can show that such a transition is reversible with respect to π, and hence leave π invariant.

1.6 Outline of the Remainder of the Thesis

We will discuss in detail our method for avoiding bias from feature selection in Chapter 2, with application to naive Bayes models and mixture models. In Chapter 3 we discuss how to compress the parameters in Bayesian regression and classification models with high-order interactions, with application to logistic sequence prediction models and to logistic classification models. We conclude separately at the end of each chapter.

Chapter 2

Avoiding Bias from Feature Selection

Abstract. For many classification and regression problems, a large number of features are available for possible use — this is typical of DNA microarray data on gene expression, for example. Often, for computational or other reasons, only a small subset of these features are selected for use in a model, based on some simple measure such as correlation with the response variable. This procedure may introduce an optimistic bias, however, in which the response variable appears to be more predictable than it actually is, because the high correlation of the selected features with the response may be partly or wholly due to chance. We show how this bias can be avoided when using a Bayesian model for the joint distribution of features and response. The crucial insight is that even if we forget the exact values of the unselected features, we should retain, and condition on, the knowledge that their correlation with the response was too small for them to be selected. In this paper we describe how this idea can be implemented for "naive Bayes" and mixture models of binary data. Experiments with simulated data confirm that this method avoids bias due to feature selection. We also apply the naive Bayes model to subsets of data relating gene expression to colon cancer, and find that correcting for bias from feature selection does improve predictive performance.

[1] Part of this Chapter appeared as a technical report coauthored with Jianguo Zhang and Radford Neal.

2.1 Introduction

Regression and classification problems that have a large number of available “features” (also known as “inputs”, “covariates”, or “predictor variables”) are becoming increasingly common. Such problems arise in many application areas. Data on the expression levels of tens of thousands of genes can now be obtained using DNA microarrays, and used for tasks such as classifying tumors. Document analysis may be based on counts of how often each word in a large dictionary occurs in each document. Commercial databases may contain hundreds of features describing each customer.

Using all the features available is often infeasible. Using too many features can result in “overfitting” when simple statistical methods such as maximum likelihood are used, with the consequence that poor predictions are made for the response variable (e.g., the class) in new items. More sophisticated Bayesian methods can avoid such statistical problems, but using a large number of features may still be undesirable. We will focus primarily on situations where the computational cost of looking at all features is too burdensome. Another issue in some applications is that using a model that looks at all features will require measuring all these features when making predictions for future items, which may sometimes be costly. In some situations, models using few features may be preferred because they are easier to interpret.

For the above reasons, modellers often use only a subset of features, chosen by some simple indicator of how useful they might be in predicting the response variable — see, for example, the papers in (Guyon, *et al.* 2006). For both regression problems with a real-valued response variable and classification problems with a binary (0/1) class variable, one suitable measure of how useful a feature may be is the sample correlation of the feature with the response. If the absolute value of this sample correlation is small, we might decide to omit the feature from our model. This criterion is not perfect, of course — it may result in a relevant feature being ignored if its relationship with the response is non-linear, and it may result in many redundant features being retained even when they all contain essentially the same information. Sample correlation is easily computed, however, and hence is an attractive criterion for screening a large number of features.

Unfortunately, a model that uses only a subset of features, selected based on their high correlation with the response, will be optimistically biased — i.e., predictions made using the model will (on average) be more confident than is actually warranted. For example, we might find that the model predicts that certain items belong to class 1 with probability 90%, when in fact only 70% of these items are in class 1. In a situation where the class is actually completely unpredictable from the features, a model using a subset of features that purely by chance had high sample correlation with the class may produce highly confident predictions that have less actual chance of being correct than just guessing the most common class. The feature selection bias has also been noticed in the literature by a few researchers, see for example, the papers (Ambroise and McLachlan 2002), (Lecocke and Hess 2004), (Singhi and Liu 2006), and (Raudys, Baumgartner and Somorjai 2005). They pointed out that if the feature selection is performed externally to the cross-validation assessment (ie, cross-validation is applied to a subset of features selected in advance based on all observations), the classification error rate will be highly underestimated (could be 0%). It is therefore suggested that feature selection should be performed internally to the cross-validation procedure, ie, re-selecting features whenever the training set and test set are changed. This modified cross-validation procedure avoids underestimating the error rate and assesses properly the predictive method plus the feature selection method. However, it does not provide a scheme for constructing a better predictive method that can give out well-calibrated predictive probabilities for test cases. We propose a Bayesian solution to this problem.

This optimistic bias comes from ignoring a basic principle of Bayesian inference — that we should base our conclusions on probabilities that are conditional on *all* the available information. If we have an appropriate model, this principle would lead us to use all the features.

This would produce the best possible predictive performance. However, we assume here that computational or other pragmatic issues make using all features unattractive. When we therefore choose to "forget" some features, we can nevertheless still retain the information about how we selected the subset of features that we use in the model. Properly conditioning on this information when forming the posterior distribution eliminates the bias from feature selection, producing predictions that are as good as possible given the information in the selected features, without the overconfidence that comes from ignoring the feature selection process.

We can use the information from feature selection procedure only when we model the features and the response jointly. We show in this Chapter this information can be easily incorporated into our inference in a Bayesian framework. We particularly apply this method to naive Bayes models and mixture models.

2.2 Our Method for Avoiding Selection Bias

Suppose we wish to predict a response variable, y, based on the information in the numerical features $x_1, \ldots, x_p$, which we sometimes write as a vector, $\boldsymbol{x}$. Our method is applicable both when y is a binary (0/1) class indicator, as is the case for the naive Bayes models discussed later, and when y is real-valued. We assume that we have complete data on n "training" cases, for which the responses are $y^{(1)}, \ldots, y^{(n)}$ (collectively written as y^{train}) and the feature vectors are $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ (collectively written as $\boldsymbol{x}^{\text{train}}$). (Note that when y, $\boldsymbol{x}$, or x_t are used without a superscript, they will refer to some unspecified case.) We wish to predict the response for one or more "test" cases, for which we know only the feature vector. Our predictions will take the form of a distribution for y, rather than just a single-valued guess.

We are interested in problems where the number of features, p, is quite big — perhaps as large as ten or a hundred thousand — and accordingly (for pragmatic reasons) we intend to select a subset of features based on the absolute value of each feature's sample correlation with the response. The sample correlation of the response with feature t is defined as follows (or as zero if the denominator below is zero):

$$\mathrm{COR}(y^{\mathrm{train}},\, x_t^{\mathrm{train}}) \;=\; \frac{\displaystyle\sum_{i=1}^{n} \left(y^{(i)} - \bar{y}\right)\left(x_t^{(i)} - \bar{x}_t\right)}{\sqrt{\sum_{i=1}^{n}\left(y^{(i)} - \bar{y}\right)^2}\;\sqrt{\sum_{i=1}^{n}\left(x_t^{(i)} - \bar{x}_t\right)^2}} \tag{2.1}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)}$ and $\bar{x}_t = \frac{1}{n}\sum_{i=1}^{n} x_t^{(i)}$. The numerator can be simplified to $\sum_{i=1}^{n}\big(y^{(i)} - \bar{y}\big)x_t^{(i)}$.

Although our interest is only in predicting the response, we assume that we have a model for the joint distribution of the response together with all the features. From such a joint distribution, with probability or density function $P(y, x_1, \ldots, x_p)$, we can obtain the conditional distribution for y given any subset of features, for instance $P(y \,|\, x_1, \ldots, x_k)$, with $k < p$. This is the distribution we need in order to make predictions based on this subset. Note that selecting a subset of features makes sense only when the omitted features can be regarded as random, with some well-defined distribution given the features that are retained, since such a distribution is essential for these predictions to be meaningful. This can be seen from the following expression:

$$\begin{aligned} P(y \,|\, x_1, \ldots, x_k) \;=\; \int \cdots \int & P(y \,|\, x_1, \ldots, x_k, x_{k+1}, \ldots, x_p) \cdot \\ & P(x_{k+1}, \ldots, x_p \,|\, x_1, \ldots, x_k)\; dx_{k+1} \cdots dx_p \end{aligned} \tag{2.2}$$

If $P(x_{k+1}, \ldots, x_p \,|\, x_1, \ldots, x_k)$ does not exist in any meaningful sense — as would be the case, for example, if the data were collected by an experimenter who just decided arbitrarily what to set $x_{k+1}, \ldots, x_p$ to — then $P(y \,|\, x_1, \ldots, x_k)$ will also have no meaning.

Consequently, features that cannot usefully be regarded as random should always be retained. Our general method can accommodate such features, provided we use a model for the joint distribution of the response together with the random features, conditional on given values for the non-random features. However, for simplicity, we will ignore the possible presence of non-random features in this paper.

We will assume that a subset of features is selected by fixing a threshold, γ, for the absolute value of the correlation of a selected feature with the response. We then omit feature t from the feature subset if $|\mathrm{COR}(y^{\mathrm{train}}, x_t^{\mathrm{train}})| \leq \gamma$, retaining those features with a greater degree of correlation. Another possible procedure is to fix the number of features, k, that we wish to retain, and then choose the k features whose correlation with the response is greatest in absolute value, breaking any tie at random. If s is the retained feature with the weakest correlation with the response, we can set γ to $|\mathrm{COR}(y^{\mathrm{train}}, x_s^{\mathrm{train}})|$, and we will again know that if t is any omitted feature, $|\mathrm{COR}(y^{\mathrm{train}}, x_t^{\mathrm{train}})| \leq \gamma$. If either the response or the features have continuous distributions, exact equality of sample correlations will have probability zero, and consequently this situation can be treated as equivalent to one in which we fixed γ rather than k. If sample correlations for different features can be exactly equal, we should theoretically make use of the information that any possible tie was broken the way that it was, but ignoring this subtlety is unlikely to have any practical effect, since ties are still likely to be rare.

Regardless of the exact procedure used to select features, we will denote the number of features retained by k, we will renumber the features so that the subset of retained features is $x_1, \ldots, x_k$, and we will assume we know that $|\mathrm{COR}(y^{\mathrm{train}}, x_t^{\mathrm{train}})| \leq \gamma$ for $t = k+1, \ldots, p$.

We can now state the basic principle behind our bias-avoidance method: When forming the posterior distribution for parameters of the model using a subset of features, we should condition not only on the values in the training set of the response and of the k features we retained, but also on the fact that the other $p-k$ features have sample correlation with the response that is less than γ in absolute value. That is, the posterior distribution should be conditional on the following information:

$$y^{\mathrm{train}}, \quad \boldsymbol{x}_{1:k}^{\mathrm{train}}, \quad |\mathrm{COR}(y^{\mathrm{train}}, x_t^{\mathrm{train}})| \leq \gamma \text{ for } t = k+1, \ldots, p \tag{2.3}$$

where $\boldsymbol{x}_{1:k}^{\text{train}} = (x_1^{\text{train}}, \ldots, x_k^{\text{train}})$.

We claim that this procedure of conditioning on the fact that selection occurred will eliminate the bias from feature selection. Here, "bias" does not refer to estimates for model parameters, but rather to our estimate of how well we can predict responses in test cases. Bias in this respect is referred to as a lack of "calibration" — that is, the predictive probabilities do not represent the actual chances of events (Dawid 1982). If the model describes the actual data generation mechanism, and the actual values of the model parameters are indeed randomly chosen according to our prior, Bayesian inference always produces well-calibrated results, on average with respect to the data and model parameters generated from the Bayesian model. The proof that the Bayesian inference is well-calibrated is given in the Appendix 1 to this Chapter.

In justifying our claim that this procedure avoids selection bias (ie, is well-calibrated), we will assume that our model for the joint distribution of the response and all features, and the prior we chose for it, are appropriate for the problem, and that we would therefore not see bias if we predicted the response using all the features. Now, imagine that rather than selecting a subset of features ourselves, after seeing all the data, we instead set up an automatic mechanism to do so, providing it with the value of γ to use as a threshold. This mechanism, which has access to all the data, will compute the sample correlations of all the features with the response, select the subset of features by comparing these sample correlations with γ, and then erase the values of the omitted features, delivering to us only the identities of the selected features and their values in the training cases. If we now condition on all the information that *we* know, but not on the information that was available to the selection mechanism but not to us, we will obtain unbiased inferences. The information we know is just that of (2.3) above.

The class of models we will consider in detail may include a vector of latent variables, $\boldsymbol{z}$, for each case. Model parameters $\theta_1, \ldots, \theta_p$ (collectively denoted $\boldsymbol{\theta}$) are associated with the

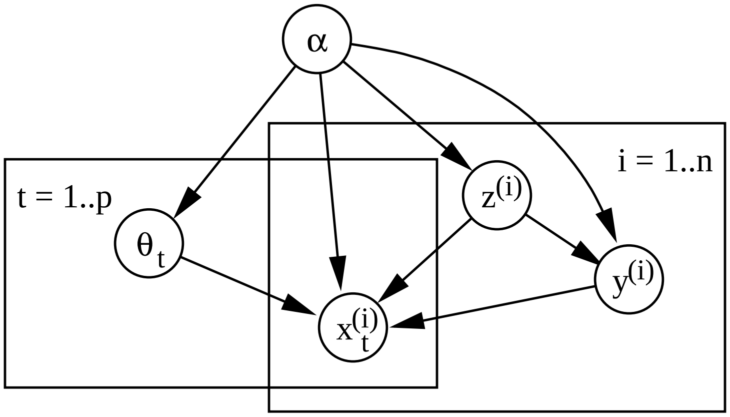


Figure 2.1: A directed graphical model for the general class of models we are considering. Circles represent variables, parameters, or hyperparameters. Arrows represent possible direct dependencies (not all of which are necessarily present in all models in this class). The rectangles enclose objects that are repeated; an object in both rectangles is repeated in both dimensions. The case index, i, is shown as ranging over the n training cases, but test cases (not shown) belong in this rectangle as well. This diagram portrays a model where cases are independent given α and $\boldsymbol{\theta}$, though this is not essential.

p features; other parameters or hyperparameters, α, not associated with particular features, may also be present. Conditional on $\boldsymbol{\theta}$ and α, the different cases may be independent, though this is not essential for our method. Our method does rely on the values of different features (in all cases) being independent, conditional on $\boldsymbol{\theta}$, α, y^{train}, and $\boldsymbol{z}^{\text{train}}$. Also, in the prior distribution for the parameters, $\theta_1, \ldots, \theta_p$ are assumed to be conditionally independent given α. These conditional independence assumptions are depicted graphically in Figure 2.1.

If we retain all features, our prediction for the response, y^*, in a test case for which we know the features, $\boldsymbol{x}^* = (x_1^*, \ldots, x_p^*)$, can be found from the joint predictive distribution for y^* and $\boldsymbol{x}^*$ given the data for all training cases, written as y^{train} and $\boldsymbol{x}^{\text{train}}$:

$$P(y^* \,|\, \boldsymbol{x}^*,\, y^{\text{train}},\, \boldsymbol{x}^{\text{train}}) \;=\; \frac{P(y^*,\, \boldsymbol{x}^* \,|\, y^{\text{train}},\, \boldsymbol{x}^{\text{train}})}{P(\boldsymbol{x}^* \,|\, y^{\text{train}},\, \boldsymbol{x}^{\text{train}})} \tag{2.4}$$

$$= \frac{\int\!\int P(y^*,\, \boldsymbol{x}^* \,|\, \alpha,\, \boldsymbol{\theta})\, P(\alpha,\, \boldsymbol{\theta} \,|\, y^{\text{train}},\, \boldsymbol{x}^{\text{train}})\, d\alpha\, d\boldsymbol{\theta}}{\int\!\int P(\boldsymbol{x}^* \,|\, \alpha,\, \boldsymbol{\theta})\, P(\alpha,\, \boldsymbol{\theta} \,|\, y^{\text{train}},\, \boldsymbol{x}^{\text{train}})\, d\alpha\, d\boldsymbol{\theta}} \tag{2.5}$$

The posterior, $P(\alpha,\, \boldsymbol{\theta} \,|\, y^{\text{train}},\, \boldsymbol{x}^{\text{train}})$, is proportional to the product of the prior and the likelihood:

$$P(\alpha,\,\boldsymbol{\theta}\,|\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}) \;\propto\; P(\alpha,\,\boldsymbol{\theta})\;P(y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}\,|\,\alpha,\,\boldsymbol{\theta}) \tag{2.6}$$

$$\propto\; P(\alpha)\,\prod_{t=1}^{p} P(\theta_t\,|\,\alpha)\,\prod_{i=1}^{n} P(y^{(i)},\,\boldsymbol{x}^{(i)}\,|\,\alpha,\,\boldsymbol{\theta}) \tag{2.7}$$

where the second expression makes use of the conditional independence properties of the model.

When we use a subset of only k features, the predictive distribution for a test case will be

$$P(y^*\,|\,\boldsymbol{x}^*_{1:k},\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}_{1:k},\,\mathcal{S}) = \frac{\int\!\int P(y^*,\,\boldsymbol{x}^*_{1:k}\,|\,\alpha,\,\boldsymbol{\theta}_{1:k})\,P(\alpha,\,\boldsymbol{\theta}_{1:k}\,|\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}_{1:k},\,\mathcal{S})\,d\alpha\,d\boldsymbol{\theta}_{1:k}}{\int\!\int P(\boldsymbol{x}^*_{1:k}\,|\,\alpha,\,\boldsymbol{\theta}_{1:k})\,P(\alpha,\,\boldsymbol{\theta}_{1:k}\,|\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}_{1:k},\,\mathcal{S})\,d\alpha\,d\boldsymbol{\theta}_{1:k}} \tag{2.8}$$

where $\mathcal{S}$ represents the information regarding selection from (2.3), namely $|\mathrm{COR}(y^{\mathrm{train}}, x^{\mathrm{train}}_t)| \leq \gamma$ for $t = k{+}1,\ldots,p$. The posterior distribution for α and $\boldsymbol{\theta}_{1:k}$ needed for this prediction can be written as follows, in terms of an integral (or sum) over the values of the latent variables, $\boldsymbol{z}^{\mathrm{train}}$:

$$P(\alpha,\,\boldsymbol{\theta}_{1:k}\,|\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}_{1:k},\,\mathcal{S})$$

$$\propto\; \int P(\alpha,\,\boldsymbol{\theta}_{1:k},\,\boldsymbol{z}^{\mathrm{train}}\,|\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}_{1:k},\,\mathcal{S})\,d\boldsymbol{z}^{\mathrm{train}} \tag{2.9}$$

$$\propto\; \int P(\alpha,\,\boldsymbol{\theta}_{1:k})\;P(\boldsymbol{z}^{\mathrm{train}},\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}_{1:k}\,|\,\alpha,\,\boldsymbol{\theta}_{1:k})\,\cdot \\ P(\mathcal{S}\,|\,\alpha,\,\boldsymbol{\theta}_{1:k},\,\boldsymbol{z}^{\mathrm{train}},\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}_{1:k})\,d\boldsymbol{z}^{\mathrm{train}} \tag{2.10}$$

$$\propto\; \int P(\alpha,\,\boldsymbol{\theta}_{1:k})\;P(\boldsymbol{z}^{\mathrm{train}},\,y^{\mathrm{train}},\,\boldsymbol{x}^{\mathrm{train}}_{1:k}\,|\,\alpha,\,\boldsymbol{\theta}_{1:k})\;P(\mathcal{S}\,|\,\alpha,\,\boldsymbol{z}^{\mathrm{train}},\,y^{\mathrm{train}})\,d\boldsymbol{z}^{\mathrm{train}} \tag{2.11}$$

Here again, the conditional independence properties of the model justify removing the conditioning on $\boldsymbol{\theta}_{1:k}$ and $\boldsymbol{x}_{1:k}^{\text{train}}$ in the last factor.

Computation of $P(\mathcal{S}\,|\,\alpha,\, \boldsymbol{z}^{\text{train}},\, y^{\text{train}})$, which adjusts the likelihood to account for feature selection, is crucial to applying our method. Two facts greatly ease this computation. First, the x_t are conditionally independent given α, $\boldsymbol{z}$, and y, which allows us to write this as a product of factors pertaining to the various omitted features. Second, these factors are *all the same*, since nothing distinguishes one omitted feature from another. Accordingly,

$$\begin{aligned} P(\mathcal{S}\,|\,\alpha,\, \boldsymbol{z}^{\text{train}},\, y^{\text{train}}) &= \prod_{t=k+1}^{p} P\Big(|\text{COR}(y^{\text{train}}, x_t^{\text{train}})| \,\leq\, \gamma\,|\,\alpha,\, \boldsymbol{z}^{\text{train}},\, y^{\text{train}}\Big) & (2.12) \\ &= \Big[P\Big(|\text{COR}(y^{\text{train}}, x_t^{\text{train}})| \,\leq\, \gamma\,|\,\alpha,\, \boldsymbol{z}^{\text{train}},\, y^{\text{train}}\Big)\Big]^{p-k} & (2.13) \end{aligned}$$

where in the second expression, t represents *any* of the omitted features. Since the time needed to compute the adjustment factor does not depend on the number of omitted features, we may hope to save a large amount of computation time by omitting many features.

Computing the single factor we do need is not trivial, however, since it involves integrals over θ_t and x_t^{train}. We can write

$$\begin{aligned} &P\Big(|\text{COR}(y^{\text{train}}, x_t^{\text{train}})| \,\leq\, \gamma\,|\,\alpha,\, \boldsymbol{z}^{\text{train}},\, y^{\text{train}}\Big) \\ &\quad = \int P(\theta_t\,|\,\alpha)\, P\Big(|\text{COR}(y^{\text{train}}, x_t^{\text{train}})| \,\leq\, \gamma\,|\,\alpha,\, \theta_t,\, \boldsymbol{z}^{\text{train}},\, y^{\text{train}}\Big)\, d\theta_t \qquad (2.14) \end{aligned}$$

Devising ways of efficiently performing this integral over θ_t and the integral over x_t^{train} implicit in the probability statement occurring in the integrand will be the main topic of our discussion of specific models below.

Once we have a way of computing this factor, we can use standard Markov chain Monte Carlo (MCMC) methods to sample from $P(\alpha,\, \boldsymbol{\theta}_{1:k},\, \boldsymbol{z}^{\text{train}}\,|\,y^{\text{train}},\, \boldsymbol{x}_{1:k}^{\text{train}},\, \mathcal{S})$. The resulting sample of values for α and $\boldsymbol{\theta}_{1:k}$ can be used to make predictions using equation (2.8), by approximating the integrals in the numerator and denominator by Monte Carlo estimates. For the naive Bayes model we will discuss in Section 2.3, however, Monte Carlo methods are unnecessary — a combination of analytical integration and numerical quadrature is faster.

2.3 Application to Bayesian Naive Bayes Models

In this chapter we show how to apply the bias correction method to Bayesian naive Bayes models in which both the features and the response are binary. Binary features are natural for some problems (e.g., test answers that are either correct or incorrect), or may result from thresholding real-valued features. Such thresholding can sometimes be beneficial — in a document classification problem, for example, whether or not a word is used at all may be more relevant to the class of the document than how many times it is used. Naive Bayes models assume that features are independent given the response. This assumption is often incorrect, but such simple naive Bayes models have nevertheless been found to work well for many practical problems (see for example Li and Jain 1998, Vaithyanathan, Mao, and Dom 2000, Eyheramendy, Lewis, and Madigan 2003). Here we show how to correct for selection bias in binary naive Bayes models, whose simplicity allows the required adjustment factor to be computed very quickly. Simulations reported in Section 2.3.5 show that substantial bias can be present with the uncorrected method, and that it is indeed corrected by conditioning on the fact that feature selection occurred. We then apply the method to real data on gene expression relating to colon cancer, and again find that our bias correction method improves predictions.

2.3.1 Definition of the Binary Naive Bayes Models

Let $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \cdots, x_p^{(i)})$ be the vector of p binary features for case i, and let $y^{(i)}$ be the binary response for case i, indicating the class. For example, $y^{(i)} = 1$ might indicates that cancer is present for patient i, and $y^{(i)} = 0$ indicate that cancer is not present. Cases are assumed

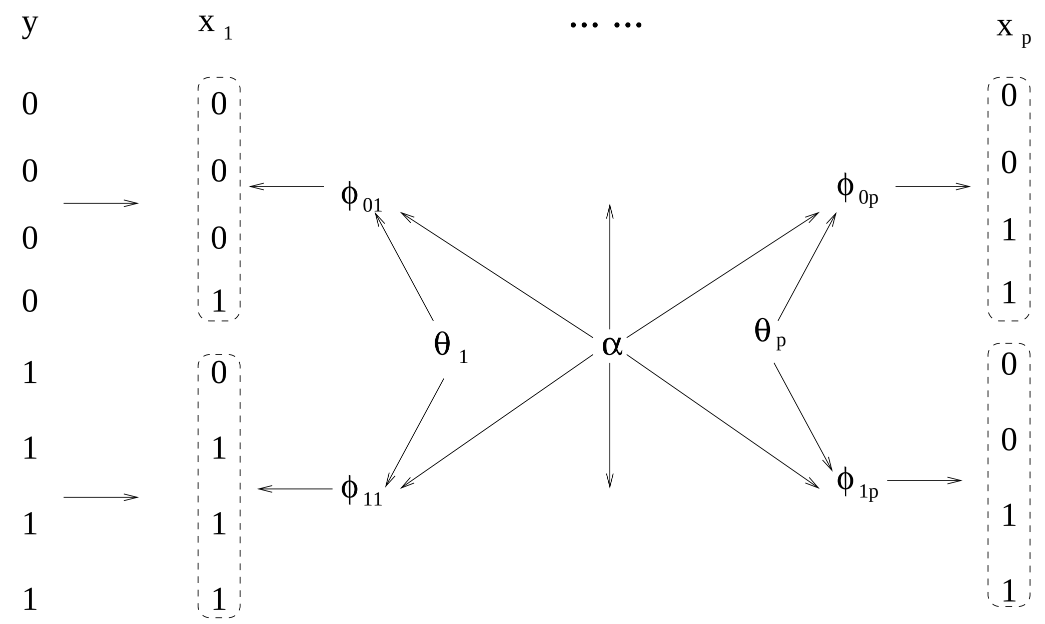


Figure 2.2: A picture of Bayesian naive Bayes models.

to be independent given the values of the model parameters (ie, exchangeable *a priori*). The probability that $y = 1$ in a case is given by the parameter ψ. Conditional on the class y in some case (and on the model parameters), the features $x_1, \ldots, x_p$ are assumed to be independent, and to have Bernoulli distributions with parameters $\phi_{y,1}, \ldots, \phi_{y,p}$, collectively written as $\boldsymbol{\phi}_y$, with $\boldsymbol{\phi} = (\boldsymbol{\phi}_0, \boldsymbol{\phi}_1)$ representing all such parameters. Figure 2.2 displays the models. Formally, the data is modeled as

$$\begin{aligned} y^{(i)} \mid \psi &\sim \text{Bernoulli}\,(\psi), \quad \text{for } i = 1, \ldots, n & (2.15) \\ x_j^{(i)} \mid y^{(i)}, \boldsymbol{\phi} &\sim \text{Bernoulli}\,(\phi_{y^{(i)},j}), \quad \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, p & (2.16) \end{aligned}$$

We use a hierarchical prior that expresses the possibility that some features may have almost the same distribution in the two classes. In detail, the prior has the following form:

$$\psi \quad \sim \quad \mathrm{Beta}\,(f_1, f_0) \tag{2.17}$$

$$\alpha \quad \sim \quad \text{Inverse-Gamma}(a, b) \tag{2.18}$$

$$\theta_1, \ldots, \theta_p \quad \overset{\text{IID}}{\sim} \quad \text{Uniform}(0, 1) \tag{2.19}$$

$$\phi_{0,j},\, \phi_{1,j} \mid \alpha,\, \theta_j \quad \overset{\text{IID}}{\sim} \quad \mathrm{Beta}\,(\alpha\theta_j,\, \alpha(1{-}\theta_j)), \quad \text{for } j = 1, \ldots, p \tag{2.20}$$

The hyperparameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ are used to introduce dependence between $\boldsymbol{\phi}_{0,j}$ and $\boldsymbol{\phi}_{1,j}$, with α controlling the degree of dependence. Features for which $\phi_{0,j}$ and $\phi_{1,j}$ differ greatly are more relevant to predicting the response. When α is small, the variance of the Beta distribution in (2.20), which is $\theta_j\,(1{-}\theta_j)\,/\,(\alpha{+}1)$, is large, and many features are likely to have predictive power, whereas when α is large, it is likely that most features will be of little use in predicting the response, since $\phi_{0,j}$ and $\phi_{1,j}$ are likely to be almost equal. We chose an Inverse-Gamma prior for α (with density function proportional to $\alpha^{-(1+a)}\exp(-b/\alpha)$) because it has a heavy upward tail, allowing for the possibility that α is large. Our method of correcting selection bias will have the effect of modifying the likelihood in a way that favors larger values for α than would result from ignoring the effect of selection.

2.3.2 Integrating Away ψ and $\boldsymbol{\phi}$

Although the above model is defined with ψ and $\boldsymbol{\phi}$ parameters for better conceptual understanding, computations are simplified by integrating them away analytically.

Integrating away ψ, the joint probability of $y^{\text{train}} = (y^{(1)}, \ldots, y^{(n)})$ is as follows, where $I(\,\cdot\,)$ is the indicator function, equal to 1 if the enclosed condition is true and 0 if it is false:

$$\begin{aligned} P(y^{\text{train}}) &= \int_0^1 \frac{\Gamma(f_0 + f_1)}{\Gamma(f_0)\Gamma(f_1)} \psi^{f_1} (1 - \psi)^{f_0}\; \psi^{\sum\limits_{i=1}^{n} I(y^{(i)}=1)} (1 - \psi)^{\sum\limits_{i=1}^{n} I(y^{(i)}=0)} d\psi & (2.21) \\ &= U\Big(f_1,\, f_0,\, \sum_{i=1}^{n} I\big(y^{(i)} = 1\big),\, \sum_{i=1}^{n} I\big(y^{(i)} = 0\big)\Big) & (2.22) \end{aligned}$$

The function U is defined as

$$U(f_1, f_0, n_1, n_0) = \frac{\Gamma(f_0 + f_1)}{\Gamma(f_0)\Gamma(f_1)} \frac{\Gamma(f_0 + n_0)\Gamma(f_1 + n_1)}{\Gamma(f_0 + f_1 + n_0 + n_1)} \tag{2.23}$$

$$= \frac{\prod_{\ell=1}^{n_0}(f_0 + \ell - 1) \prod_{\ell=1}^{n_1}(f_1 + \ell - 1)}{\prod_{\ell=1}^{n_0+n_1}(f_0 + f_1 + \ell - 1)} \tag{2.24}$$

The products above have the value one when the upper limits of n_0 or n_1 are zero. The joint probability of y^{train} and the response, y^*, for a test case is similar:

$$P(y^{\text{train}}, y^*)$$
$$= U\Big(f_1,\, f_0,\, \sum_{i=1}^{n} I\big(y^{(i)} = 1\big) + I(y^* = 1),\, \sum_{i=1}^{n} I\big(y^{(i)} = 0\big) + I(y^* = 0)\Big) \tag{2.25}$$

Dividing $P(y^{\text{train}}, y^*)$ by $P(y^{\text{train}})$ gives

$$P(y^* \mid y^{\text{train}}) = \text{Bernoulli}\,(y^*; \hat{\psi}) \tag{2.26}$$

Here, $\text{Bernoulli}\,(y; \psi) = \psi^y\,(1 - \psi)^{1-y}$ and $\hat{\psi} = (f_1 + N_1)\,/\,(f_0 + f_1 + n)$, with $N_y = \sum_{\ell=1}^{n} I(y^{(\ell)} = y)$. Note that $\hat{\psi}$ is just the posterior mean of ψ based on $y^{(1)}, \ldots, y^{(n)}$.

Similarly, integrating over $\phi_{0,j}$ and $\phi_{1,j}$, we find that

$$P(x_j^{\text{train}} \mid \theta_j,\, \alpha,\, y^{\text{train}}) = \prod_{y=0}^{1} U(\alpha\theta_j,\, \alpha(1-\theta_j),\, I_{y,j},\, O_{y,j}) \tag{2.27}$$

where $O_{y,j} = \sum_{i=1}^{n} I(y^{(i)} = y,\, x_j^{(i)} = 0)$ and $I_{y,j} = \sum_{i=1}^{n} I(y^{(i)} = y,\, x_j^{(i)} = 1)$.

With ψ and $\boldsymbol{\phi}$ integrated out, we need deal only with the remaining parameters, α and $\boldsymbol{\theta}$. Note that after eliminating ψ and the $\boldsymbol{\phi}$, the cases are no longer independent (though they are exchangeable). However, conditional on the responses, y^{train}, and on α, the values of different features are still independent. This is crucial to the efficiency of the computations described below.

2.3.3 Predictions for Test Cases using Numerical Quadrature

We first describe how to predict the class for a test case when we are either using all features, or using a subset of features without any attempt to correct for selection bias. We then consider how to make predictions using our method of correcting for selection bias.

Suppose we wish to predict the response, y^*, in a test case for which we know the retained features $\boldsymbol{x}^*_{1:k} = (\boldsymbol{x}^*_1, \cdots, \boldsymbol{x}^*_k)$ (having renumbered features as necessary). For this, we need the following predictive probability:

$$P(y^* \,|\, \boldsymbol{x}^*_{1:k},\, \boldsymbol{x}^{\text{train}}_{1:k},\, y^{\text{train}}) \;=\; \frac{P(y^* \,|\, y^{\text{train}})\, P(\boldsymbol{x}^*_{1:k} \,|\, y^*,\, \boldsymbol{x}^{\text{train}}_{1:k},\, y^{\text{train}})}{\sum\limits_{y=0}^{1} P(y^* = y \,|\, y^{\text{train}})\, P(\boldsymbol{x}^*_{1:k} \,|\, y^* = y,\, \boldsymbol{x}^{\text{train}}_{1:k},\, y^{\text{train}})} \tag{2.28}$$

Ie, we evaluate the numerator above for $y^* = 0$ and $y^* = 1$, then divide by the sum to obtain the predictive probabilities. The first factor in the numerator, $P(y^* \,|\, y^{\text{train}})$, is given by equation (2.26). It is sufficient to obtain the second factor up to a proportionality constant that doesn't depend on y^*, as follows:

$$\begin{aligned} P(\boldsymbol{x}^*_{1:k} \mid y^*,\, \boldsymbol{x}^{\text{train}}_{1:k},\, y^{\text{train}}) &= \frac{P(\boldsymbol{x}^*_{1:k},\, x^{\text{train}}_{1:k} \mid y^*,\, y^{\text{train}})}{P(x^{\text{train}}_{1:k} \mid y^{\text{train}})} && (2.29)\\ &\propto P(\boldsymbol{x}^*_{1:k},\, x^{\text{train}}_{1:k} \mid y^*,\, y^{\text{train}}) && (2.30) \end{aligned}$$

This can be computed by integrating over α, noting that conditional on α the features are independent:

$$\begin{aligned} P(\boldsymbol{x}^*_{1:k},\, x^{\text{train}}_{1:k} \mid y^*,\, y^{\text{train}}) &= \int P(\alpha)\, P(\boldsymbol{x}^*_{1:k},\, x^{\text{train}}_{1:k} \mid \alpha,\, y^*,\, y^{\text{train}})\, d\alpha && (2.31)\\ &= \int P(\alpha) \prod_{j=1}^{k} P(\boldsymbol{x}^*_j,\, x^{\text{train}}_j \mid \alpha,\, y^*,\, y^{\text{train}})\, d\alpha && (2.32) \end{aligned}$$

Each factor in the product above is found by using equation (2.27) and integrating over θ_j:

$$
\begin{aligned}
& P(\boldsymbol{x}_j^*,\, x_j^{\text{train}} \mid \alpha,\, y^*,\, y^{\text{train}}) \\
& \quad = \int_0^1 P(\boldsymbol{x}_j^* \mid \theta_j,\, \alpha,\, \boldsymbol{x}_j^{\text{train}},\, y^{\text{train}},\, y^*)\, P(\boldsymbol{x}_j^{\text{train}} \mid \theta_j,\, \alpha,\, y^{\text{train}})\, d\theta_j && (2.33) \\
& \quad = \int_0^1 \text{Bernoulli}\,(\boldsymbol{x}_j^*; \hat{\phi}_{y^*,j}) \prod_{y=0}^{1} U(\alpha\theta_j,\, \alpha(1-\theta_j),\, I_{y,j},\, O_{y,j})\, d\theta_j && (2.34)
\end{aligned}
$$

where $\hat{\phi}_{y^*,j} = (\alpha\theta_j + I_{y^*,j}) \ / \ (\alpha + N_{y^*})$, the posterior mean of $\phi_{y^*,j}$ given α and θ_j.

When using k features selected from a larger number, p, the predictions above, which are conditional on only $x_{1:k}^{\text{train}}$ and y^{train}, are not correct — we should also condition on the event, $\mathcal{S}$, that $|\text{COR}(y^{\text{train}}, x_j^{\text{train}})| \ \le \ \gamma$ for $j = k+1, \ldots, p$. We need to modify the predictive probability of equation (2.28) by replacing $P(\boldsymbol{x}_{1:k}^* \mid y^*,\, \boldsymbol{x}_{1:k}^{\text{train}},\, y^{\text{train}})$ with $P(\boldsymbol{x}_{1:k}^* \mid y^*,\, \boldsymbol{x}_{1:k}^{\text{train}},\, y^{\text{train}},\, \mathcal{S})$, which is proportional to $P(\boldsymbol{x}_{1:k}^*,\, \boldsymbol{x}_{1:k}^{\text{train}},\, \mathcal{S} \mid y^*,\, y^{\text{train}})$. Analogously to equations (2.31) and (2.32), we obtain

$$
\begin{aligned}
& P(\boldsymbol{x}_{1:k}^*,\, x_{1:k}^{\text{train}},\, \mathcal{S} \mid y^*,\, y^{\text{train}}) \\
& \quad = \int P(\alpha)\, P(\boldsymbol{x}_{1:k}^*,\, x_{1:k}^{\text{train}},\, \mathcal{S} \mid \alpha,\, y^*,\, y^{\text{train}})\, d\alpha && (2.35) \\
& \quad = \int P(\alpha)\, P(\mathcal{S} \mid \alpha,\, y^{\text{train}}) \prod_{j=1}^{k} P(\boldsymbol{x}_j^*,\, x_j^{\text{train}} \mid \alpha,\, y^*,\, y^{\text{train}})\, d\alpha && (2.36)
\end{aligned}
$$

The factors for the k retained features are computed as before, using equation (2.34). The additional correction factor that is needed (presented earlier as equation (2.13)) is

$$
\begin{aligned}
P(\mathcal{S} \mid \alpha,\, y^{\text{train}}) \ &= \ \prod_{j=k+1}^{p} P(|\text{COR}(y^{\text{train}}, x_j^{\text{train}})| \le \gamma \mid \alpha,\, y^{\text{train}}) && (2.37) \\
&= \ \Big[P(|\text{COR}(y^{\text{train}}, x_t^{\text{train}})| \le \gamma \mid \alpha,\, y^{\text{train}}) \Big]^{p-k} && (2.38)
\end{aligned}
$$

where t is any of the omitted features, all of which have the same probability of having a small correlation with y. We discuss how to compute this adjustment factor in the next section.

To see intuitively why this adjustment factor will correct for selection bias, recall that as discussed in Section (2.3.1), when α is small, features will be more likely to have a strong relationship with the response. If the likelihood of α is based only on the selected features, which have shown high correlations with the response in the training dataset, it will favor values of α that are inappropriately small. Multiplying by the adjustment factor, which favors larger values for α, undoes this bias.

We compute the integrals over α in equations (2.32) and (2.36) by numerical quadrature. We use the midpoint rule, applied to $u = F(\alpha)$, where F is the cumulative distribution function for the Inverse-Gamma(a, b) prior for α. The prior for u is uniform over $(0, 1)$, and so needn't be explicitly included in the integrand. With K points for the midpoint rule, the effect is that we average the value of the integrand, without the prior factor, for values of α that are the $0.5/K, 1.5/K, \ldots, 1 - 0.5/K$ quantiles of its Inverse-Gamma prior. For each α, we use Simpson's Rule to compute the one-dimensional integrals over θ_j in equation (2.34).

2.3.4 Computation of the Adjustment Factor for Naive Bayes Models

Our remaining task is to compute the adjustment factor of equation (2.38), which depends on the probability that a feature will have correlation less than γ in absolute value. Computing this seems difficult — we need to sum the probabilities of $\boldsymbol{x}_t^{\text{train}}$ given y^{train}, α and θ_t over all configurations of $\boldsymbol{x}_t^{\text{train}}$ for which $|\text{COR}(y^{\text{train}}, x_t^{\text{train}})| \leq \gamma$ — but the computation can be simplified by noticing that $\text{COR}(x_t^{\text{train}}, y^{\text{train}})$ can be written in terms of

$I_0 = \sum_{i=1}^n I(y^{(i)} = 0,\, x_t^{(i)} = 1)$ and $I_1 = \sum_{i=1}^n I(y^{(i)} = 1,\, x_t^{(i)} = 1)$, as follows:

$$\mathrm{COR}(x_t^{\mathrm{train}},\, y^{\mathrm{train}}) \;=\; \frac{\displaystyle\sum_{i=1}^{n} \left(y^{(i)} - \bar{y}\right) x_t^{(i)}}{\sqrt{\displaystyle\sum_{i=1}^{n} \left(y^{(i)} - \bar{y}\right)^2}\,\sqrt{\displaystyle\sum_{i=1}^{n} \left(x_t^{(i)} - \bar{x}_t\right)^2}} \tag{2.39}$$

$$= \frac{(0-\overline{y})\, I_0 \;+\; (1-\overline{y})\, I_1}{\sqrt{n\overline{y}(1-\overline{y})}\,\sqrt{I_0 + I_1 - (I_0+I_1)^2/n}} \tag{2.40}$$

We write the above as $\mathrm{Cor}(I_0, I_1, \overline{y})$, taking n as known. This function is defined for $0 \leq I_0 \leq n(1-\overline{y})$ and $0 \leq I_1 \leq n\overline{y}$.

Fixing n, $\overline{y}$, and γ, we can define the following sets of values for I_0 and I_1 (for some feature x_t) in terms of the resulting correlation with y:

$$L_0 \;=\; \{\, (I_0, I_1) \;:\; \mathrm{Cor}(I_0, I_1, \overline{y}) = 0 \,\} \tag{2.41}$$

$$L_+ \;=\; \{\, (I_0, I_1) \;:\; 0 < \mathrm{Cor}(I_0, I_1, \overline{y}) \leq \gamma \,\} \tag{2.42}$$

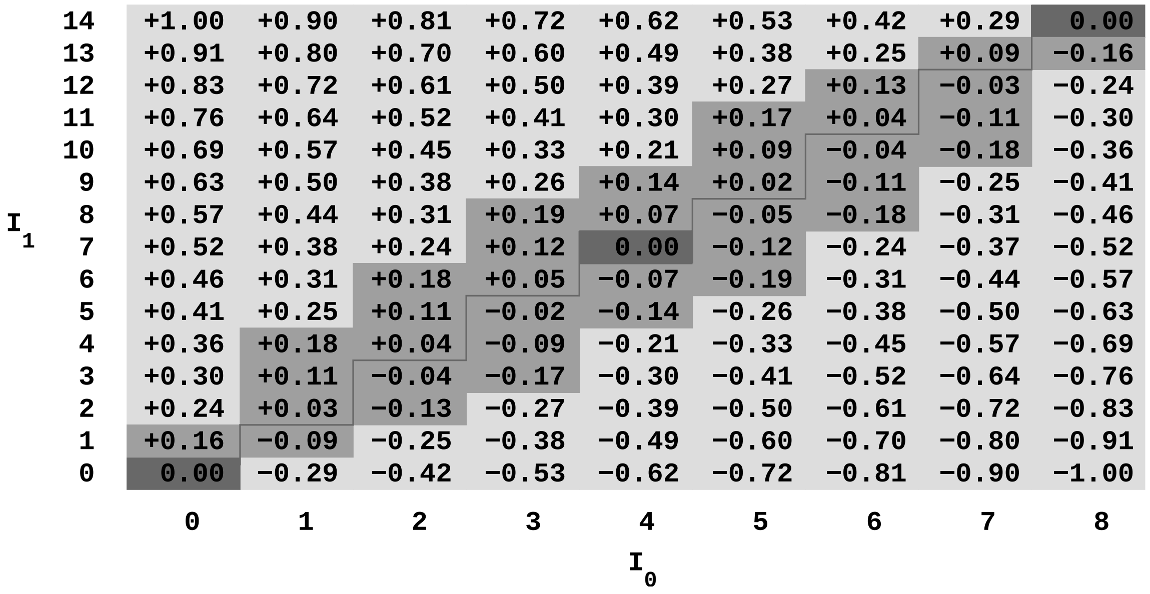
$$L_- \;=\; \{\, (I_0, I_1) \;:\; -\gamma \leq \mathrm{Cor}(I_0, I_1, \overline{y}) < 0 \,\} \tag{2.43}$$

$$H_+ \;=\; \{\, (I_0, I_1) \;:\; \gamma < \mathrm{Cor}(I_0, I_1, \overline{y}) \,\} \tag{2.44}$$

$$H_- \;=\; \{\, (I_0, I_1) \;:\; \mathrm{Cor}(I_0, I_1, \overline{y}) < -\gamma \,\} \tag{2.45}$$

A feature will be discarded if $(I_0, I_1) \in L_- \cup L_0 \cup L_+$ and retained if $(I_0, I_1) \in H_- \cup H_+$. These sets are illustrated in Figure 2.3.

We can write the probability needed in equation (2.38) using either L_-, L_0, and L_+ or



	14	+1.00	+0.90	+0.81	+0.72	+0.62	+0.53	+0.42	+0.29	0.00
	13	+0.91	+0.80	+0.70	+0.60	+0.49	+0.38	+0.25	+0.09	-0.16
	12	+0.83	+0.72	+0.61	+0.50	+0.39	+0.27	+0.13	-0.03	-0.24
	11	+0.76	+0.64	+0.52	+0.41	+0.30	+0.17	+0.04	-0.11	-0.30
	10	+0.69	+0.57	+0.45	+0.33	+0.21	+0.09	-0.04	-0.18	-0.36
	9	+0.63	+0.50	+0.38	+0.26	+0.14	+0.02	-0.11	-0.25	-0.41
I_1	8	+0.57	+0.44	+0.31	+0.19	+0.07	-0.05	-0.18	-0.31	-0.46
	7	+0.52	+0.38	+0.24	+0.12	0.00	-0.12	-0.24	-0.37	-0.52
	6	+0.46	+0.31	+0.18	+0.05	-0.07	-0.19	-0.31	-0.44	-0.57
	5	+0.41	+0.25	+0.11	-0.02	-0.14	-0.26	-0.38	-0.50	-0.63
	4	+0.36	+0.18	+0.04	-0.09	-0.21	-0.33	-0.45	-0.57	-0.69
	3	+0.30	+0.11	-0.04	-0.17	-0.30	-0.41	-0.52	-0.64	-0.76
	2	+0.24	+0.03	-0.13	-0.27	-0.39	-0.50	-0.61	-0.72	-0.83
	1	+0.16	-0.09	-0.25	-0.38	-0.49	-0.60	-0.70	-0.80	-0.91
	0	0.00	-0.29	-0.42	-0.53	-0.62	-0.72	-0.81	-0.90	-1.00
		0	1	2	3	4	5	6	7	8
						I_0				

Figure 2.3: The Cor function for a dataset with $n = 22$ and $\overline{y} = 14/22$. The values of $\text{Cor}(I_0, I_1, \overline{y})$ are shown for the valid range of I_0 and I_1. Using $\gamma = 0.2$, the values of (I_0, I_1) in L_0 are shown in dark grey, those in L_- or L_+ in medium grey, and those in H_- or H_+ in light grey.

H_- and H_+. We will take the latter approach here, as follows:

$$P(\,|\text{COR}(x_t^{\text{train}}, y^{\text{train}})| \leq \gamma \mid \alpha,\, y^{\text{train}})$$

$$= \;1 \;-\; P(\,(I_0, I_1) \;\in\; H_- \cup H+ \;\mid \alpha,\, y^{\text{train}}) \tag{2.46}$$

$$= \;1 \;-\; \sum_{(I_0, I_1) \,\in\, H_- \cup H_+} P(I_0,\, I_1 \mid \alpha,\, y^{\text{train}}) \tag{2.47}$$

We can now exploit symmetries of the prior and of the Cor function to speed up computation. First, note that $\text{Cor}(I_0, I_1, \overline{y}) \;=\; -\text{Cor}(n(1-\overline{y}) - I_0, n\overline{y} - I_1, \overline{y})$, as can be derived from equation (2.40), or by simply noting that exchanging labels for the classes should change only the sign of the correlation. The one-to-one mapping $(I_0, I_1) \rightarrow (n(1-\overline{y}) - I_0, n\overline{y} - I_1)$, which maps H_- and H_+ and vice versa (similarly for L_- and L_+), therefore leaves Cor unchanged. The priors for θ and ϕ (see (2.19) and (2.20)) are symmetrical with respect to the class labels 0 and 1, so the prior probability of $(I_0,\, I_1)$ is the same as that of $(n(1-\overline{y}) - I_0,\, n\overline{y} - I_1)$. We can therefore rewrite equation (2.47) as

$$P(\,|\mathrm{COR}(x_t^{\mathrm{train}}, y^{\mathrm{train}})| \le \gamma \mid \alpha,\, y^{\mathrm{train}}) \;=\; 1 \;-\; 2 \sum_{(I_0,I_1)\,\in\, H_+} P(I_0,\, I_1 \mid \alpha,\, y^{\mathrm{train}}) \qquad (2.48)$$

At this point we write the probabilities for I_0 and I_1 in terms of an integral over θ_t, and then swap the order of summation and integration, obtaining

$$\sum_{(I_0,I_1)\,\in\, H_+} P(I_0,\, I_1 \mid \alpha,\, y^{\mathrm{train}}) \;=\; \int_0^1 \sum_{(I_0,I_1)\,\in\, H_+} P(I_0,\, I_1 \mid \alpha,\, \theta_t,\, y^{\mathrm{train}})\; d\theta_t \qquad (2.49)$$

The integral over θ_t can be approximated using some one-dimensional numerical quadrature method (we use Simpson's Rule), provided we can evaluate the integrand.

The sum over H_+ can easily be delineated because $\mathrm{Cor}(I_0, I_1, \overline{y})$ is a monotonically decreasing function of I_0, and a monotonically increasing function of I_1, as may be confirmed by differentiating with respect to I_0 and I_1. Let b_0 be the smallest value of I_1 for which $\mathrm{Cor}(0, I_1, \overline{y}) > \gamma$. Taking the ceiling of the solution of $\mathrm{Cor}(0, I_1, \overline{y}) = \gamma$, we find that $b_0 = \lceil 1/(1/n + (1-\overline{y})/(n\overline{y}\gamma^2)) \rceil$. For $b_0 \le I_1 \le n\overline{y}$, let r_{I_1} be the largest value of I_0 for which $\mathrm{Cor}(I_0, I_1, \overline{y}) > \gamma$. We can write

$$\sum_{(I_0,I_1)\,\in\, H_+} P(I_0,\, I_1 \mid \alpha,\, \theta_t,\, y^{\mathrm{train}}) \;=\; \sum_{I_1=b_0}^{n\overline{y}} \sum_{I_0=0}^{r_{I_1}} P(I_0,\, I_1 \mid \alpha,\, \theta_t,\, y^{\mathrm{train}}) \qquad (2.50)$$

Given α and θ_t, I_0 and I_1 are independent, so we can reduce the computation needed by rewriting the above expression as follows:

$$\begin{aligned} &\sum_{(I_0,I_1)\,\in\, H_+} P(I_0,\, I_1 \mid \alpha,\, \theta_t,\, y^{\mathrm{train}}) \\ &= \sum_{I_1=b_0}^{n\overline{y}} P(I_1 \mid \alpha,\, \theta_t,\, y^{\mathrm{train}}) \sum_{I_0=0}^{r_{I_1}} P(I_0 \mid \alpha,\, \theta_t,\, y^{\mathrm{train}}) \end{aligned} \qquad (2.51)$$

Note that the inner sum can be updated from one value of I_1 to the next by just adding any additional terms needed. This calculation therefore requires $1+n\overline{y}-b_0 \leq n$ evaluations of $P(I_1 \mid \alpha,\, \theta_t,\, y^{\text{train}})$ and $1+r_{n\overline{y}} \leq n$ evaluations of $P(I_0 \mid \alpha,\, \theta_t,\, y^{\text{train}})$.

To compute $P(I_1 \mid \alpha,\, \theta_t,\, y^{\text{train}})$, we multiply the probability of any particular value for x_t^{train} in which there are I_1 cases with $y=1$ and $x_t=1$ by the number of ways this can occur. The probabilities are found by integrating over $\phi_{0,t}$ and $\phi_{1,t}$, as described in Section 2.3.2. The result is

$$P(I_1 \mid \alpha,\, \theta_t,\, y^{\text{train}}) \;=\; \binom{n\overline{y}}{I_1} U(\alpha\theta_t,\, \alpha(1-\theta_t),\, I_1,\, n\overline{y}-I_1) \tag{2.52}$$

Similarly,

$$P(I_0 \mid \alpha,\, \theta_t,\, y^{\text{train}}) \;=\; \binom{n(1-\overline{y})}{I_0} U(\alpha\theta_t,\, \alpha(1-\theta_t),\, I_0,\, n(1-\overline{y})-I_0) \tag{2.53}$$

One can easily derive simple expressions for $P(I_1 \mid \alpha,\, \theta_t,\, y^{\text{train}})$ and $P(I_0 \mid \alpha,\, \theta_t,\, y^{\text{train}})$ in terms of $P(I_1-1 \mid \alpha,\, \theta_t,\, y^{\text{train}})$ and $P(I_0-1 \mid \alpha,\, \theta_t,\, y^{\text{train}})$, which avoid the need to compute gamma functions or large products for each value of I_0 or I_1 when these values are used sequentially, as in equation (2.51).

2.3.5 A Simulation Experiment

In this section, we use a dataset generated from the naive Bayes model defined in Section 2.3.1 to demonstrate the lack of calibration that results when only a subset of features is used, without correcting for selection bias. We show that our bias-correction method eliminates this lack of calibration. We will also see that for the naive Bayes model only a small amount of extra computational time is needed to compute the adjustment factor needed by our method.

Fixing $\alpha=300$, and $p=10000$, we used equations (2.16), (2.19) and (2.20) to generate a

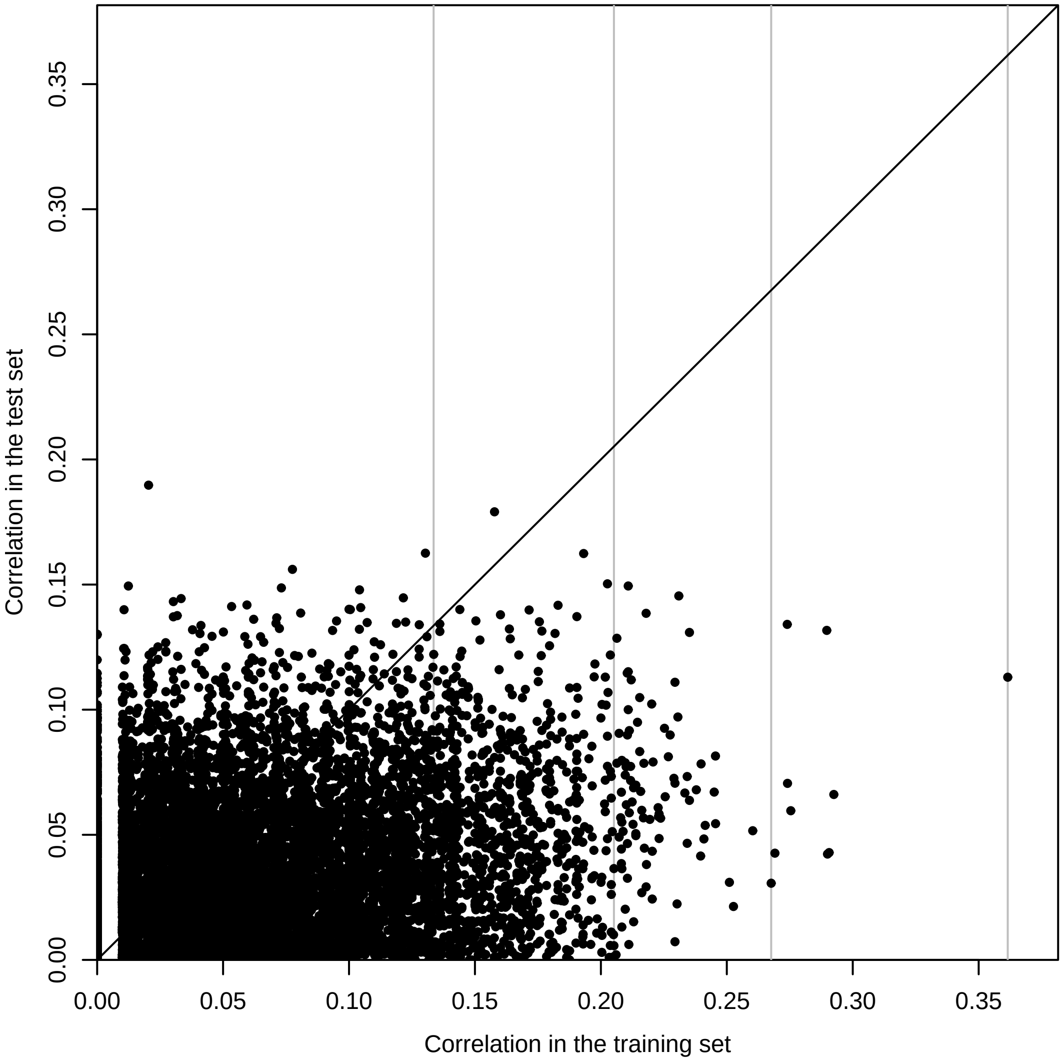


Figure 2.4: The absolute value of the sample correlation of each feature with the binary response, in the training set, and in the test set. Each dot represents one of the 10000 binary features. The training set correlations of the 1st, 10th, 100th, and 1000th most correlated features are marked by vertical lines.

set of 200 training cases and a set of 2000 test cases, both having equal numbers of cases with $y = 0$ and $y = 1$. We then selected four subsets of features, containing 1, 10, 100, and 1000 features, based on the absolute values of the sample correlations of the features with y. The smallest correlation (in absolute value) of a selected feature with the class was 0.36, 0.27, 0.21, and 0.13 for these four subsets. These are the values of γ used by the bias correction method when computing the adjustment factor of equation (2.38). Figure 2.4 shows the absolute value of the sample correlation in the training set of all 10000 features, plotted against the sample correlation in the test set. As can be seen, the high sample correlation of many selected features in the training set is partly or wholly a matter of chance, with the sample correlation in the test set (which is close to the real correlation) often being much less. The role of chance is further illustrated by the fact that the feature with highest sample correlation in the test set is not even in the top 1000 by sample correlation in the training set.

For each number of selected features, we fit this data using the naive Bayes model with the prior for ψ (equation (2.17)) having $f_0 = f_1 = 1$ and the prior for α (equation (2.18)) having shape parameter $a = 0.5$ and rate parameter $b = 5$. We then made predictions for the test cases using the methods described in Section 2.3.3. The "uncorrected" method, based on equation (2.28), makes no attempt to correct for the selection bias, whereas the "corrected" method, with the modification of equation (2.36), produces predictions that account for the procedure used to select the subset of features. We also made predictions using all 10000 features, for which bias correction is unnecessary.

We compared the predictive performance of the corrected method with the uncorrected method in several ways. First, we looked at the error rate when classifying test cases by thresholding the predictive probabilities at 1/2. As can be seen in Figure 2.5, there is little difference in the error rates with and without correction for bias. However, the methods differ drastically in terms of the *expected* error rate — the error rate we would expect based

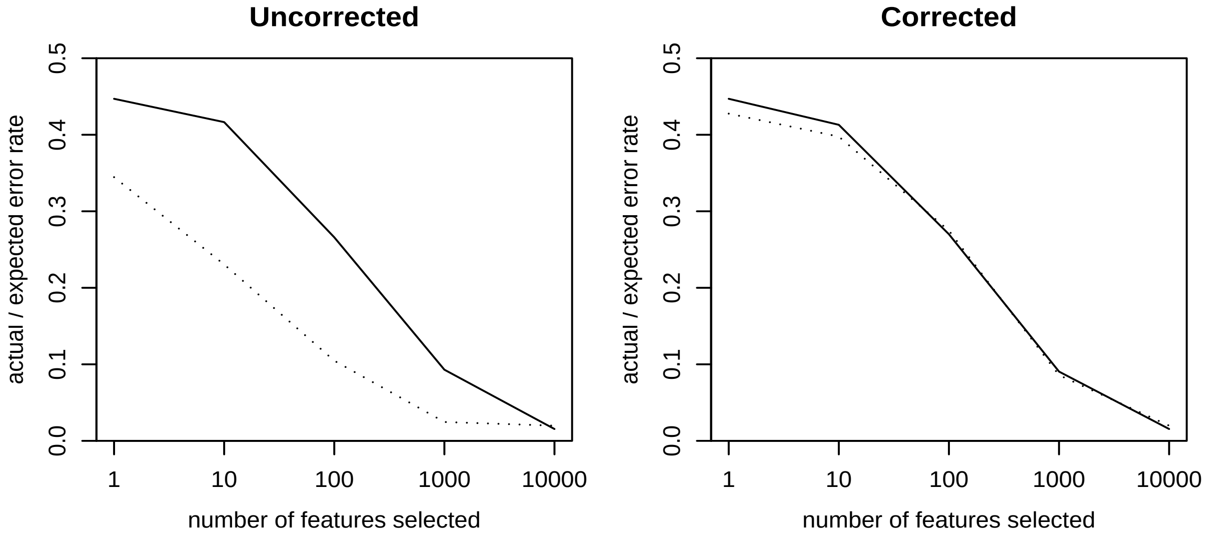


Figure 2.5: Actual and expected error rates with varying numbers of features selected, with and without correction for selection bias. The solid line is the actual error rate on test cases. The dotted line is the error rate that would be expected based on the predictive probabilities.

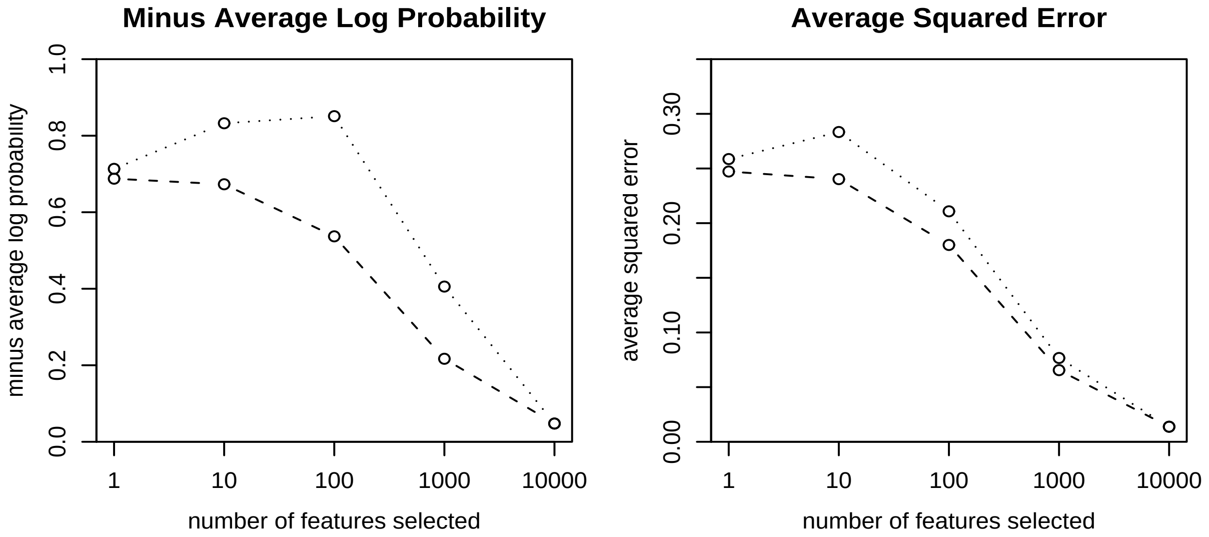


Figure 2.6: Performance in terms of average minus log probability and average squared error, with varying numbers of features selected, with and without correction for selection bias. The left plot shows minus the average log probability of the correct class for test cases, with 1, 10, 100, 1000, and all 10000 features selected. The dashed line is with bias correction, the dotted line without. The right plot is similar, but shows average squared error on test cases. Note that when all 10000 features are used, there is no difference between the corrected and uncorrected methods.

on the predictive probabilities for the test cases, equal to $(1/N)\sum_i \hat{p}^{(i)}\, I(\hat{p}^{(i)} < 0.5) + (1-\hat{p}^{(i)})\, I(\hat{p}^{(i)} \geq 0.5)$, where $\hat{p}^{(i)}$ is the predictive probability of class 1 for test case i. The predictive probabilities produced by the uncorrected method would lead us to believe that we would have a much lower error rate than the actual performance. In contrast, the expected error rates based on the predictive probabilities produced using bias correction closely match the actual error rates.

Two additional measures of predictive performance are shown in Figure 2.6. One measure of performance is minus the average log probability of the correct class in the N test cases, which is $-(1/N)\sum_{i=1}^{N}[y^{(i)}\log(\hat{p}^{(i)}) + (1-y^{(i)})\log(1-\hat{p}^{(i)})]$. This measure heavily penalizes test cases where the actual class has a predictive probability near zero. Another measure, less sensitive to such drastic errors, is the average squared error between the actual class (0 or 1) and the probability of class 1, given by $(1/N)\sum_{i=1}^{N}(y^{(i)} - \hat{p}^{(i)})^2$. The corrected method outperforms the uncorrected one by both these measures, with the difference being greater for minus average log probability. Interestingly, performance of the uncorrected method actually gets worse when going from 1 feature to 10 features. This may be because the single feature with highest sample correlation with the response does have a strong relationship with the response (as may be likely in general), whereas some other of the top 10 features by sample correlation have little or no real relationship.

We also looked in more detail at how well calibrated the predictive probabilities were. Table 2.1 shows the average predictive probability for class 1 and the actual fraction of cases in class 1 for test cases grouped according to the first decimal of their predictive probabilities, for both the uncorrected and the corrected method. Results are shown using subsets of 1, 10, 100, and 1000 features, and using all features. We see that the uncorrected method produces overconfident predictive probabilities, either too close to zero or too close to one. The corrected method avoids such bias (the values for "Pred" and "Actual" are much closer), showing that it is well calibrated.

	1 feature selected out of 10000						**10 features selected out of 10000**					
	Corrected			Uncorrected			Corrected			Uncorrected		
C	#	Pred	Actual	#	Pred	Actual	#	Pred	Actual	#	Pred	Actual
0	0	–	–	0	–	–	0	–	–	237	0.046	0.312
1	0	–	–	0	–	–	3	0.174	0.000	349	0.149	0.444
2	0	–	–	0	–	–	126	0.270	0.294	68	0.249	0.500
3	0	–	–	1346	0.384	0.461	467	0.360	0.420	300	0.360	0.443
4	1346	0.446	0.461	0	–	–	566	0.462	0.461	189	0.443	0.487
5	0	–	–	0	–	–	461	0.554	0.566	48	0.546	0.417
6	654	0.611	0.581	0	–	–	276	0.643	0.616	238	0.650	0.588
7	0	–	–	654	0.736	0.581	97	0.733	0.742	180	0.737	0.567
8	0	–	–	0	–	–	4	0.825	0.750	192	0.864	0.609
9	0	–	–	0	–	–	0	–	–	199	0.943	0.668

	100 features selected out of 10000						**1000 features selected out of 10000**					
	Corrected			Uncorrected			Corrected			Uncorrected		
C	#	Pred	Actual	#	Pred	Actual	#	Pred	Actual	#	Pred	Actual
0	155	0.067	0.077	717	0.017	0.199	774	0.018	0.027	954	0.004	0.066
1	247	0.151	0.162	133	0.150	0.391	97	0.143	0.165	28	0.149	0.500
2	220	0.247	0.286	70	0.251	0.429	63	0.243	0.302	13	0.248	0.846
3	225	0.352	0.356	68	0.351	0.515	48	0.346	0.438	17	0.349	0.412
4	237	0.450	0.494	58	0.451	0.500	45	0.446	0.600	14	0.449	0.786
5	227	0.545	0.586	78	0.552	0.603	44	0.547	0.614	16	0.546	0.375
6	202	0.650	0.728	77	0.654	0.532	53	0.647	0.698	16	0.667	0.812
7	214	0.749	0.785	80	0.746	0.662	81	0.755	0.815	22	0.751	0.636
8	182	0.847	0.857	98	0.852	0.633	124	0.854	0.863	25	0.865	0.560
9	91	0.935	0.923	621	0.979	0.818	671	0.977	0.982	895	0.995	0.946

	Complete data		
C	#	Pred	Actual
0	964	0.004	0.006
1	21	0.145	0.238
2	8	0.246	0.375
3	10	0.342	0.300
4	12	0.436	0.500
5	7	0.544	1.000
6	20	0.656	1.000
7	13	0.743	0.846
8	22	0.851	0.818
9	923	0.994	0.998

Table 2.1: Comparison of calibration for predictions found with and without correction for selection bias, on data simulated from the binary naive Bayes model. Results are shown with four subsets of features and with the complete data (for which no correction is necessary). The test cases were divided into 10 categories by the first decimal of the predictive probability of class 1, which is indicated by the 1st column "C". The table shows the number of test cases in each category for each method ("#"), the average predictive probability of class 1 for cases in that category ("Pred"), and the actual fraction of these cases that were in class 1 ("Actual").

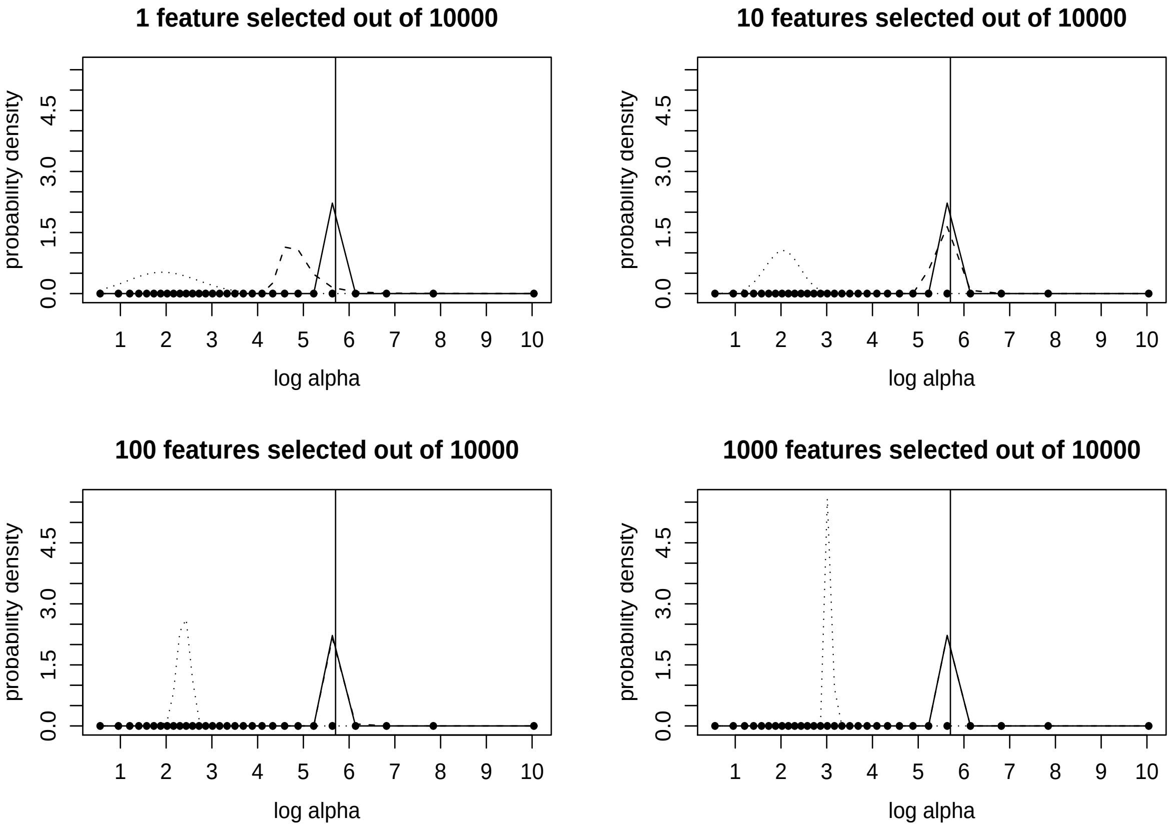


Figure 2.7: Posterior distributions of $\log(\alpha)$ for the simulated data, with different numbers of features selected. The true value of $\log(\alpha)$ is 5.7, shown by the vertical line. The solid line is the posterior density using all features. For each number of selected features, the dashed line is the posterior density including the factor that corrects for selection bias; the dotted line is the posterior density without bias correction. The dashed and solid lines overlap in the bottom two graphs. The dots mark the values of $\log(\alpha)$ used to approximate the density, at the $0.5/K, 1.5/K, \ldots, (K-0.5)/K$ quantiles of the prior distribution (where $K = 30$). The probabilities of x^{train} at each of these values for α were computed, rescaled to sum to K, and finally multiplied by the Jacobian, $\alpha P(\alpha)$, to obtain the approximation to the posterior density of $\log(\alpha)$

Number of Features Selected	1	10	100	1000	Complete data
Uncorrected Method	11	19	107	1057	10639
Corrected Method	12	19	107	1057	10639

Table 2.2: Computation times from simulation experiments with naive Bayes models

The biased predictions of the uncorrected method result from an incorrect posterior distribution for α, as illustrated in Figure 2.7. Without bias correction, the posterior based on only the selected features incorrectly favours values of α smaller than the true value of 300. Multiplying by the adjustment factor corrects this bias in the posterior distribution.

Our software (available from `http://www.utstat.utoronto.ca/∼longhai`) is written in the R language, with some functions for intensive computations such as numerical integration and computation of the adjustment factor written in C for speed. We approximated the integral with respect to α using the midpoint rule with $K = 30$ values for $F(\alpha)$, as discussed at the end of Section 2.3.3. The integrals with respect to θ in equations (2.34) and (2.49) were approximated using Simpson's Rule, evaluating θ at 21 points.

Computation times for each method (on a 1.2 GHz UltraSPARC III processor) are shown in Table 2.2. The corrected method is almost as fast as the uncorrected method, since the time to compute the adjustment factor is negligible compared to the time spent computing the integrals over θ_j for the selected features. Accordingly, considerable time can be saved by selecting a subset of features, rather than using all of them, without introducing an optimistic bias, though some accuracy in predictions may of course be lost when we discard the information contained in the unselected features.

2.3.6 A Test Using Gene Expression Data

We also tested our method using a publicly available dataset on gene expression in normal and cancerous human colon tissue. This dataset contains the expression levels of 6500 genes in 40 cancerous and 22 normal colon tissues, measured using the Affymetrix technology. The dataset is available at `http://geneexpression.cinj.org/∼notterman/affyindex.html`. We used only the 2000 genes with highest minimal intensity, as selected by Alon, Barkai, Notterman, Gish, Mack, and Levine (1999). In order to apply the binary naive Bayes model to the data, we transformed the real-value data into binary data by thresholding at the median, separately for each feature.

We divided these 2000 genes randomly into 10 equal groups, producing 10 smaller datasets, each with 200 features. We applies the corrected and uncorrected methods separately to each of these 10 datasets, allowing some assessment of variability when comparing performance. For each of these 10 datasets, we used leave-one-out cross validation to obtain the predictive probabilities over the 62 cases. In this cross-validation procedure, we left out each of the 62 cases in turn, selected the five features with the largest sample correlation with the response (in absolute value), and found the predictive probability for the left-out case using the binary naive Bayes model, with and without bias correction. The absolute value of the correlation of the last selected feature was around 0.5 in all cases. We used the same prior distribution, and the same computational methods, as for the demonstration in Section 2.3.5.

Figure 2.8 plots the predictive probabilities of class 1 for all cases, with each of the 10 subsets of features. The tendency of the uncorrected method to produce more extreme probabilities (closer to 0 and 1) is clear. However, when the predictive probability is close to 0.5, there is little difference between the corrected and uncorrected methods. Accordingly, the two methods almost always classify cases the same way, if prediction is made by thresholding the predictive probability at 0.5, and have very similar error rates. Note, however, that correcting for bias would have a substantial effect if cases were classified by thresholding the predictive probability at some value other than 0.5, as would be appropriate if the consequences of an error are different for the two classes.

Figure 2.10 compares the two methods in terms of average minus log probability of the correct class and in terms of average squared error. From these plots it is clear that bias correction improves the predictive probabilities. In terms of average minus log probability, the corrected method is better for all 10 datasets, and in terms of average squared error, the corrected method is better for 8 out of 10 datasets. (A paired t test with these two measures

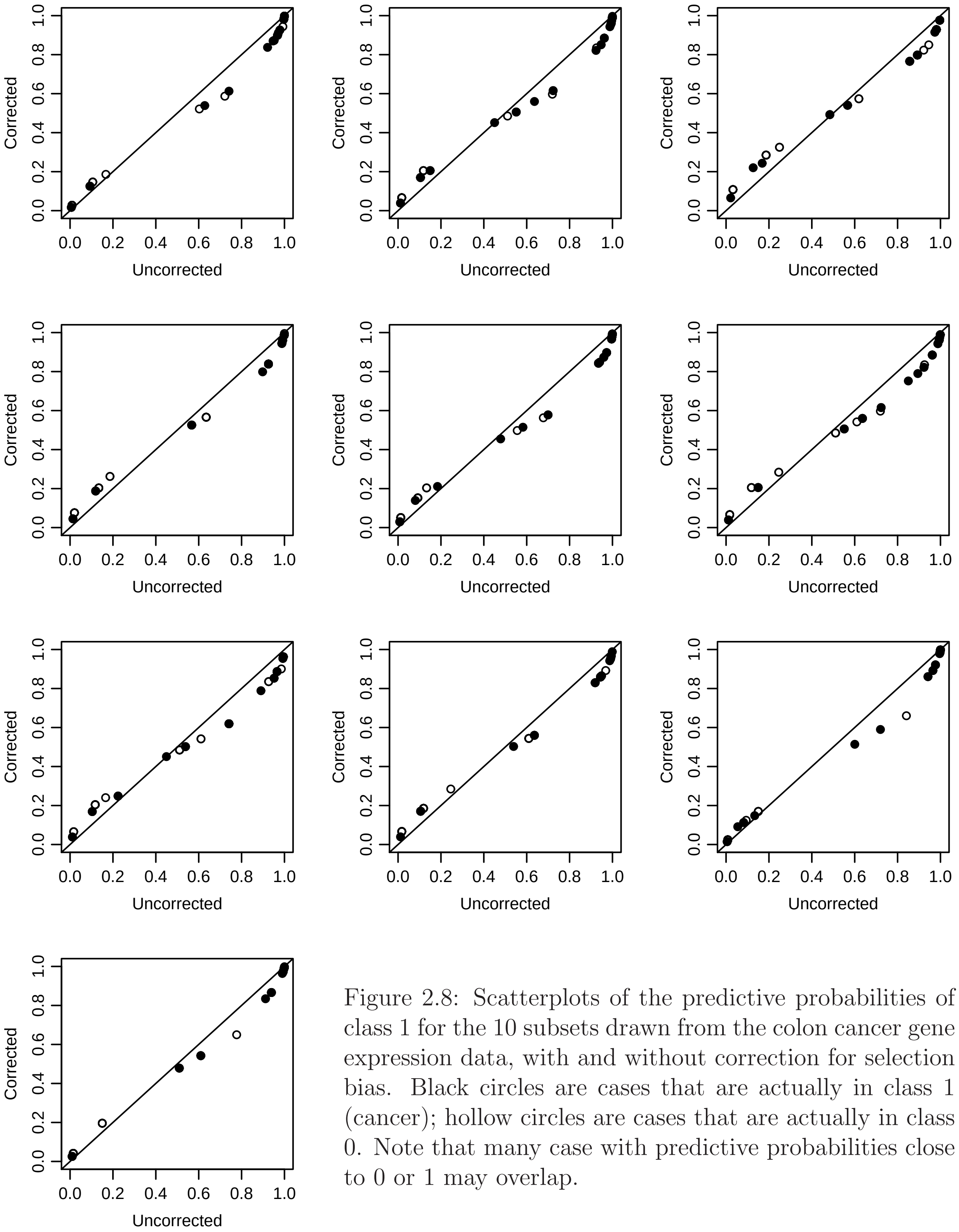


Figure 2.8: Scatterplots of the predictive probabilities of class 1 for the 10 subsets drawn from the colon cancer gene expression data, with and without correction for selection bias. Black circles are cases that are actually in class 1 (cancer); hollow circles are cases that are actually in class 0. Note that many case with predictive probabilities close to 0 or 1 may overlap.

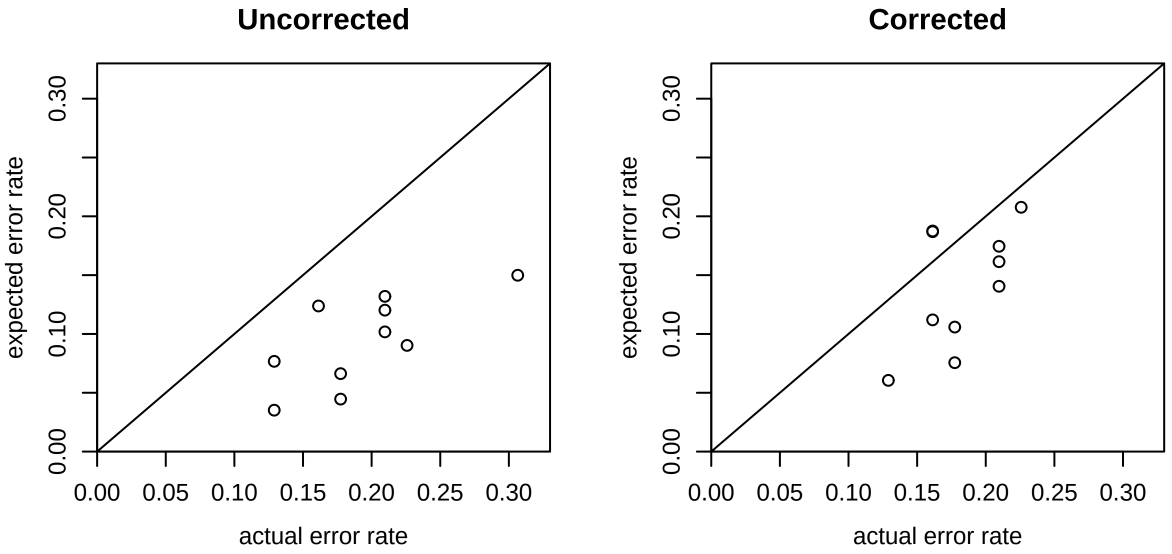


Figure 2.9: Actual versus expected error rates on the colon cancer datasets, with and without bias correction. Points are shown for each of the 10 subsets of features used for testing.

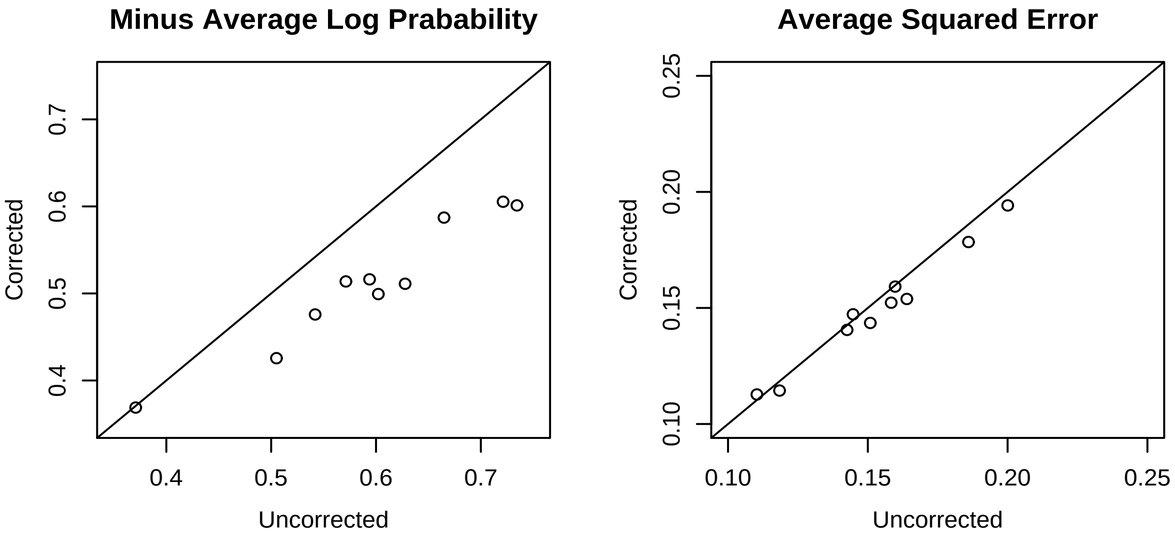


Figure 2.10: Scatterplots of the average minus log probability of the correct class and of the average squared error (assessed by cross validation) when using the 10 subsets of features for the colon cancer gene expression data, with and without correcting for selection bias.

produced p-values of 0.00007 and 0.019 respectively.)

Finally, Figure 2.9 shows that our bias correction method reduces optimistic bias in the predictions. For each of the 10 datasets, this plot shows the actual error rate (in the leave-one-out cross-validation assessment) and the error rate expected from the predictive probabilities. For all ten datasets, the expected error rate with the uncorrected method is substantially less than the actual error rate. This optimistic bias is reduced in the corrected method, though it is not eliminated entirely. The remaining bias presumably results from the failure in this dataset of the naive Bayes assumption that features are independent within a class.

2.4 Application to Bayesian Mixture Models

Mixture modelling is another way to model the joint distributions of the response variable and the predictor variables, from which we can find the conditional distribution of the response variable given the predictor variables. In this section we describe the application of the selection bias correction method to a class of binary mixture models, which is a generalization of the naive Bayes models.

2.4.1 Definition of the Binary Mixture Models

A complex distribution can be modeled using a mixture of finitely or infinitely many simple distributions, often called mixture components, for example, independent Gaussian distributions for real values or independent Bernoulli distributions for binary values. Mixture models are often applied in density estimation, classification, and latent class analysis problems, as discussed, for example, by Everitt and Hand (1981), McLachlan and Basford (1988), and Titterington, *et al.* (1985). For finite mixture models with K components, the density or

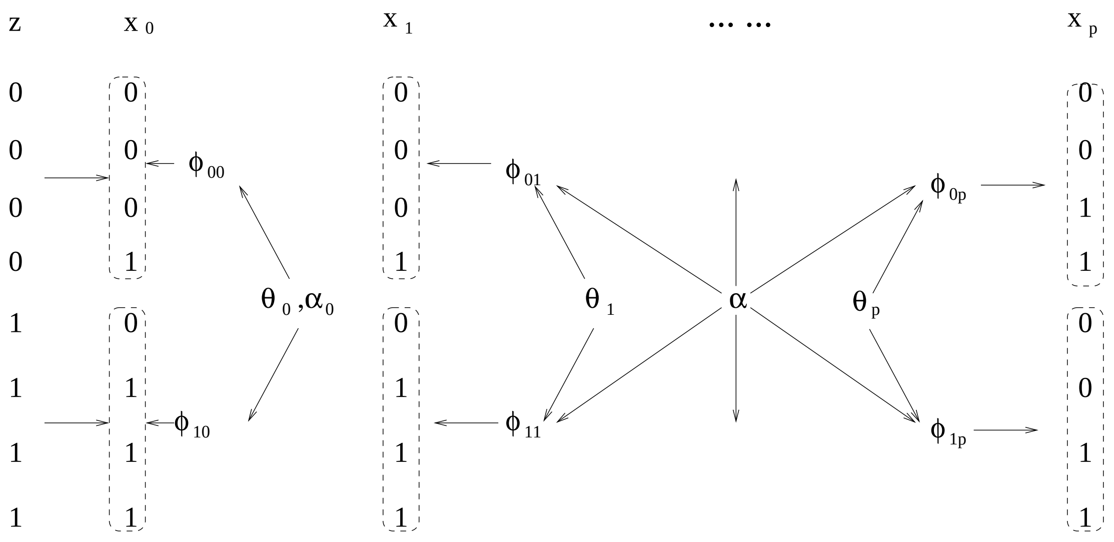


Figure 2.11: A picture of Bayesian binary mixture models

probability function of the observation $\boldsymbol{x}$ is written as

$$f(\boldsymbol{x} \mid \boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{K-1}) = \sum_{k=0}^{K-1} p_k \, f_k(\boldsymbol{x} \mid \boldsymbol{\phi}_k) \tag{2.54}$$

where p_k is the mixing proportion of component f_k, and $\boldsymbol{\phi}_k$ is the parameter associated with component f_k.

A Bayesian mixture model is often defined by introducing a latent label variable for each case, written as z. Given $z = k$, the conditional distribution of the observation $\boldsymbol{x}$ is the distribution for component k:

$$\boldsymbol{x} \mid z = k, \boldsymbol{\phi}_k \quad \sim \quad f_k(\boldsymbol{x} \mid \boldsymbol{\phi}_k) \tag{2.55}$$

We consider a two-component mixture model in this section, which is a generalization of the naive Bayes model in Section 2.3. Most of the notation is therefore the same as in that section, except that we use the 0th feature, x_0, to represent the response y here (and so x_0^{train} is equivalent to y^{train}) for convenience of presentation. The parameters of the Bernoulli distributions are $\phi_{z,0}$ (for the response x_0), and $\phi_{z,1}, \cdots, \phi_{z,p}$ (for the features), collectively denoted as $\boldsymbol{\phi}_z$, for $z = 0, 1$. The priors for $\boldsymbol{\phi}_0$ and $\boldsymbol{\phi}_1$ are assigned in the same way as for binary naive Bayes model. The labels of the cases are assigned a prior in the same way as for y in naive Bayes models. Conditional on the component labels, all the features and the response are assumed to be mutually independent. The models are displayed by Figure 2.11, and are described formally as follows:

$$\psi \sim \text{Beta}\,(f_1, f_0) \tag{2.56}$$

$$z^{(1)}, \cdots, z^{(n)} \mid \psi \overset{\text{IID}}{\sim} \text{Bernoulli}\,(\psi) \tag{2.57}$$

$$\alpha_0 \sim \text{Inverse-Gamma}(a_0, b_0) \tag{2.58}$$

$$\alpha \sim \text{Inverse-Gamma}(a, b) \tag{2.59}$$

$$\alpha_j = \begin{cases} \alpha_0 & \text{if } j = 0 \\ \alpha & \text{if } j > 0 \end{cases} \tag{2.60}$$

$$\theta_0, \theta_1, \cdots, \theta_p \overset{\text{IID}}{\sim} \text{Uniform}(0, 1) \tag{2.61}$$

$$\phi_{0,j}, \phi_{1,j} \mid \alpha_j, \theta_j \overset{\text{IID}}{\sim} \text{Beta}\,(\alpha_j\theta_j, \alpha_j(1-\theta_j)) \quad j = 0, 1, \cdots, p \tag{2.62}$$

$$\boldsymbol{x}_j^{(i)} \mid z^{(i)}, \phi_{z^{(i)},j} \sim \text{Bernoulli}\,(\phi_{z^{(i)},j}) \quad i = 1, \cdots, n \tag{2.63}$$

The naive Bayes models described in Section 2.3 can be seen as a simpler form of the above mixture models by letting $z^{(1)}, \cdots, z^{(n)}$ equal the responses $\boldsymbol{x}_0^{\text{train}}$, i.e., fixing $\phi_{0,0} = 1$ and $\phi_{1,0} = 0$. As for Bayesian naive Bayes models, ψ and $\theta_0, \ldots, \theta_p$ can be integrated away analytically from Bayesian binary mixture models. The cases are then no longer independent, but still are exchangeable. This integration reduces the number of the parameters, therefore improves Markov chain sampling or numerical quadrature if they are needed, though the resulting model may be harder to manipulate.

2.4.2 Predictions for Test Cases using MCMC

Let us start with no attempt to correct for the selection bias. We want to predict the response, $\boldsymbol{x}_0^*$, of a test case for which we know the retained features $x_1^*, \cdots, x_k^*$ (renumbering the features as necessary), based on the training data $\boldsymbol{x}_{0:k}^{\text{train}}$. For this, we need to calculate the predictive distribution:

$$P(x_0^* = 1 \mid \boldsymbol{x}_{1:k}^*, \boldsymbol{x}_{0:k}^{\text{train}}) = \frac{P(x_0^* = 1, \boldsymbol{x}_{1:k}^* \mid \boldsymbol{x}_{0:k}^{\text{train}})}{P(x_0^* = 1, \boldsymbol{x}_{1:k}^* \mid \boldsymbol{x}_{0:k}^{\text{train}}) + P(x_0^* = 0, \boldsymbol{x}_{1:k}^* \mid \boldsymbol{x}_{0:k}^{\text{train}})} \tag{2.64}$$

We therefore need to calculate $P(\boldsymbol{x}_{0:k}^* \mid \boldsymbol{x}_{0:k}^{\text{train}})$ for $x_0^* = 1$ and $x_0^* = 0$. $P(\boldsymbol{x}_{0:k}^* \mid \boldsymbol{x}_{0:k}^{\text{train}})$ can be written as:

$$\begin{aligned}
& P(\boldsymbol{x}_{0:k}^* \mid \boldsymbol{x}_{0:k}^{\text{train}}) \\
= \ & \sum_{\boldsymbol{z}^{\text{train}}} \int_{\alpha_0} \int_{\alpha} \int_{\boldsymbol{\theta}} P(\boldsymbol{x}_{0:k}^* \mid \boldsymbol{x}_{0:k}^{\text{train}}, \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}}) \ \cdot \\
& \qquad\qquad P(\boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}} \mid \boldsymbol{x}_{0:k}^{\text{train}}) \, d\boldsymbol{\theta}_{0:k} \, d\alpha \, d\alpha_0 \qquad (2.65) \\
= \ & \frac{1}{P(\boldsymbol{x}_{0:k}^{\text{train}})} \sum_{\boldsymbol{z}^{\text{train}}} \int_{\alpha_0} \int_{\alpha} \int_{\boldsymbol{\theta}} P(\boldsymbol{x}_{0:k}^* \mid \boldsymbol{x}_{0:k}^{\text{train}}, \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}}) \ \cdot \\
& \qquad P(\boldsymbol{x}_{0:k}^{\text{train}} \mid \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}}) \, P(\boldsymbol{\theta}_{0:k}) \, P(\alpha_0) \, P(\alpha) \, P(z^{\text{train}}) \, d\boldsymbol{\theta}_{0:k} \, d\alpha \, d\alpha_0 \quad (2.66)
\end{aligned}$$

The above integral is intractable analytically. We first approximate the integrals with respect to α_0 and α with the midpoint rule applied to the transformed variables $u_0 = F_0(\alpha_0)$ and $u = F(\alpha)$, where F_0 and F are the cumulative distribution functions of the priors for α_0 and α. Accordingly, the priors for u_0 and u are uniform over $(0, 1)$. Suppose the midpoint rule evaluates the integrands with respect to u_0 and u at K points respectively (K can be different for u_0 and u, for simplicity of presentation assume the same). After rewriting the summation with respect to u_0 and u over K points, $(1-0.5)/K, (2-0.5)/K, \ldots, (K-0.5)/K$, in terms of α_0 and α, the midpoint rule approximation is equivalent to summing the integrand in (2.66), without the prior distribution for α_0 and α, over the quantiles of the priors for

α_0 and α corresponding to probabilities $(1-0.5)/K, (2-0.5)/K, \ldots, (K-0.5)/K$. Let us denote the K quantiles of α_0 by $\mathcal{A}_0$, and denote the K quantiles of α by $\mathcal{A}$. The integral in (2.66), with $1/P(\boldsymbol{x}_{0:k})$ omitted (since it is the same proportionality factor for all x_0^*), is approximated by:

$$\sum_{\boldsymbol{z}^{\text{train}}} \sum_{\alpha_0 \in \mathcal{A}_0} \sum_{\alpha \in \mathcal{A}} \int_{\boldsymbol{\theta}} P(\boldsymbol{x}^*_{0:k} \mid \boldsymbol{x}^{\text{train}}_{0:k}, \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}}) \cdot \\ P(\boldsymbol{x}^{\text{train}}_{0:k} \mid \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}})\, P(\boldsymbol{\theta}_{0:k})\, P(\boldsymbol{z}^{\text{train}})\, d\boldsymbol{\theta}_{0:k} \qquad (2.67)$$

Since there are no prior terms in (2.67) for α_0 and α, the above approximation with midpoint rule applied to u_0 and u can also be seen as approximating the continuous Inverse-Gamma priors for α_0 and α by the uniform distributions over the finite sets $\mathcal{A}_0$ and $\mathcal{A}$. Based on these discretized priors for α_0 and α, we use Gibbs sampling method to draw samples from the posterior distribution of $\boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}}$, allowing the integral in (2.67) to be approximated with the Monte Carlo method. The reason we use such a discretization for the prior for α is to ease the computation of the adjustment factor, which depends on α. As will be discussed later, when α is discrete, we can cache the values of the adjustment factors for future use when the same α is used again.

We now start to derive the necessary formulae for performing Gibbs sampling for estimating (2.67). Using the results from Section 2.3.2, $P(\boldsymbol{x}^*_{0:k} \mid \boldsymbol{x}^{\text{train}}_{0:k}, \boldsymbol{\theta}_{0:k}, \alpha, \boldsymbol{z}^{\text{train}})$ in (2.67) can be calculated as follows:

$$\begin{aligned} & P(\boldsymbol{x}^*_{0:k} \mid \boldsymbol{x}^{\text{train}}_{0:k}, \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}}) \\ & \quad = \sum_{z^*=0}^{1} P(z^* \mid \boldsymbol{z}^{\text{train}})\, P(\boldsymbol{x}^*_{0:k} \mid \boldsymbol{x}^{\text{train}}_{0:k}, \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha, \boldsymbol{z}^{\text{train}}, z^*) && (2.68) \\ & \quad = \sum_{z^*=0}^{1} \text{Bernoulli}\left(z^*; \hat{\psi}\right) \prod_{j=0}^{k} \text{Bernoulli}\,(x_j^*; \hat{\phi}_{z^*,j}) && (2.69) \end{aligned}$$

where $\hat{\phi}_{z^*,j} = (\alpha_j \theta_j + I_{z^*,j}) \;/\; (\alpha_j + n^{[z^*]})$, $\hat{\psi} = (f_1 + n^{[1]})/(f_0 + f_1 + n)$, $n^{[z]} = \sum_{i=1}^{n} I(z_i = z)$,

$I_{z,j} = \sum_{i=1}^{n} I(z^{(i)} = z, x_j^{(i)} = 1)$, and $O_{z,j} = \sum_{i=1}^{n} I(z^{(i)} = z, x_j^{(i)} = 0)$.

Again, using the results from Section 2.3.2, the distribution of $\boldsymbol{x}_{0:k}^{(i)}$ given other training cases, denoted by $\boldsymbol{x}_{0:k}^{(-i)}$, which is needed to update $z^{(i)}$ with Gibbs sampling, can be found:

$$P(\boldsymbol{x}_{0:k}^{(i)} \mid \boldsymbol{x}_{0:k}^{(-i)}, \boldsymbol{z}^{\text{train}}, \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha) \;=\; \prod_{j=0}^{k} \text{Bernoulli}\,(x_j^{(i)}; \hat{\phi}_{z^{(i)},j}^{(-i)}) \tag{2.70}$$

where $\hat{\phi}_{z^{(i)},j}^{(-i)} = (\alpha_j \theta_j + I_{z^{(i)},j}^{(-i)})/(\alpha_j + n^{[z^{(i)}](-i)} - 1)$, $I_{z^{(i)},j}^{(-i)} = \sum_{s=1}^{n} I(x_j^{(s)} = 1, z^{(s)} = z^{(i)}, s \neq i)$ and $n^{[z^{(i)}](-i)} = \sum_{s=1}^{n} I(z^{(s)} = z^{(i)}, s \neq i)$.

Similarly, the distribution of the whole training data given $\boldsymbol{z}^{\text{train}}$, $\boldsymbol{\theta}_{0:k}, \alpha_0, \alpha$ based on the k retained features can be found:

$$P(\boldsymbol{x}_{0:k}^{\text{train}} \mid \boldsymbol{z}^{\text{train}},\; \alpha_0, \alpha,\; \boldsymbol{\theta}_{0:k}) \;=\; \prod_{j=0}^{k} \prod_{z=0}^{1} U(\alpha_j \theta_j,\; \alpha_j(1-\theta_j),\; I_{z,j},\; O_{z,j}) \tag{2.71}$$

Integrating away ψ gives the prior for $z^{(1)}, \cdots, z^{(n)}$:

$$P(z^{(1)}, \cdots, z^{(n)}) \;=\; U(f_1, f_0, n^{[1]}, n^{[0]}) \tag{2.72}$$

From the priors for $z^{(1)}, \cdots, z^{(n)}$, the conditional distribution of $z^{(i)}$ given all other $z^{(j)}$ except $z^{(i)}$, written as $z^{(-i)}$, can be found:

$$P(z^{(i)} \mid z^{(-i)}) = \text{Bernoulli}\left(z^{(i)}; \hat{\psi}^{(-i)}\right) \tag{2.73}$$

where $\hat{\psi}^{(-i)} = \frac{f_1 + n^{[1](-i)}}{f_0 + f_1 + n - 1}$ and $n^{[1](-i)} = \sum_{s=1}^{n} I(z_s = 1, s \neq i)$

We now can write out the conditional distributions needed for performing Gibbs sampling. The conditional distribution of $z^{(i)}$ is proportional to the product of (2.73) and (2.70):

$$P(z^{(i)} \mid \boldsymbol{x}^{\text{train}}, z^{(-i)}, \boldsymbol{\theta}_{0:k}, \alpha_0, \alpha) \propto \text{Bernoulli}\left(z^{(i)}; \hat{\psi}^{(-i)}\right) \prod_{j=0}^{k} \text{Bernoulli}\,(x_j^{(i)}; \hat{\phi}_{z^{(i)},j}^{(-i)}) \tag{2.74}$$

The conditional distribution of θ_j is related to only feature j since the features are independent given $z^{(1)}, \cdots, z^{(n)}$:

$$P(\theta_j \mid \boldsymbol{x}_j^{\text{train}}, \alpha_j, \boldsymbol{z}^{\text{train}}) \;\propto\; \prod_{z=0}^{1} U(\alpha\theta_j,\, \alpha(1-\theta_j),\, I_{z,j},\, O_{z,j}) \tag{2.75}$$

The conditional distribution of α given $\boldsymbol{x}_{0:k}^{\text{train}}$, $\boldsymbol{z}^{\text{train}}$, and $\boldsymbol{\theta}_{0:k}$ is proportional to the products of the factors for $j > 0$ in (2.71) since the prior for α is uniform over $\mathcal{A}$. And the conditional distribution of α_0 is proportional to the factor for $j = 0$ in (2.71).

The prediction described above is, however, invalid if the k features are selected from a large number. It needs to be modified to condition also on the information $\mathcal{S}$, i.e., we should compute $P(x_0^* \mid \boldsymbol{x}_{1:k}^*, \boldsymbol{x}_{0:k}^{\text{train}}, \mathcal{S})$. The calculations are similar to the above, but with $P(\boldsymbol{x}_{0:k}^{\text{train}} \mid \boldsymbol{\theta}_{0:k}, \alpha, \boldsymbol{z}^{\text{train}})$ replaced by $P(\boldsymbol{x}_{0:k}^{\text{train}}, \mathcal{S} \mid \boldsymbol{\theta}_{0:k}, \alpha, \boldsymbol{z}^{\text{train}})$ in (2.66). Accordingly, the conditional distributions of α and $z^{(1)}, \cdots, z^{(n)}$ are multiplied by the following adjustment factor:

$$P(\mathcal{S} \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \alpha) = \left(\int_0^1 P(\, |\text{COR}(x_t^{\text{train}}, x_0^{\text{train}})| \le \gamma \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \theta_t, \alpha) d\theta_t \right)^{p-k} \tag{2.76}$$

Compared with the adjustment factor for the Bayesian naive Bayes models, the adjustment factor (2.76) is more difficult to calculate, as we will discuss in the next section. Furthermore, this adjustment factor depends on both α and the unknown latent label variables $z^{(1)}, \cdots, z^{(n)}$, for which we need to sample using Markov chain sampling method. We therefore need to recompute the adjustment factor whenever we change $z^{(1)}, \cdots, z^{(n)}$ during Markov chain sampling run. But we still need only to calculate the probability of one feature being discarded then raise it to the power of $p - k$.

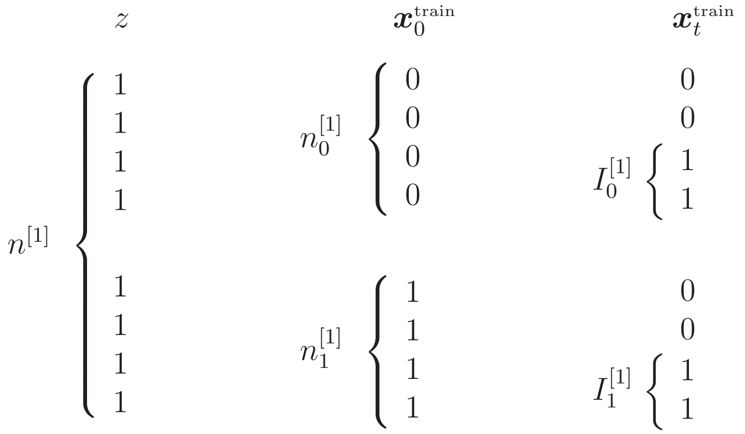


Figure 2.12: Notations used in deriving the adjustment factor of Bayesian mixture models

2.4.3 Computation of the Adjustment Factor for Mixture Models

Computing $P(\mathcal{S} \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \alpha)$ is similar to what was done for Bayesian naive Bayes models in Section 2.3.4, with the difference that we condition on both x_0^{train} and $\boldsymbol{z}^{\text{train}}$. The region $|\text{COR}(x_t^{\text{train}}, x_0^{\text{train}})| \le \gamma$ can still be seen from Figure 2.3. The $P(\mathcal{S} \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \alpha)$ is equal to the sum of $P(I_0, I_1 \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \alpha)$ over $L_+ \cup L_- \cup L_0$, or equivalently 1 minus the sum over $H_+ \cup H_-$. The probability over H_+ is equal to the probability over H_- since the prior for θ_t is symmetrical about 1/2. We therefore need to compute the probability for each point only in either H_+ or H_-. We then exchange the summation over H_+ with the integration with respect to θ_t. Next we discuss only how to calculate $P(I_0, I_1 \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \theta_t, \alpha)$ for each point (I_0, I_1).

We divide the training cases according to $\boldsymbol{z}^{\text{train}}$ into two groups, and let $I_0^{[z]} = \sum_{i=1}^n I(z^{(i)} = z, x_0^{(i)} = 0, x_t^{(i)} = 1)$, and $I_1^{[z]} = I(z^{(i)} = z, x_0^{(i)} = 1, x_t^{(i)} = 1)$, where $z = 0, 1$. The probability of $(I_0^{[z]}, I_1^{[z]})$ is found by summing over all configurations of feature t that have $z^{(i)} = z$ and results in $(I_0^{[z]}, I_1^{[z]})$:

$$\begin{aligned} & P(I_0^{[z]}, I_1^{[z]} \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \theta_t, \alpha) \\ = & \binom{n_0^{[z]}}{I_0^{[z]}} \binom{n_1^{[z]}}{I_1^{[z]}} U(\alpha\theta_t, \alpha(1-\theta_t), I_0^{[z]} + I_1^{[z]}, n^{[z]} - (I_0^{[z]} + I_1^{[z]}) \end{aligned} \tag{2.77}$$

where $n_0^{[z]} = \sum_{i=1}^n I(z^{(i)} = z, x_0^{(i)} = 0), n_1^{[z]} = \sum_{i=1}^n I(z^{(i)} = z, x_0^{(i)} = 1)$ and $n^{[z]} = \sum_{i=1}^n I(z_i = z)$.

Then, the joint probability function of (I_0, I_1) is found by summing over all possible combinations of $(I_0^{[0]}, I_1^{[0]})$ and $(I_0^{[1]}, I_1^{[1]})$ that result in $I_0^{[0]} + I_0^{[1]} = I_0, I_1^{[0]} + I_1^{[1]} = I_1$:

$$P(I_0, I_1 \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \theta_t, \alpha) = \sum_{\substack{I_0^{[0]} + I_0^{[1]} = I_0 \\ I_1^{[0]} + I_1^{[1]} = I_1}} \prod_{z=0}^{1} P(I_0^{[z]}, I_1^{[z]} \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \theta_t, \alpha) \quad (2.78)$$

The way of finding the combinations of $(I_0^{[0]}, I_1^{[0]})$ and $(I_0^{[1]}, I_1^{[1]})$ that satisfy $I_0^{[0]} + I_0^{[1]} = I_0$ and $I_1^{[0]} + I_1^{[1]} = I_1$ is given in the Appendix 2 to this Chapter.

2.4.4 A Simulation Experiment

We tested our method using a data with 200 training cases and 2000 test cases, which are generated from a Bayesian mixture model, by setting $\alpha = 300$, $\phi_{00} = 0.1$, and $\phi_{10} = 0.9$, and letting the number of $z = 1$ and $z = 0$ be equal in both training and test sets.

We then selected four subsets of features, containing 1, 10, 100, and 1000 features, based on the absolute values of the sample correlations of the features with y. The smallest correlation (in absolute value) of a selected feature with the class was 0.30, 0.24, 0.18, and 0.12 for these four subsets. These are the values of γ used by the bias correction method when computing the adjustment factor.

For each number of selected features, we fit this data using the Bayesian mixture model with the prior for ψ (equation (2.56)) having $f_0 = f_1 = 1$ and the Inverse-Gamma prior for both α_0 and α (equation (2.58) and (2.59)) both having shape parameter $a = 0.5$ and rate parameter $b = 5$. After using Gibbs sampling to train the model, with and without correction for the selection bias, we made predictions for the test cases.

We compared the predictive performance of the methods with and without correction for

	1 feature selected out of 10000						**10 features selected out of 10000**					
	Corrected			Uncorrected			Corrected			Uncorrected		
C	#	Pred	Actual	#	Pred	Actual	#	Pred	Actual	#	Pred	Actual
0	0	–	–	0	–	–	0	–	–	19	0.089	0.105
1	0	–	–	0	–	–	0	–	–	340	0.155	0.415
2	0	–	–	0	–	–	19	0.259	0.105	45	0.269	0.622
3	0	–	–	0	–	–	317	0.368	0.420	406	0.359	0.480
4	1083	0.488	0.472	1083	0.424	0.472	530	0.466	0.494	52	0.424	0.558
5	917	0.536	0.523	0	–	–	552	0.549	0.500	54	0.560	0.407
6	0	–	–	917	0.617	0.523	480	0.639	0.529	443	0.649	0.519
7	0	–	–	0	–	–	100	0.735	0.620	49	0.735	0.449
8	0	–	–	0	–	–	2	0.832	1.000	329	0.854	0.505
9	0	–	–	0	–	–	0	–	–	263	0.945	0.593

	100 features selected out of 10000						**1000 features selected out of 10000**					
	Corrected			Uncorrected			Corrected			Uncorrected		
C	#	Pred	Actual	#	Pred	Actual	#	Pred	Actual	#	Pred	Actual
0	71	0.072	0.183	605	0.033	0.286	692	0.033	0.140	919	0.016	0.185
1	195	0.154	0.308	122	0.144	0.361	129	0.148	0.271	33	0.145	0.455
2	237	0.250	0.300	86	0.246	0.465	88	0.247	0.375	28	0.240	0.429
3	229	0.349	0.328	63	0.350	0.508	64	0.350	0.500	21	0.350	0.619
4	234	0.454	0.504	62	0.448	0.597	68	0.454	0.426	20	0.441	0.400
5	259	0.549	0.556	90	0.551	0.589	71	0.548	0.634	25	0.552	0.480
6	253	0.652	0.565	66	0.650	0.530	69	0.646	0.580	20	0.648	0.700
7	251	0.748	0.673	87	0.749	0.529	83	0.744	0.795	31	0.749	0.645
8	192	0.848	0.729	140	0.856	0.564	143	0.857	0.804	44	0.856	0.545
9	79	0.928	0.734	679	0.965	0.666	593	0.966	0.841	859	0.980	0.818

Table 2.3: Comparison of calibration for predictions found with and without correction for selection bias, on data simulated from a binary mixture model. The test cases were divided into 10 categories by the first decimal of the predictive probability of class 1, which is indicated by the 1st column "C". The table shows the number of test cases in each category for each method ("#"), the average predictive probability of class 1 for cases in that category ("Pred"), and the actual fraction of these cases that were in class 1 ("Actual").

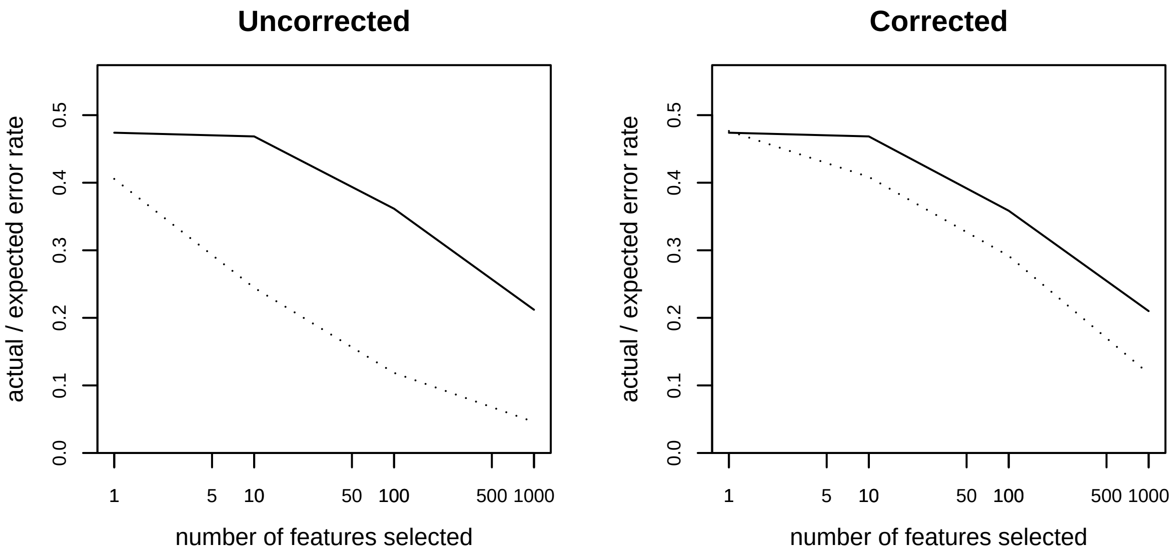


Figure 2.13: Actual and expected error rates with varying numbers (in log scale) of features selected, with and without correction for selection bias. The solid line is the actual error rate on test cases. The dotted line is the error rate that would be expected based on the predictive probabilities.

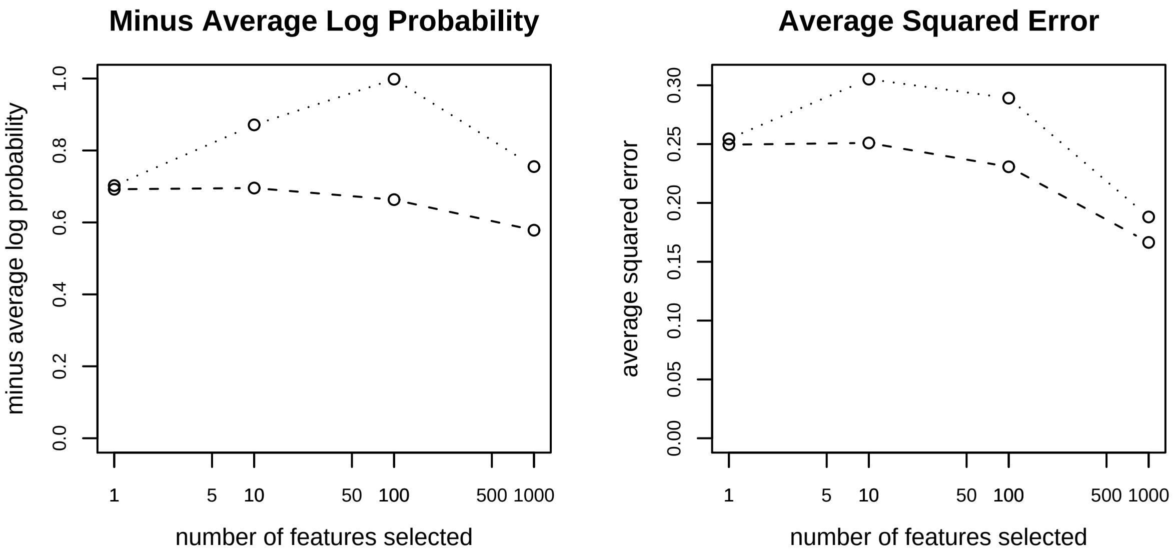


Figure 2.14: Performance in terms of average minus log probability and average squared error, with varying numbers (in log scale) of features selected, with and without correction for selection bias. The left plot shows minus the average log probability of the correct class for test cases, with 1, 10, 100, and 1000 features selected. The dashed line is with bias correction, the dotted line without. The right plot is similar, but shows average squared error on test cases.

selection bias in several ways. Table 2.3 shows how well calibrated the predictive probabilities were, by looking at the actual fraction of class 1 of the test cases with predictive probabilities within each of the ten intervals evenly spaced in $(0,1)$. This shows that the methods with correction for selection bias are better calibrated than without correction. For the methods with correction for selection bias, the actual fractions are closer to predictive probabilities, whereas for the methods without such correction, they are more different, with predictive probabilities incorrectly close to 0 or 1 for many cases. But we have seen that some bias, although less severe, still exists for the methods with correction, which we will explain later.

The calibration can also be illustrated by comparing the actual error rate, from making predictions by thresholding the predictive probabilities at 0.5, to the expected error rate, equal to $(1/N)\sum_i \hat{p}^{(i)} I(\hat{p}^{(i)} < 0.5) + (1-\hat{p}^{(i)}) I(\hat{p}^{(i)} \geq 0.5)$, where $\hat{p}^{(i)}$ is the predictive probability for test case i. As shown by Figure 2.13, the expected error rates for the methods without correction for selection bias are much lower than the actual error rates, showing that the predictions are overconfident. In contrast, the expected error rates and the actual error rates are much closer for the methods with correction for selection bias, though there are still gaps between them. From Figure 2.13, we also see that the actual error rates for the methods with and without correction for selection bias are almost the same.

The remaining bias for the methods with correction for selection bias presumably results from Markov chain Monte Carlo method. Since the two groups are not very apart with $\alpha = 300$, the latent values $z^{(1)}, \cdots, z^{(n)}$ temporarily converge to the responses $x_0^{(1)} \ldots, x_0^{(n)}$, even with correction for selection bias. The estimates of ϕ_{00} and ϕ_{10} are therefore very close to 0 and 1. But the traces of α with correction for selection bias still move around much bigger values in $\mathcal{A}$ than without correction. The probabilities of a test case belonging to two groups with correction are therefore closer than without correction, making it difficult to decide the label of the test case. The correction method, implemented with simple Gibbs sampling, reduces the selection bias, but does not eliminate it entirely. This remaining bias will be eliminated entirely if one uses a more sophisticated Markov chain sampling method that allows the Markov chains to explore more thoroughly in the space of $z^{(1)}, \cdots, z^{(n)}$. Note that, however, this is a matter of computation rather than of theory. In theory, the bias will be eliminated entirely by conditioning on all information available in making Bayesian inference if the data sets are generated from the Bayesian model.

The methods with and without correction for selection bias are also compared in terms of average minus log probability and average squared error, as shown in Figure 2.14. In both measures, the methods with the selection bias corrected are superior over the methods without correction.

The predictive performance using the complete data was not shown in previous table and figures, because it is ironically worse than using selected features. This is because the Markov chain, using the simple Gibbs sampling, converges temporarily to only one group if the initial labels are drawn randomly from the permutations of $1, \ldots, n$. Again, a more sophisticated Markov chain sampling method will solve this problem.

We used Gibbs sampling to train the model. In each iteration of Gibbs sampling, we update α and α_0 once, and repeat 5 time the combination of updating the labels $z^{(1)}, \cdots, z^{(n)}$ once and updating each of $\boldsymbol{\theta}_{1:k}$ 20 times. In approximating the continuous priors for α_0 and α with the uniform distribution over the quantiles of the priors (equation (2.67)), we chose $K = 10$, giving $\mathcal{A} = \mathcal{A}_0 = \{2.60, 4.83, 7.56, 11.45, 17.52, 27.99, 48.57, 98.49, 279.60, 2543.14\}$. Simpson's Rule, which is used to approximate the integral with respect to θ_t for computing the adjustment factor, evaluates the integrand at 11 points.

Our software (available from `http://www.utstat.utoronto.ca/~longhai`) is written entirely in R language. Computation times for each method (on a 2.2 GHz Opteron processor, running 50 iterations of Gibbs sampling as described above) are shown in Table 2.4. The computation of adjustment factor takes a large amount of extra time. This is because the computation of a single adjustment factor is more complex than naive Bayes models and

Number of Features Selected	1	10	100	1000
Uncorrected Method	70	82	193	1277
Corrected Method	3324	2648	2881	4300

Table 2.4: Computation times from simulation experiments with mixture models.

the computation needs to be redone whenever the latent values $z^{(1)}, \cdots, z^{(n)}$ change. The current method for computing the adjustment factor can still be improved. However, the methods using selecting features and correcting for the selection bias still work faster than using complete data, which takes about 40000 seconds for updating 50 iterations.

2.5 Conclusion and Discussion

We have proposed a Bayesian method for making well-calibrated predictions for a response variable when using a subset of features selected from a larger number based on some measure of dependency between the feature and the response. Our method results from applying the basic principle that predictive probabilities should be conditional on all available information — in this case, including the information that some features were discarded because they appear weakly related to the response variable. This information can only be utilized when using a model for the joint distribution of the response and the features, even though we are interested only in the conditional distribution of the response given the features.

We applied this method to naive Bayes models with binary features that are assumed to be independent conditional on the value of the binary response (class) variable. With these models, we can compute the adjustment factor needed to correct for selection bias. Crucially, we need only compute the probability that a single feature will exhibit low correlation with the response, and then raise this probability to the number of discarded features. Due to the simplicity of naive Bayes models, the methods with the selection bias corrected work as fast as the methods without considering this corrrection. Substantial computation time can therefore be saved by discarding features that appear to have little relationship with the response.

We also applied this method to mixture models for binary data. The computation of the adjustment factor is more complex than for naive Bayes models, and it needs to be computed many times. But the method is still feasible, and will be faster than using all features when the number of available features is huge.

The practical utility of the bias correction method we describe would be much improved if methods for more efficiently computing the required adjustment factor could be found, which could be applied to a wide class of models.

Appendix 1:

Proof of the well-calibration of the Bayesian Prediction

Suppose we are interested in predicting whether a random vector $\boldsymbol{Y}$ is in a set $\mathcal{A}$ if *we* know the value of another random vector $\boldsymbol{X}$. Here, $\boldsymbol{X}$ is all the information we know for predicting $\boldsymbol{Y}$, such as the information from the training data and the feature values of a test case. And $\boldsymbol{Y}$ could be any unknown quantity, for example a model parameters or the unknown response of a test case. For discrete $\boldsymbol{Y}$, $\mathcal{A}$ may contain only a single value; for continuous $\boldsymbol{Y}$, it is a set such that the probability of $\boldsymbol{Y} \in \mathcal{A}$ is not 0 (otherwise any predictive method giving predictive probability 0 is well-calibrated). From a Bayesian model for $\boldsymbol{X}$ and $\boldsymbol{Y}$, we can derive a marginal joint distribution for $\boldsymbol{X}$ and $\boldsymbol{Y}$, $P(\boldsymbol{X}, \boldsymbol{Y})$ (which may be a probability function or a density function, or a combination of probability and density function), by integrating over the prior for the model parameters.

Let us denote a series of independent experiments from $P(\boldsymbol{X}, \boldsymbol{Y})$ as $(\boldsymbol{X}_i, \boldsymbol{Y}_i)$, for $i = 1, 2, \ldots$. Suppose a predictive method predicts that event $\boldsymbol{Y} \in \mathcal{A}$ will occur with probability $\hat{Y}(\boldsymbol{x})$ after seeing $\boldsymbol{X} = \boldsymbol{x}$. $\hat{Y}(\boldsymbol{x})$ is said to be well-calibrated if, for any two numbers

$c_1, c_2 \in (0,1)$ (assuming $c_1 < c_2$) such that $P(\,\hat{Y}(\boldsymbol{X}_i) \in (c_1, c_2)\,) \neq 0$, the fraction of $\boldsymbol{Y}_i \in \mathcal{A}$ among those experiments with predictive probability, $\hat{Y}(\boldsymbol{X}_i)$, between c_1 and c_2, will be equal to the average of the predictive proabilities (with P-probability 1), when the number of experiments, k, goes to ∞, that is,

$$\frac{\sum_{i=1}^{k} I(\,\boldsymbol{Y}_i \in \mathcal{A} \text{ and } \hat{Y}(\boldsymbol{X}_i) \in (c_1, c_2)\,)}{\sum_{i=1}^{k} I(\,\hat{Y}(\boldsymbol{X}_i) \in (c_1, c_2)\,)} - \frac{\sum_{i=1}^{k} \hat{Y}(\boldsymbol{X}_i)\, I(\,\hat{Y}(\boldsymbol{X}_i) \in (c_1, c_2)\,)}{\sum_{i=1}^{k} I(\,\hat{Y}(\boldsymbol{X}_i) \in (c_1, c_2)\,)} \longrightarrow 0 \quad (2.79)$$

This definition of well-calibration is a special case for *iid* experiments of what is defined in (Dawid 1982). Note that this concept of calibration is with respect to averaging over both the data and the parameters drawn from the prior.

We will show that under the above definition of calibration, the Bayesian predictive function $\hat{Y}(\boldsymbol{x}) = P(\boldsymbol{Y} \in \mathcal{A} \mid X = \boldsymbol{x})$ is well-calibrated.

First, from the strong law of large numbers, the left-hand of (2.79) converges to:

$$\frac{P(\boldsymbol{Y} \in \mathcal{A} \text{ and } \hat{Y}(\boldsymbol{X}) \in (c_1, c_2))}{P(\hat{Y}(\boldsymbol{X}) \in (c_1, c_2))} - \frac{E(\,\hat{Y}(\boldsymbol{X})\, I(\hat{Y}(\boldsymbol{X}) \in (c_1, c_2)\,)\,)}{P(\hat{Y}(\boldsymbol{X}) \in (c_1, c_2))} \quad (2.80)$$

We then need only show that the expression (2.80) is actually equal to 0, i.e., the numerators in two terms are the same. This equality can be shown as follows:

$$\begin{aligned}
& P(\boldsymbol{Y} \in \mathcal{A} \text{ and } \hat{Y}(\boldsymbol{X}) \in (c_1, c_2)) \\
&= \int I(\,\hat{Y}(\boldsymbol{x}) \in (c_1, c_2))\, P(\boldsymbol{Y} \in \mathcal{A} \mid \boldsymbol{X} = \boldsymbol{x}) P_{\boldsymbol{X}}(\boldsymbol{x})\, d\boldsymbol{x} && (2.81) \\
&= \int I(\,\hat{Y}(\boldsymbol{x}) \in (c_1, c_2))\, \hat{Y}(\boldsymbol{x})\, P_{\boldsymbol{X}}(\boldsymbol{x})\, d\boldsymbol{x} && (2.82) \\
&= E(\,\hat{Y}(\boldsymbol{X})\, I(\hat{Y}(\boldsymbol{X}) \in (c_1, c_2)\,)\,) && (2.83)
\end{aligned}$$

What is essential from (2.81) to (2.82) is that the Bayesian predictive function $\hat{Y}(\boldsymbol{x})$ is just the conditional probability $P(\boldsymbol{Y} \in \mathcal{A} \mid \boldsymbol{X} = \boldsymbol{x})$.

The Bayesian predictive function $\hat{Y}(\boldsymbol{x}) = P(\boldsymbol{Y} \in \mathcal{A} \mid X = \boldsymbol{x})$ also has the following property, which is helpful in understanding the concept of well-calibration:

$$P(\boldsymbol{Y} \in \mathcal{A} \mid \hat{Y}(\boldsymbol{X}) \in (c_1, c_2)) = E(\hat{Y}(\boldsymbol{X}) \mid \hat{Y}(\boldsymbol{X}) \in (c_1, c_2)) \;\; \in \;\; (c_1, c_2) \qquad (2.84)$$

$P(\boldsymbol{Y} \in \mathcal{A} \mid \hat{Y}(\boldsymbol{X}) \in (c_1, c_2))$ is just the first term in (2.80), and is equal to the second term in (2.80), which can be written as $E(\hat{Y}(\boldsymbol{X}) \mid \hat{Y}(\boldsymbol{X}) \in (c_1, c_2))$. This conditional expectation is obviously between c_1 and c_2.

Appendix 2: Details of the Computation of the Adjustment Factor for Binary Mixture Models

Delineating H_+

With the monotonicity of $\mathrm{Cor}(I_1, I_0; x_0^{\text{train}})$ with respect to either I_1 or I_0, we can easily determine the bound of I_1 for each I_0 that satisfies $\mathrm{Cor}(I_1, I_0; x) \leq \gamma$ by solving the equation:

$$|\mathrm{Cor}(I_0, I_1; x_0^{\text{train}})| \; = \gamma \qquad (2.85)$$

then rounding to the appropriate integers and truncating them by 0 and n_1. I.e. the lower bound is $\max(0, \lceil l \rceil)$ and the upper bound is $\min(n_1, \lfloor u \rfloor)$, where l and u are the two solutions of equation (2.85). When $I_0 = 0$, if I_1 is also 0, we assume the correlation of x_0^{train} and x_t^{train} is 0, therefore the lower bound is set to be 0. Similarly, when $I_0 = n_0$, the upper bound is set to be n_1.

Computation of $P(I_0^{[z]}, I_1^{[z]} \mid x_0^{\mathbf{train}}, \boldsymbol{z}^{\mathbf{train}}, \theta_t, \alpha)$ with formula (2.77)

We need to calculate $P(I_0^{[z]}, I_1^{[z]} \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \theta_t, \alpha)$ for all $I_0^{[z]} \in \{0, \cdots, n_0^{[z]}\}$ and $I_1^{[z]} \in \{0, \cdots, n_1^{[z]}\}$. These values are saved with a matrix $T^{[z]}$ for convenience. We don't have to evaluate each element of $T^{[z]}$ by noting that the third factor of equation (2.78) depends only on $I_0^{[z]} + I_1^{[z]}$, i.e. the number of $x_t^{\text{train}} = 1$. For each $k \in \{0, 1, \cdots, n_0^{[z]} + n_1^{[z]}\}$, we need to evaluate it only once. Then go along the diagonal line $I_0^{[z]} + I_1^{[z]} = k$ to obtain the elements of $T^{[z]}$. The lower bound of $I_1^{[z]}$ on this line is $\max(0, k - n_0^{[z]})$ and the upper bound is $\min(k, n_1^{[z]})$. For each $I_1^{[z]}$ between the lower and upper bounds, correspondingly $I_0^{[z]} = k - I_1^{[z]}$.

For each line associated with k, only when $I_1^{[z]} = \max(0, k - n_0^{[z]})$ we need to evaluate (2.78), then we can obtain the remaining values using the following relation between two successive elements on the line:

$$\frac{P(I_0^{[z]}, I_1^{[z]} \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \alpha, \theta_t)}{P(I_0^{[z]}+1, I_1^{[z]}-1 \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \alpha, \theta_t)} = \frac{\binom{n_0^{[z]}}{I_0^{[z]}}\binom{n_1^{[z]}}{I_1^{[z]}}}{\binom{n_0^{[z]}}{I_0^{[z]}+1}\binom{n_1^{[z]}}{I_1^{[z]}-1}} \tag{2.86}$$

$$= \frac{(I_0^{[z]}+1)(n_1^{[z]} - I_1^{[z]} + 1)}{(n_0^{[z]} - I_0^{[z]})I_1^{[z]}} \tag{2.87}$$

Computation of $P(I_0, I_1 \mid x_0^{\mathbf{train}}, \boldsymbol{z}^{\mathbf{train}}, \theta_t, \alpha)$ with formula (2.78)

For each $(I_0, I_1) \in H_+$, we need find all the pairs of $(I_0^{[0]}, I_1^{[0]})$ and $(I_0^{[1]}, I_1^{[1]})$ that satisfy

$$I_0^{[0]} + I_0^{[1]} = I_0, \text{and } 0 \le I_0^{[0]} \le n_0^{[0]}, 0 \le I_0^{[1]} \le n_0^{[1]} \tag{2.88}$$

$$I_1^{[0]} + I_1^{[1]} = I_1, \text{and } 0 \le I_1^{[0]} \le n_1^{[0]}, 0 \le I_1^{[1]} \le n_1^{[1]} \tag{2.89}$$

The decompositions of I_0 and I_1 are independent and the methods are identical. Taking I_0 as example, we determine the bound of $I_0^{[0]}$ and $I_0^{[1]}$ by truncating the straight line of

$I_0^{[0]} + I_0^{[1]} = I_0$ with the square determined by $(0, n_0^{[0]}) \times (0, n_0^{[1]})$. By this way, we obtain the decompositions as follows:

$$I_0^{[0]} \in \{\max(0, I_0 - n_0^{[1]}), \cdots, \min(n_0^{[0]}, I_0)\} \equiv B_{01}^{[0]}, \text{ and } I_0^{[1]} = I_0 - I_0^{[0]} \tag{2.90}$$

$$I_1^{[0]} \in \{\max(0, I_1 - n_1^{[1]}), \cdots, \min(n_1^{[0]}, I_1)\} \equiv B_{11}^{[0]}, \text{ and } I_1^{[1]} = I_1 - I_1^{[0]} \tag{2.91}$$

We make a sub-matrix $S^{[0]}$ from $T^{[0]}$, which has been computed in Section 2.5, by taking the rows in $B_{01}^{[0]}$ and the columns in $B_{11}^{[0]}$, and accordingly make $S^{[1]}$ from $T^{[1]}$ by taking the rows in $I_0 - B_{01}^{[0]}$ and the columns in $I_1 - B_{11}^{[0]}$. Then multiplying $S^{[0]}$ and $S^{[1]}$ element by element (i.e. the corresponding elements of $S^{[0]}$ and $S^{[1]}$ are multiplied together) makes matrix S. Summing all the elements of S together yields $P(I_0, I_1 \mid x_0^{\text{train}}, \boldsymbol{z}^{\text{train}}, \theta_t, \alpha)$.

Chapter 3

Compressing Parameters in Bayesian Models with High-orderInteractions

Abstract. Bayesian regression and classification with high order interactions is largely infeasible because Markov chain Monte Carlo (MCMC) would need to be applied with a huge number of parameters, which typically increases exponentially with the order. In this chapter we show how to make it feasible by effectively reducing the number of parameters, exploiting the fact that many interactions have the same values for all training cases. Our method uses a single "compressed" parameter to represent the sum of all parameters associated with a set of patterns that have the same value for all training cases. Using symmetric stable distributions as the priors of the original parameters, we can easily find the priors of these compressed parameters. We therefore need to deal only with a much smaller number of compressed parameters when training the model with MCMC. The number of compressed parameters may have converged before considering the highest possible order. After training the model, we can split these compressed parameters into the original ones as needed to make predictions for test cases. We show in detail how to compress parameters for logistic sequence prediction and logistic classification models. Experiments on both simulated and real data demonstrate that a huge number of parameters can indeed be reduced by our compression method.

3.1 Introduction

In many regression and classification problems, the response variable y depends on high-order interactions of “features” (also called “covariates”, “inputs”, “predictor variables”, or “explanatory variables”). Some complex human diseases are found to be related to high-order interactions of susceptibility genes and environmental exposures (Ritchie et. al. 2001). The prediction of the next character in English text is improved by using a large number of preceding characters (Bell, Cleary and Witten 1990). Many biological sequences have long-memory properties.

When the features are discrete, we can employ high-order interactions in regression and classification models by introducing, as additional predictor variables, the indicators for each possible interaction pattern, equal to 1 if the pattern occurs for a subject and 0 otherwise. In this chapter we will use “features” for the original discrete measurements and “predictor variables” for these derived variables, to distinguish them. The number of such predictor variables increases exponentially with the order of interactions. The total number of order-k interaction patterns with k binary (0/1) features is 2^k, accordingly we will have 2^k predictor variables. A model with interactions of even a moderate order is prohibitive in real applications, primarily for computational reasons. People are often forced to use a model with very small order, say only 1 or 2, which, however, may omit useful high-order predictor variables.

Besides the computational considerations, regression and classification with a great many predictor variables may “overfit” the data. Unless the number of training cases is much larger than the number of predictor variables the model may fit the noise instead of the signal in the data, with the result that predictions for new test cases are poor. This problem can be solved by using Bayesian modeling with appropriate prior distributions. In a Bayesian model, we use a probability distribution over parameters to express our prior belief about which configurations of parameters may be appropriate. One such prior belief is that a parsimonious model can approximate the reality well. In particular, we may believe that most high-order interactions are largely irrelevant to predicting the response. We express such a prior by assigning each regression coefficient a distribution with mode 0, such as a Gaussian or Cauchy distribution centered at 0. Due to its heavy tail, a Cauchy distribution may be more appropriate than a Gaussian distribution to express the prior belief that almost all coefficients of high order interactions are close to 0, with a very small number of exceptions. Additionally, the priors we use for the widths of Gaussian or Cauchy distributions for higher order interaction should favor small values. The resulting joint prior for all coefficients favors a model with most coefficients close to 0, that is, a model emphasizing low order interactions. By incorporating such prior information into our inference, we will not overfit the data with an unnecessarily complex model.

However, the computational difficulty with a huge number of parameters is even more pronounced for a Bayesian approach than other approaches, if we have to use Markov chain Monte Carlo methods to sample from the posterior distribution, which is computationally burdensome even for a moderate number of parameters. With more parameters, a Markov chain sampler will take longer for each iteration and require more memory, and may need more iterations to converge or get trapped more easily in local modes. Applying Markov chain Monte Carlo methods to regression and classification with high-order interactions therefore seems infeasible.

In this chapter, we show how these problems can be solved by effectively reducing the number of parameters in a Bayesian model with high-order interactions, using the fact that in a model that uses all interaction patterns, from a low order to a high order, many predictor variables have the same values for all training cases. For example, if an interaction pattern occurs in only one training case, all the interaction patterns of higher order contained in it will also occur in only that case and have the same values for all training cases — 1 for that training case and 0 for all others. Consequently, only the sum of the coefficients associated with these predictor variables matters in the likelihood function. We can therefore use a single "compressed" parameter to represent the sum of the regression coefficients for a group of predictor variables that have the same values in training cases. For models with very high order of interactions, the number of such compressed parameters will be much smaller than the number of original parameters. If the priors for the original parameters are symmetric stable distributions, such as Gaussian or Cauchy, we can easily find the prior distributions of these compressed parameters, as they are also symmetric stable distributions of the same type. In training the model with Markov chain Monte Carlo methods we need to deal only with these compressed parameters. After training the model, the compressed parameters can be split into the original ones as needed to make predictions for test cases. Using our method for compressing parameters, one can handle Bayesian regression and classification problems with very high order of interactions in a reasonable amount of time.

This chapter will be organized as follows. We first describe Bayesian logistic sequence prediction models and Bayesian logistic classification models to which our compression method can be applied. Then, in Section 3.3 we describe in general terms the method of compressing parameters, and how to split them to make predictions for test cases. We then apply the method to logistic sequence models in Section 3.4, and to logistic classification models in Section 3.5. There, we will describe the specific schemes for compressing parameters for these models, and use simulated data and real data to demonstrate our method. We draw conclusions and discuss future work in Section 3.6.

3.2 Two Models with High-order Interactions

3.2.1 Bayesian Logistic Sequence Prediction Models

We often need to predict the next state of a sequence given its preceding states, for example in speech recognition (Jelinek 1998), in text compression (Bell, Cleary, and Witten 1990), and in many others. We write a sequence of length $O+1$ as $x_1, \ldots, x_O, x_{O+1}$, where x_t

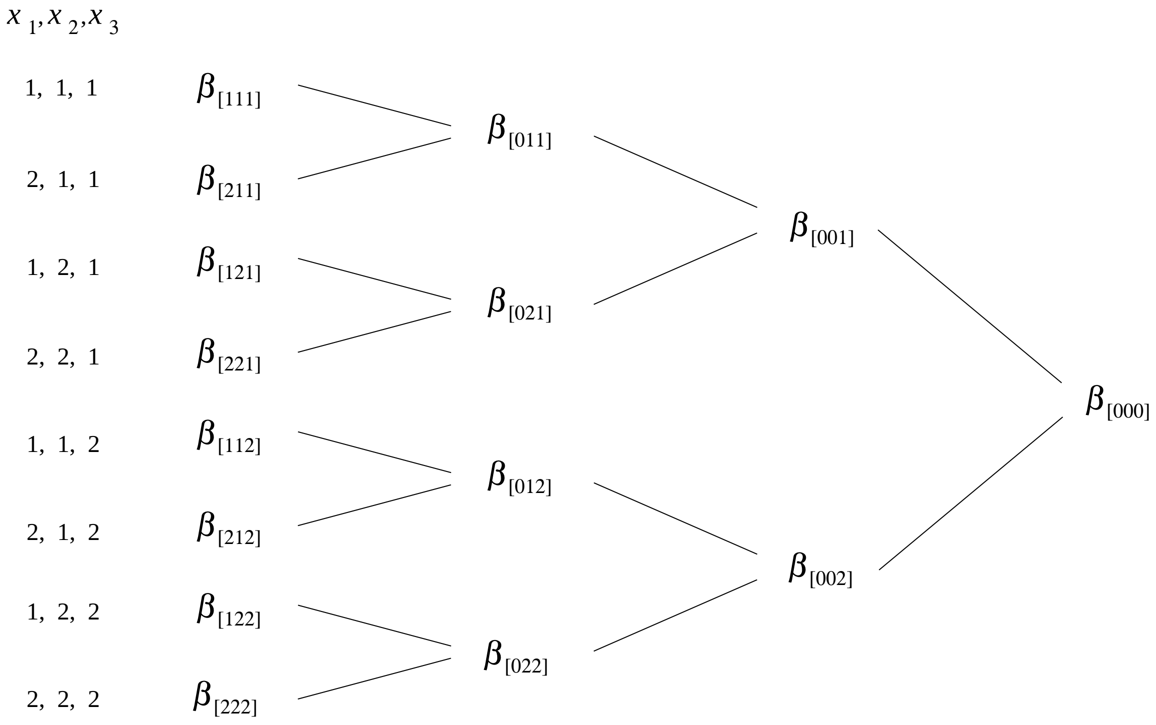


Figure 3.1: A picture of the coefficients, $\boldsymbol{\beta}$, for all patterns in binary sequences of length $O = 3$. $\beta_{[A_1 A_2 A_3]}$ is associated with the pattern written as $[A_1 A_2 A_3]$, with $A_t = 0$ meaning that x_t is allowed to be either 1 or 2, in other words, x_t is ignored in defining this pattern. For example, $\beta_{[000]}$ is the intercept term. These coefficients are used in defining the linear function $l\left((x_1, x_2, x_3), \boldsymbol{\beta}\right)$ in the logistic model (3.1). For each combination of (x_1, x_2, x_3) on the left column, $l\left((x_1, x_2, x_3), \boldsymbol{\beta}\right)$ is equal to the sum of β's along the path linked by lines, from $\beta_{[x_1 x_2 x_3]}$ to $\beta_{[000]}$.

takes values from 1 to K_t, for $t = 1, \ldots, O$, and x_{O+1} takes values from 1 to K. We call $x_1, \ldots, x_O = \boldsymbol{x}_{1:O}$ the historic sequence. For subject i we write its historic sequence and response as $\boldsymbol{x}_{1:O}^{(i)}$ and $x_{O+1}^{(i)}$. We are interested in modelling the conditional distribution $P(x_{O+1} \mid \boldsymbol{x}_{1:O})$.

An interaction pattern $\mathcal{P}$ is written as $[A_1 A_2 \ldots A_O]$, where A_t can be from 0 to K_t, with $A_t = 0$ meaning that x_t can be any value from 1 to K_t. For example, $[0 \ldots 01]$ denotes the pattern that fixes $x_O = 1$ and allows $x_1, \ldots, x_{O-1}$ to be any values in their ranges. When all nonzero elements of $\mathcal{P}$ are equal to the corresponding elements of a historic sequence,

$\boldsymbol{x}_{1:O}$, we say that pattern $\mathcal{P}$ occurs in $\boldsymbol{x}_{1:O}$, or pattern $\mathcal{P}$ is expressed by $\boldsymbol{x}_{1:O}$, denoted by $\boldsymbol{x}_{1:O} \in \mathcal{P}$. We will use the indicator $I(x_{1:O} \in \mathcal{P})$ as a predictor variable, whose coefficient is denoted by $\beta_{\mathcal{P}}$. For example, $\beta_{[0\cdots 0]}$ is the intercept term. A logistic model assigns each possible value of the response a linear function of the predictor variables. We use $\beta_{\mathcal{P}}^{(k)}$ to denote the coefficient associated with pattern $\mathcal{P}$ and used in the linear function for $x_{O+1} = k$.

For modeling sequences, we consider only the patterns where all zeros (if any) are at the start. Let us denote all such patterns by $\boldsymbol{\mathcal{S}}$. We write all coefficients for $x_{O+1} = k$, i.e., $\left\{\beta_{\mathcal{P}}^{(k)} \mid \mathcal{P} \in \boldsymbol{\mathcal{S}}\right\}$, collectively as $\boldsymbol{\beta}^{(k)}$. Figure (3.1) displays $\boldsymbol{\beta}^{(k)}$ for binary sequence of length $O = 3$, for some k, placed in a tree-shape.

Conditional on $\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(K)}$ and $\boldsymbol{x}_{1:O}$, the distribution of x_{O+1} is defined as

$$P(x_{O+1} = k \mid \boldsymbol{x}_{1:O}, \boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(K)}) = \frac{\exp(l\,(\boldsymbol{x}_{1:O}, \boldsymbol{\beta}^{(k)}))}{\sum_{j=1}^{K} \exp(l\,(\boldsymbol{x}_{1:O}, \boldsymbol{\beta}^{(j)}))} \tag{3.1}$$

where

$$l\,(\boldsymbol{x}_{1:O}, \boldsymbol{\beta}^{(k)}) \;=\; \sum_{\mathcal{P} \in \boldsymbol{\mathcal{S}}} \beta_{\mathcal{P}}^{(k)}\; I(\boldsymbol{x}_{1:O} \in \mathcal{P}) = \beta_{[0\cdots 0]}^{(k)} + \sum_{t=1}^{O} \beta_{[0\cdots x_t \cdots x_O]}^{(k)} \tag{3.2}$$

In Figure 3.1, we display the linear functions for each possible combination of (x_1, x_2, x_3) on the left column, by linking together all β's in the summation (3.2) with lines, from $\beta_{[x_1 x_2 x_3]}$ to $\beta_{[000]}$.

The prior for each $\beta_{\mathcal{P}}^{(k)}$ is a Gaussian or Cauchy distribution centered at 0, whose width depends on the order, $o(\mathcal{P})$, of $\mathcal{P}$, which is the number of nonzero elements of $\mathcal{P}$. There are $O+1$ such width parameters, denoted by $\sigma_0, \ldots, \sigma_O$. The σ_o's are treated as hyperparameters, assigned Inverse Gamma prior distributions with some shape and rate parameters, leaving their values to be determined by the data. In summary, the hierarchy of the priors is:

$$\begin{aligned} \sigma_o &\sim \text{Inverse-Gamma}(\alpha_o\,, (\alpha_o + 1)\, w_o), \ \text{ for } o = 0, \ldots, O \\ \beta^{(k)}_{\mathcal{P}} \mid \sigma_{o(\mathcal{P})} &\sim \text{Cauchy}(0, \sigma_{o(\mathcal{P})}) \text{ or } N(0, \sigma^2_{o(\mathcal{P})}), \ \text{ for } \mathcal{P} \in \boldsymbol{\mathcal{S}} \end{aligned} \tag{3.3}$$

where Inverse-Gamma(α, λ) denotes an Inverse Gamma distribution with density function $x^{-\alpha-1}\, \lambda^\alpha\, \exp(-\lambda/x)/\Gamma(\alpha)$. We express α and λ in (3.3) so that the mode of the prior is w_o.

3.2.2 Remarks on the Sequence Prediction Models

The Inverse Gamma distributions have heavy upward tails when α is small, and particularly when $\alpha \leq 1$, they have infinite means. An Inverse Gamma distribution with $\alpha_o \leq 1$ and small w_o, favors small values around w_o, but still allows σ_o to be exceptionally large, as needed by the data. Similarly, the Cauchy distributions have heavy two-sided tails. The absolute value of a Cauchy random variable has infinite mean. When a Cauchy distribution with center 0 and a small width is used as the prior for a group of parameters, such as all β's of the interaction patterns with the same order in (3.3), a few parameters may be much larger in absolute value than others in this group. As the priors for the coefficients of high-order interaction patterns, the Cauchy distributions can therefore express more accurately than the Gaussian distributions the prior belief that most high-order interaction patterns are useless in predicting the response, but a small number may be important.

It seems redundant to use a $\boldsymbol{\beta}^{(k)}$ for each $k = 1, \ldots, K$ in (3.1) since only the differences between $\boldsymbol{\beta}^{(k)}$ matter in (3.1). A non-Bayesian model could fix one of them, say $\boldsymbol{\beta}^{(1)}$, all equal to 0, so as to make the parameters identifiable. However, when $K \neq 2$, forcing $\boldsymbol{\beta}^{(1)} = 0$ in a Bayesian model will result in a prior that is not symmetric for all k, which we may not be able to justify. When $K = 2$, we do require that $\boldsymbol{\beta}^{(1)}$ are all equal to 0, as there is no asymmetry problem.

Inclusion of $\beta_{\mathcal{P}}$ other than the highest order is also a redundancy, which facilitates the expression of appropriate prior beliefs. The prior distributions of linear functions of similar historic sequences $x_{1:O}$ are positively correlated since they share some common β's, for example, in the model displayed by Figure 3.1, $l\left((1,1,1),\boldsymbol{\beta}\right)$ and $l\left((2,1,1),\boldsymbol{\beta}\right)$ share $\beta_{[011]}, \beta_{[001]}$ and $\beta_{[000]}$. Consequently, the predictive distributions of x_O are similar given similar $x_{1:O}$. By incorporating such a prior belief into our inference, we borrow "statistical strength" for those historic sequences with few replications in the training cases from other similar sequences with more replications, avoiding making an unreasonably extreme conclusion due to a small number of replications.

3.2.3 Bayesian Logistic Classification Models

In this section we define the general Bayesian classification models with high-order interactions. We write the feature vector of dimension p as $(x_1, \ldots, x_p)$, or collectively $\boldsymbol{x}_{1:p}$, where x_t takes values from 1 to K_t, for $t = 1, \ldots, p$. In this thesis, we consider only classification problems in which the response y is discrete, assumed to take values from 1 to K. But our compression method could be applied to regression problems without any difficulty. The features and response for subject i are written as $\boldsymbol{x}_{1:p}^{(i)}$ and $y^{(i)}$. We are interested in modeling the conditional distribution $P(y \mid \boldsymbol{x}_{1:p})$.

A pattern $\mathcal{P}$ is written as $[A_1 A_2 \ldots A_p]$, where A_t can be from 0 to K_t, with $A_t = 0$ meaning that x_t can be any value from 1 to K_t. For example, $[0 \ldots 01]$ denotes the pattern that fixes $x_p = 1$ and allows $x_1, \ldots, x_{p-1}$ to be any values in their ranges. When all nonzero elements of $\mathcal{P}$ are equal to the corresponding elements of a feature vector, $\boldsymbol{x}_{1:p}$, we say that pattern $\mathcal{P}$ occurs in $\boldsymbol{x}_{1:p}$, or pattern $\mathcal{P}$ is expressed by $\boldsymbol{x}_{1:p}$, denoted by $\boldsymbol{x}_{1:p} \in \mathcal{P}$. The number of nonzero elements of a pattern $\mathcal{P}$ is called the order of $\mathcal{P}$, denoted by $o(\mathcal{P})$. All the patterns of order o are denoted by $\boldsymbol{\mathcal{P}}^o$, and all the patterns from order 0 to order O are denoted by $\boldsymbol{\mathcal{P}}^{0:O} = \bigcup_{o=0}^{O} \mathcal{P}^o$. All the patterns that are of order o and expressed by a feature vector $\boldsymbol{x}_{1:p}$, are denoted by $\boldsymbol{\mathcal{P}}^o_{\boldsymbol{x}_{1:p}} = \{\,[A_1, \ldots, A_p] \mid A_t = 0 \text{ or } x_t \text{ and } \sum_{t=1}^{p} I(A_t \neq 0) = o\}$.

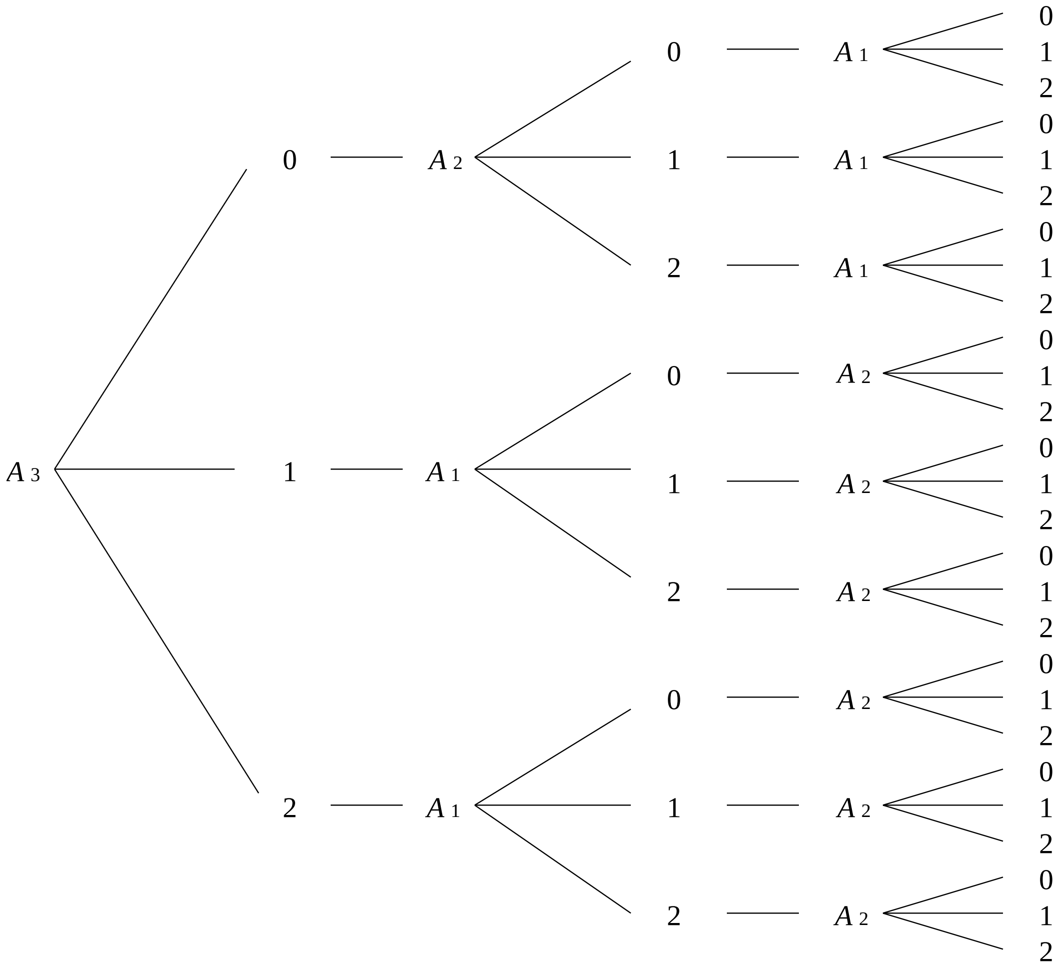


Figure 3.2: A picture displaying all the interaction patterns, from order 0 to order 3, of 3 features x_1, x_2, x_3, where x_t is either 1 or 2. A pattern, written as $[A_1 A_2 A_3]$, is shown by 3 numbers linked by lines, where A_t can be an integer from 0 to 2, with $A_t = 0$ meaning x_t could be either 1 or 2. The order of A_1, A_2, A_3 on the above graph can be changed to any permutation of A_1, A_2, A_3.

There are totally $\left(\begin{array}{c} p \\ o \end{array}\right)$ patterns in $\mathcal{P}^o_{\boldsymbol{x}_{1:p}}$. For example, $\boldsymbol{\mathcal{P}}^2_{(1,2,1)} = \{[0,2,1],\ [1,0,1],\ [1,2,0]\}$. Figure 3.2 displays $\boldsymbol{\mathcal{P}}^{0:3}$ for 3 binary (1 / 2) features x_1, x_2, x_3. The patterns expressed by a feature vector (x_1, x_2, x_3) can be found from such a graph, by searching from the root along the lines pointing to $A_t = 0$ and $A_t = x_t$.

We will use the indicator $I(x_{1:p} \in \mathcal{P})$ as a predictor variable, with coefficient denoted by $\beta_{\mathcal{P}}$. For example, $\beta_{[0\cdots 0]}$ is the intercept term. A logistic model assigns each possible value of the response a linear function of the predictor variables. We use $\beta^{(k)}_{\mathcal{P}}$ to denote the coefficient associated with pattern $\mathcal{P}$ and used in the linear function for $y = k$. All the coefficients for $y = k$ are written as $\boldsymbol{\beta}^{(k)} = \{\beta^{(k)}_{\mathcal{P}} \mid \mathcal{P} \in \boldsymbol{\mathcal{P}}^{0:O}\}$.

A Bayesian logistic classification model using all interaction patterns from order 0 to order O is defined as follows:

$$P(y = k \mid \boldsymbol{x}_{1:p}, \boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(K)}) = \frac{\exp(l\,(\boldsymbol{x}_{1:p}, \boldsymbol{\beta}^{(k)}))}{\sum_{j=1}^{K} \exp(l\,(\boldsymbol{x}_{1:p}, \boldsymbol{\beta}^{(j)}))} \tag{3.4}$$

where

$$l\,(\boldsymbol{x}_{1:p}, \boldsymbol{\beta}^{(k)}) = \sum_{\mathcal{P} \in \boldsymbol{\mathcal{P}}^{0:O}} \beta^{(k)}_{\mathcal{P}}\, I(\boldsymbol{x}_{1:p} \in \mathcal{P}) = \sum_{o=0}^{O} \sum_{\mathcal{P} \in \boldsymbol{\mathcal{P}}^o_{\boldsymbol{x}_{1:p}}} \beta^{(k)}_{\mathcal{P}} \tag{3.5}$$

The priors for $\beta^{(k)}_{\mathcal{P}}$ are given in the same way as in (3.3).

The remarks regarding the Bayesian sequence prediction models in Section 3.2.2 still apply to the above classification models. Compared with the classification models, the Bayesian sequence prediction models are more restrictive models, using only the interaction patterns with all zeros at the start as predictor variables.

3.3 Our Method for Compressing Parameters

In this section we describe in general terms our method for compressing parameters in Bayesian models, and how the original parameters can later be retrieved as needed for use in making predictions for test cases.

3.3.1 Compressing Parameters

In the above high-order models, the regression parameters of the likelihood function can be divided into a number of groups such that the likelihood function depends only on the sums over these groups, as shown by equation (3.8) below. We first use the Bayesian logistic sequence models to illustrate this fact. The likelihood function of $\boldsymbol{\beta}^{(k)}$, for $k = 1, \ldots, K$, is the product of probabilities in (3.1) applied to the training cases, $\boldsymbol{x}_{1:O}^{(i)}, x_{O+1}^{(i)}$, for $i = 1, \ldots, N$ (collectively denoted by $\mathcal{D}$). It can be written as follows:

$$L^{\beta}(\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(K)} \mid \mathcal{D}) = \prod_{i=1}^{N} \frac{\exp(l\,(\boldsymbol{x}_{1:O}^{(i)}, \boldsymbol{\beta}^{(x_{O+1}^{(i)})}))}{\sum_{j=1}^{K} \exp(l\,(\boldsymbol{x}_{1:O}^{(i)}, \boldsymbol{\beta}^{(j)}))} \tag{3.6}$$

(When $K = 2$, $\boldsymbol{\beta}^{(1)}$ is fixed at 0, and therefore not included in the above likelihood function. But for simplicity, we do not write another expression for $K = 2$.)

As can be seen in (3.2), the function $l\,(\boldsymbol{x}_{1:O}, \boldsymbol{\beta})$ is the sum of the β's associated with the interaction patterns expressed by $\boldsymbol{x}_{1:O}$. If a group of interaction patterns are expressed by the same training cases, the associated β's will appear *simultaneously* in the same factors of (3.6). The likelihood function (3.6) therefore depends only on the sum of these β's, rather than the individual ones. Suppose the number of such groups is G. The parameters in group g are rewritten as $\beta_{g1}, \ldots, \beta_{g,n_g}$, and the sum of them is denoted by s_g:

$$s_g = \sum_{k=1}^{n_g} \beta_{gk}, \qquad \text{for } g = 1, \ldots, G \tag{3.7}$$

The likelihood function can then be rewritten as:

$$\begin{aligned} & L^\beta(\beta_{11},\ldots,\beta_{1,n_1},\ \ldots\ ,\beta_{G1},\ldots,\beta_{G,n_G}) \\ = & \ L\left(\sum_{k=1}^{n_1}\beta_{1k},\ \ldots,\ \sum_{k=1}^{n_G}\beta_{Gk}\right) = L(s_1,\ \ldots\ ,s_G) \end{aligned} \tag{3.8}$$

(The above β's are only the regression coefficients for the interaction patterns occurring in training cases. The predictive distribution for a test case may use extra regression coefficients, whose distributions depend only on the priors given relevant hyperparameters.)

We need to define priors for the β_{gk} in a way that lets us easily find the priors of the s_g. For this purpose, we could assign each β_{gk} a symmetric stable distribution centered at 0 with width parameter σ_{gk}. Symmetric stable distributions (Feller 1966) have the following additive property: If random variables $X_1,\ldots,X_n$ are independent and have symmetric stable distributions of index α, with location parameters 0 and width parameters $\sigma_1,\ldots,\sigma_n$, then the sum of these random variables, $\sum_{i=1}^n X_i$, also has a symmetric stable distribution of index α, with location parameter 0 and width parameter $(\sum_{i=1}^n \sigma_i^\alpha)^{1/\alpha}$. Symmetric stable distributions exist and are unique for $\alpha \in (0,2]$. The symmetric stable distributions with $\alpha = 1$ are Cauchy distributions. The density function of a Cauchy distribution with location parameter 0 and width parameter σ is $[\pi\sigma(1+x^2/\sigma^2)]^{-1}$. The symmetric stable distributions with $\alpha = 2$ are Gaussian distributions, for which the width parameter is the standard deviation. Since the symmetric stable distributions with α other than 1 or 2 do not have closed form density functions, we will use only Gaussian or Cauchy priors. That is, each parameter β_{gk} has a Gaussian or Cauchy distribution with location parameter 0 and width parameter σ_{gk}:

$$\beta_{gk} \sim N(0,\sigma_{gk}^2) \quad \text{or} \quad \beta_{gk} \sim \text{Cauchy}(0,\sigma_{gk}) \tag{3.9}$$

As can been seen from the definitions of the priors (equation (3.3)), some σ_{gk} may be common

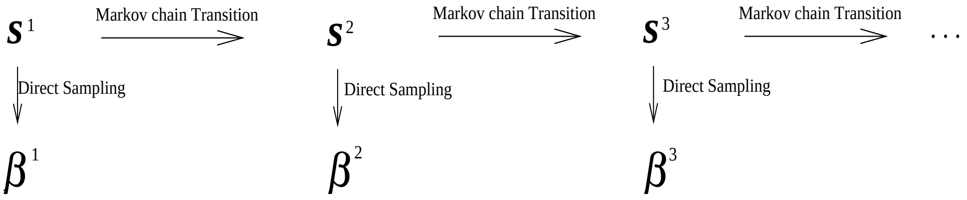


Figure 3.3: A picture depicting the sampling procedure after compressing parameters.

for different β_{gk}, but for simplicity we denote them individually. We might also treat the σ_{gk}'s as unknown hyperparameters, but again we assume them fixed for the moment.

If the prior distributions for the β_{gk}'s are as in (3.9), the prior distribution of s_g can be found using the property of symmetric stable distributions:

$$s_g \sim N\left(0,\ \sum_{k=1}^{n_g} \sigma_{gk}^2\right) \quad \text{or} \quad s_g \sim \text{Cauchy}\left(0,\ \sum_{k=1}^{n_g} \sigma_{gk}\right) \tag{3.10}$$

Let us denote the density of s_g in (3.10) by P_g^s (either a Gaussian or Cauchy), and denote $s_1, \ldots, s_G$ collectively by $\boldsymbol{s}$. The posterior distribution can be written as follows:

$$P(\boldsymbol{s} \mid \mathcal{D}) = \frac{1}{c(\mathcal{D})}\, L(s_1,\ \ldots\ , s_G)\ P_1^s(s_1)\ \cdots\ P_g^s(s_G) \tag{3.11}$$

where $\mathcal{D}$ is the training data, and $c(\mathcal{D})$ is the marginal probability or density function of $\mathcal{D}$.

Since the likelihood function $L(s_1,\ \ldots\ , s_G)$ typically depends on $s_1, \ldots, s_G$ in a complicated way, we may have to use some Markov chain sampling method to sample for $\boldsymbol{s}$ from distribution (3.11).

3.3.2 Splitting Compressed Parameters

After we have obtained samples of s_g, we may need to split them into their original components $\beta_{g1}, \ldots, \beta_{g,n_g}$ to make predictions for test cases. This "splitting" distribution depends only on the prior distributions, and is independent of the training data $\mathcal{D}$. In other words, the splitting distribution is just the conditional distribution of $\beta_{g1}, \ldots, \beta_{gn_g}$ given $\sum_{k=1}^{n_g} \beta_{gk} = s_g$, whose density function is:

$$P(\beta_{g1}, \ldots, \beta_{g,n_g-1} \mid s_g) = \left[\prod_{k=1}^{n_g-1} P_{gk}(\beta_{gk})\right] P_{g,n_g}\left(s_g - \sum_{k=1}^{n_g-1} \beta_{gk}\right) / P_g^s(s_g) \tag{3.12}$$

where P_{gk} is the density function of the prior for β_{gk}. Note that β_{g,n_g} is omitted since it is equal to $s_g - \sum_{k=1}^{n_g-1} \beta_{gk}$.

As will be discussed in the Section 3.3.4, sampling from (3.12) can be done efficiently by a direct sampling method, which does not involve costly evaluations of the likelihood function. We need to use Markov chain sampling methods and evaluate the likelihood function only when sampling for $\boldsymbol{s}$. Figure 3.3 shows the sampling procedure after compressing parameters, where $\boldsymbol{\beta}$ is a collective representation of β_{gk}, for $g = 1, \ldots, G, k = 1, \ldots, n_g - 1$. When we consider high-order interactions, the number of groups, G, will be much smaller than the number of β_{gk}'s. This procedure is therefore much more efficient than applying Markov chain sampling methods to all the original β_{gk} parameters.

Furthermore, when making predictions for a particular test case, we actually do not need to sample from the distribution (3.12), of dimension $n_g - 1$, but only from a derived 1-dimensional distribution, which saves a huge amount of space.

Before discussing how to sample from (3.12), we first phrase this compressing-splitting procedure more formally in the next section to show its correctness.

3.3.3 Correctness of the Compressing-Splitting Procedure

The above procedure of compressing and splitting parameters can be seen in terms of a transformation of the original parameters β_{gk} to a new set of parameters containing s_g's, as defined in (3.7), in light of the training data. The posterior distribution (3.11) of $\boldsymbol{s}$ and the splitting distribution (3.12) can be derived from the joint posterior distribution of the new parameters.

The invertible mappings from the original parameters to the new parameters are shown as follows, for $g = 1, \ldots, G$,

$$(\beta_{g1}, \ldots, \beta_{g,n_g-1}, \beta_{g,n_g}) \implies (\beta_{g1}, \ldots, \beta_{g,n_g-1}, \sum_{k=1}^{n_g} \beta_{gk}) = (\beta_{g1}, \ldots, \beta_{g,n_g-1}, s_g) \quad (3.13)$$

In words, the first $n_g - 1$ original parameters β_{gk}'s are mapped to themselves (we might use another set of symbols, for example b_{gk}, to denote the new parameters, but here we still use the old ones for simplicity of presentation while making no confusion), and the sum of all $\beta_{g,k}$'s, is mapped to s_g. Let us denote the new parameters β_{gk}, for $g = 1, \ldots, G, k = 1, \ldots, n_g - 1$, collectively by $\boldsymbol{\beta}$, and denote $s_1, \ldots, s_g$ by $\boldsymbol{s}$. (Note that $\boldsymbol{\beta}$ does not include β_{g,n_g}, for $g = 1, \ldots, G$. Once we have obtained the samples of $\boldsymbol{s}$ and $\boldsymbol{\beta}$ we can use $\beta_{g,n_g} = s_g - \sum_{k=1}^{n_g-1} \beta_{gk}$ to obtain the samples of β_{g,n_g}.)

The posterior distribution of the original parameters, β_{gk}, is:

$$P(\beta_{11}, \ \ldots \ , \beta_{G,n_G} \mid \mathcal{D}) = \frac{1}{c(\mathcal{D})} L\left(\sum_{k=1}^{n_1} \beta_{1k}, \ \ldots, \ \sum_{k=1}^{n_G} \beta_{Gk} \right) \prod_{g=1}^{G} \prod_{k=1}^{n_g} P_{gk}(\beta_{gk}) \quad (3.14)$$

By applying the standard formula for the density function of transformed random variables, we can obtain from (3.14) the posterior distribution of the $\boldsymbol{s}$ and $\boldsymbol{\beta}$:

$$P(\boldsymbol{s}, \boldsymbol{\beta} \mid \mathcal{D}) = \frac{1}{c(\mathcal{D})} L\left(s_1, \ \ldots, \ s_G\right) \prod_{g=1}^{G} \left[\prod_{k=1}^{n_g-1} P_{gk}(\beta_{gk}) \right] P_{g,n_g} \left(s_g - \sum_{k=1}^{n_g-1} \beta_{gk} \right) |\det(J)| (3.15)$$

where the $|\det(J)|$ is absolute value of the determinant of the Jacobian matrix, J, of the mapping (3.13), which can be shown to be 1.

Using the additive property of symmetric stable distributions, which is stated in section 3.3.1, we can analytically integrate out $\boldsymbol{\beta}$ in $P(\boldsymbol{s}, \boldsymbol{\beta} \mid \mathcal{D})$, resulting in the marginal distribution $P(\boldsymbol{s} \mid \mathcal{D})$:

$$\begin{aligned}
P(\boldsymbol{s} \mid \mathcal{D}) &= \int P(\boldsymbol{s}, \boldsymbol{\beta} \mid \mathcal{D})\, d\boldsymbol{\beta} && (3.16)\\
&= \frac{1}{c(\mathcal{D})} L\left(s_1, \ldots, s_G\right) \cdot \\
&\quad \prod_{g=1}^{G} \int \cdots \int \left[\prod_{k=1}^{n_g-1} P_{gk}(\beta_{gk})\right] P_{g,n_g}\left(s_g - \sum_{k=1}^{n_g-1} \beta_{gk}\right) d\beta_{g1} \cdots d\beta_{g,n_g-1} && (3.17)\\
&= \frac{1}{c(\mathcal{D})} L\left(s_1, \ldots, s_G\right)\; P_1^s(s_1)\; \cdots\; P_G^s(s_G) && (3.18)
\end{aligned}$$

The conditional distribution of $\boldsymbol{\beta}$ given $\mathcal{D}$ and $\boldsymbol{s}$ can then be obtained as follows:

$$\begin{aligned}
P(\boldsymbol{\beta} \mid \boldsymbol{s}, \mathcal{D}) &= P(\boldsymbol{s}, \boldsymbol{\beta} \mid \mathcal{D}) \,/\, P(\boldsymbol{s} \mid \mathcal{D}) && (3.19)\\
&= \prod_{g=1}^{G} \left[\prod_{k=1}^{n_g-1} P_{gk}(\beta_{gk})\right] P_{g,n_g}\left(s_g - \sum_{k=1}^{n_g-1} \beta_{gk}\right) \,/\, P_g^s(s_g) && (3.20)
\end{aligned}$$

From the above expression, it is clear that $P(\boldsymbol{\beta} \mid \boldsymbol{s}, \mathcal{D})$ is unrelated to $\mathcal{D}$, i.e., $P(\boldsymbol{\beta} \mid \boldsymbol{s}, \mathcal{D}) = P(\boldsymbol{\beta} \mid \boldsymbol{s})$, and is independent for different groups. Equation (3.12) gives this distribution only for one group g.

3.3.4 Sampling from the Splitting Distribution

In this section, we discuss how to sample from the splitting distribution (3.12) to make predictions for test cases after we have obtained samples of $s_1, \ldots, s_G$.

If we sampled for all the β_{gk}'s, storing them would require a huge amount of space when the number of parameters in each group is huge. We therefore sample for $\boldsymbol{\beta}$ conditional on $s_1, \ldots, s_G$ only temporarily, for a particular test case. As can be seen in Section 3.2.1 and 3.2.3, the predictive function needed to make prediction for a particular test case, for example the probability that a test case is in a certain class, depends only on the sums of subsets of β_{gk}'s in groups. After re-indexing the β_{gk}'s in each group such that the $\beta_{g1}, \ldots, \beta_{g,t_g}$ are those needed by the test case, the variables needed for making a prediction for the test case are:

$$s_g^t \;=\; \sum_{k=1}^{t_g} \beta_{gk}\,, \text{ for } g = 1, \ldots, G, \tag{3.21}$$

When $t_g = 0$, $s_g^t = 0$, and when $t_g = n_g$, $s_g^t = s_g$. The predictive function may also use some sums of extra regression coefficients associated with the interaction patterns that occur in this test case but not in training cases. Suppose the extra regression coefficients need to be divided into Z groups, as required by the form of the predictive function, which we denote by $\beta^*_{11}, \ldots, \beta^*_{1,n_1^*}, \ldots, \beta^*_{Z,1}, \ldots, \beta^*_{Z,n_Z^*}$. The variables needed for making prediction for the test cases are:

$$s_z^* \;=\; \sum_{k=1}^{n_z^*} \beta^*_{zk}\,, \text{ for } z = 1, \ldots, Z \tag{3.22}$$

In terms of the above variables, the function needed to make a prediction for a test case can be written as

$$a\left(\sum_{k=1}^{t_1} \beta_{1k},\; \ldots\;, \sum_{k=1}^{t_G} \beta_{Gk},\; \sum_{k=1}^{n_1^*} \beta^*_{1k},\; \ldots\;, \sum_{k=1}^{n_Z^*} \beta^*_{Zk}\right) = a(s_1^t,\; \ldots\;, s_G^t, s_1^*,\; \ldots\;, s_Z^*) \tag{3.23}$$

Let us write $s_1^t, \ldots, s_G^t$ collectively as $\boldsymbol{s}^t$, and write $s_1^*, \ldots, s_Z^*$ as $\boldsymbol{s}^*$. The integral required to make a prediction for this test case is

$$\int a(\boldsymbol{s}^t, \boldsymbol{s}^*)\; P(\boldsymbol{s}^*)\; P(\boldsymbol{s} \mid \mathcal{D}) \prod_{g=1}^{G} P(s_g^t \mid s_g)\; d\boldsymbol{s}\; d\boldsymbol{s}^t d\boldsymbol{s}^*. \tag{3.24}$$

The integral over $\boldsymbol{s}^t$ is done by MCMC. We also need to sample for $\boldsymbol{s}^*$ from $P(\boldsymbol{s}^*)$, which is the prior distribution of $\boldsymbol{s}^*$ given some hyperparameters (from the current MCMC iteration) and can therefore be sampled easily. Finally, we need to sample from $P(s_g^t \mid s_g)$, which can be derived from (3.12), shown as follows:

$$P(s_g^t \mid s_g) = P_g^{(1)}(s_g^t)\; P_g^{(2)}(s_g - s_g^t) \,/\, P_g^s(s_g) \tag{3.25}$$

where $P_g^{(1)}$ and $P_g^{(2)}$ are the priors (either Gaussian or Cauchy) of $\sum_1^{t_g} \beta_{gk}$ and $\sum_{t_g+1}^{n_g} \beta_{gk}$, respectively. We can obtain (3.25) from (3.12) analogously as we obtained the density of s_g, that is, by first mapping $\boldsymbol{\beta}$ and $\boldsymbol{s}$ to a set of new parameters containing $\boldsymbol{s}$ and $\boldsymbol{s}^t$, then integrating away other parameters, using the additive property of symmetric stable distributions. The distribution (3.25) splits s_g into two components.

When the priors for the β_{gk}'s are Gaussian distributions, the distribution (3.25) is also a Gaussian distribution, given as follows:

$$s_g^t \mid s_g \;\sim\; N\left(s_g \,\frac{\Sigma_1^2}{\Sigma_1^2 + \Sigma_2^2}\;,\; \Sigma_1^2\left(1 \;-\; \frac{\Sigma_1^2}{\Sigma_1^2 + \Sigma_2^2}\right)\right) \tag{3.26}$$

where $\Sigma_1^2 = \sum_{k=1}^{t_g} \sigma_{gk}^2$ and $\Sigma_2^2 = \sum_{t_g+1}^{n_g} \sigma_{gk}^2$. Since (3.26) is a Gaussian distribution, we can sample from it by standard methods.

When we use Cauchy distributions as the priors for the β_{gk}'s, the density function of (3.25) is:

$$P(s_g^t \mid s_g) = \frac{1}{C}\,\frac{1}{\Sigma_1^2 + (s_g^t)^2}\,\frac{1}{\Sigma_2^2 + (s_g^t - s_g)^2} \tag{3.27}$$

where $\Sigma_1 = \sum_{k=1}^{t_g} \sigma_{gk}$, $\Sigma_2 = \sum_{t_g+1}^{n_g} \sigma_{gk}$, and C is the normalizing constant given below by (3.29).

When $s_g = 0$ and $\Sigma_1 = \Sigma_2$, the distribution (3.27) is a t-distribution with 3 degrees of freedom, mean 0 and width $\Sigma_1/\sqrt{3}$, from which we can sample by standard methods. Otherwise, the cumulative distribution function (CDF) of (3.27) can be shown to be:

$$\begin{aligned} F(s_g^t\,;\, s_g, \Sigma_1, \Sigma_2) \;\;=\;\; & \frac{1}{C}\left[r \log\left(\frac{(s_g^t)^2 + \Sigma_1^2}{(s_g^t - s_g)^2 + \Sigma_2^2}\right) + \right. \\ & p_0\left(\arctan\left(\frac{s_g^t}{\Sigma_1}\right) + \frac{\pi}{2}\right) + \\ & \left. p_s\left(\arctan\left(\frac{s_g^t - s_g}{\Sigma_2}\right) + \frac{\pi}{2}\right)\right] \end{aligned} \tag{3.28}$$

where

$$\begin{aligned}
C &= \frac{\pi\left(\Sigma_1+\Sigma_2\right)}{\Sigma_1\Sigma_2\left(s_g^2+(\Sigma_1+\Sigma_2)^2\right)}\,, && (3.29)\\
r &= \frac{s_g}{s_g^4+2\left(\Sigma_1^2+\Sigma_2^2\right)\,s_g^2+\left(\Sigma_1^2-\Sigma_2^2\right)^2}\,, && (3.30)\\
p_0 &= \frac{1}{\Sigma_1}\,\frac{s_g^2-\left(\Sigma_1^2-\Sigma_2^2\right)}{s_g^4+2\left(\Sigma_1^2+\Sigma_2^2\right)\,s_g^2+\left(\Sigma_1^2-\Sigma_2^2\right)^2}, && (3.31)\\
p_s &= \frac{1}{\Sigma_2}\,\frac{s_g^2+\left(\Sigma_1^2-\Sigma_2^2\right)}{s_g^4+2\left(\Sigma_1^2+\Sigma_2^2\right)\,s_g^2+\left(\Sigma_1^2-\Sigma_2^2\right)^2} && (3.32)
\end{aligned}$$

When $s_g \neq 0$, the derivation of (3.28) uses the equations below from (3.33) to (3.35) as follows, where $p = (a^2-c)/b, q = b+q, r = pc - a^2q$, and we assume $4c - b^2 > 0$,

$$\begin{aligned}
\frac{1}{x^2+a^2}\,\frac{1}{x^2+bx+c} = \frac{1}{r}\left(\frac{x+p}{x^2+a^2}-\frac{x+q}{x^2+bx+c}\right) && (3.33)\\
\int_{-\infty}^{x}\frac{u+p}{u^2+a^2}du = \frac{1}{2}\,\log(x^2+a^2)+\frac{p}{a}\,\arctan\left(\frac{x}{a}\right)+\frac{\pi}{2} && (3.34)\\
\int_{-\infty}^{x}\frac{u+q}{u^2+bu+c}du = \frac{1}{2}\,\log(x^2+bx+c)+\frac{2q-b}{\sqrt{4c-b^2}}\,\arctan\left(\frac{2x+b}{\sqrt{4c-b^2}}\right)+\frac{\pi}{2} && (3.35)
\end{aligned}$$

When $s_g = 0$, the derivation of (3.28) uses the following equations:

$$\begin{aligned}
\frac{1}{x^2+a^2}\,\frac{1}{x^2+b^2} &= \frac{1}{b^2-a^2}\left(\frac{1}{x^2+a^2}-\frac{1}{x^2+b^2}\right) && (3.36)\\
\int_{-\infty}^{x}\frac{1}{u^2+a^2}\,du &= \frac{1}{a}\left(\arctan\left(\frac{x}{a}\right)+\frac{\pi}{2}\right) && (3.37)
\end{aligned}$$

Since we can compute the CDF of (3.27) with (3.28) explicitly, we can use the inversion method to sample from (3.27), with the inverse CDF computed by some numerical method. We chose the Illinois method (Thisted 1988, Page 171), which is robust and fairly fast.

When sampling for $s_1^t, \ldots, s_G^t$ temporarily for each test case is not desired, for example, when we need to make predictions for a huge number of test cases at a time, we can still apply the above method that splits a Gaussian or Cauchy random variable into two parts

$n_g - 1$ times to split s_g into n_g parts. Our method for compressing parameters is still useful because sampling from the splitting distributions uses direct sampling methods, which are much more efficient than applying Markov chain sampling method to the original parameters. However, we will not save space if we take this approach of sampling for all β's.

3.4 Application to Sequence Prediction Models

In this section, we show how to compress parameters of logistic sequence prediction models in which states of a sequence are discrete, as defined in Section 3.2.1. To demonstrate our method, we use a binary data set generated using a hidden Markov model, and a data set created from English text, in which each state has 3 possibilities (consonant, vowel, and others). These experiments show that our compression method produces a large reduction in the number of parameters needed for training the model, when the prediction for the next state of a sequence is based on a long preceding sequence, i.e., a high-order model. We also show that good predictions on test cases result from being able to use a high-order model.

3.4.1 Grouping Parameters of Sequence Prediction Models

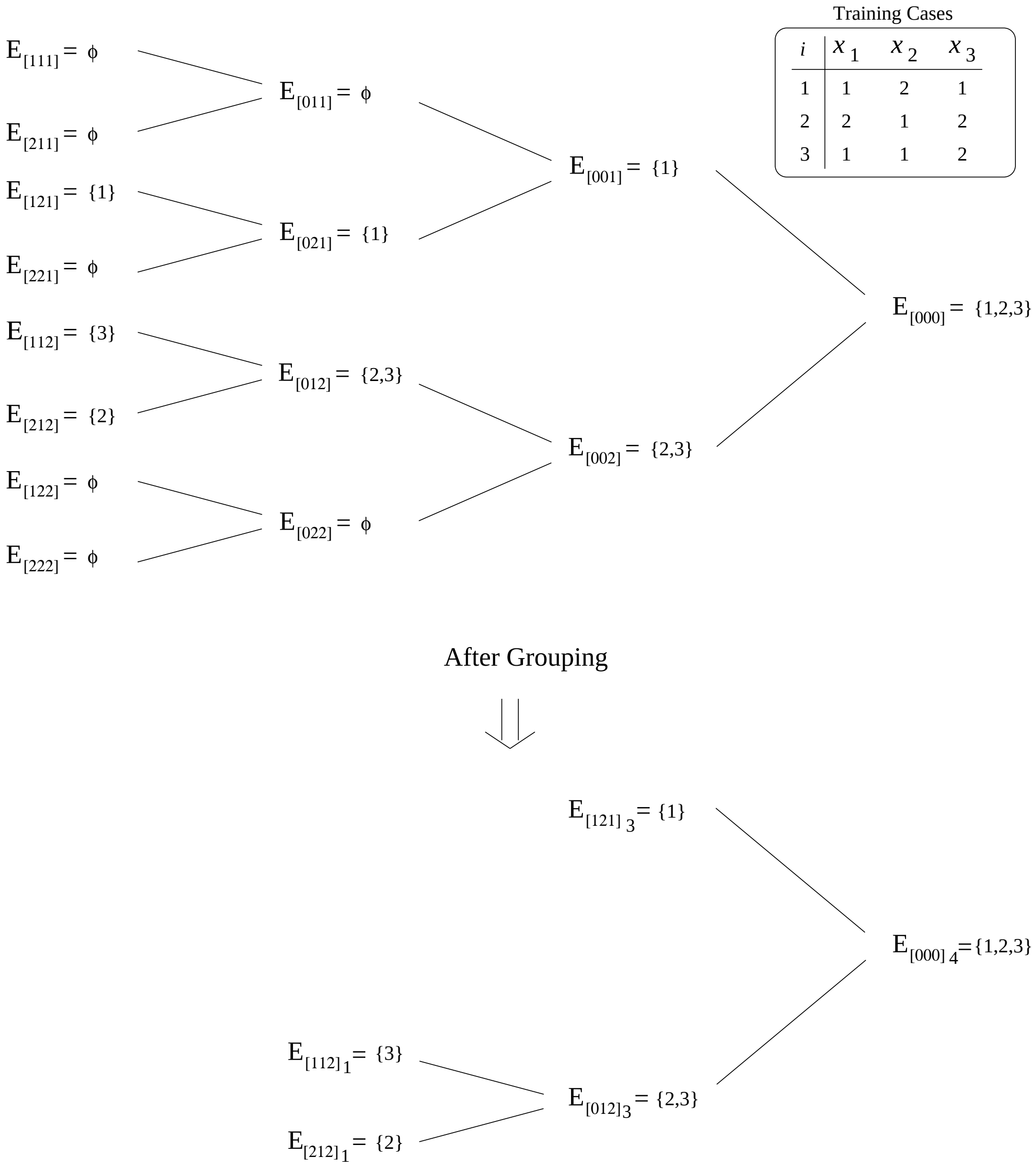
In this section, we describe a scheme for dividing the β's into a number of groups, based on the training data, such that the likelihood function depends only on the sums in groups, as shown by (3.8).

Since the linear functions for different values of response have the same form except the superscript, the way we divide $\boldsymbol{\beta}^{(k)}$ into groups is the same for all k. Our task is to find the groups of interaction patterns expressed by the same training cases.

Let us use $E_{\mathcal{P}}$ to denote the "expression" of the pattern $\mathcal{P}$ — the indices of training cases in which $\mathcal{P}$ is expressed, a subset of $1, \ldots, N$. For example, $E_{[0\cdots0]} = \{1, \ldots, N\}$. In other words, the indicator for pattern $\mathcal{P}$ has value 1 for the training cases in $E_{\mathcal{P}}$, and 0 for others. We can display $E_{\mathcal{P}}$ in a tree-shape, as we displayed $\beta_{\mathcal{P}}$. The upper part of Figure 3.4 shows such expressions for each pattern of binary sequence of length $O = 3$, based on 3 training cases: $\boldsymbol{x}_{1:3}^{(1)} = (1, 2, 1)$,$\boldsymbol{x}_{1:3}^{(2)} = (2, 1, 2)$ and $\boldsymbol{x}_{1:3}^{(3)} = (1, 1, 2)$. From Figure 3.4, we can see that the expression of a "stem" pattern is equal to the union of the expressions of its "leaf" patterns, for example, $E_{[000]} = E_{[001]} \bigcup E_{[002]}$.

When a stem pattern has only one leaf pattern with non-empty expression, the stem and leaf patterns have the same expression, and can therefore be grouped together. This grouping procedure will continue by taking the leaf pattern as the new stem pattern, until encountering a stem pattern that "splits", i.e. has more than one leaf pattern with non-empty expression. For example, $E_{[001]}, E_{[021]}$ and $E_{[121]}$ in Figure 3.4 can be grouped together. All such patterns must be linked by lines, and can be represented collectively with a "superpattern" SP, written as $[0 \cdots 0 A_b \cdots A_O]_f = \bigcup_{t=f}^{b} [0 \cdots 0 A_t \cdots A_O]$, where $1 \leq b \leq f \leq O + 1$, and in particular when $t = O + 1$, $[0 \cdots 0 A_t \cdots A_O] = [0 \cdots 0]$. One can easily translate the above discussion into a computer algorithm. Figure 3.5 describes the algorithm for grouping parameters of Bayesian logistic sequence prediction models, in a C-like language, using a recursive function.

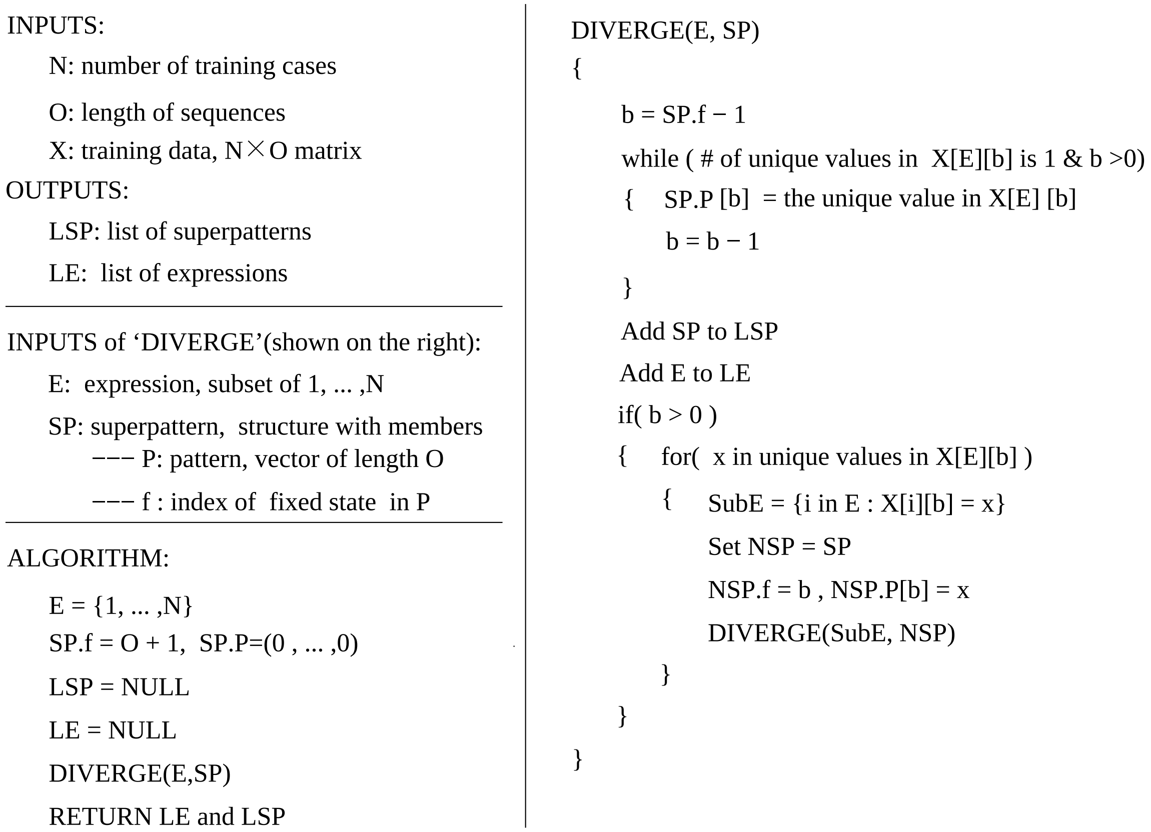
An important property of our method for compressing parameters of sequence prediction models is that given N sequences as training data, conceivably of infinite length, denoted by $x_{-\infty}^{(i)}, \ldots, x_{-1}^{(i)}$, for $i = 1, \ldots, N$, the number of superpatterns with unique expressions, and accordingly the number of compressed parameters, will converge to a finite number as O increases. The justification of this claim is that if we keep splitting the expressions following the tree shown in Figure 3.4, at a certain time, say t, every expression will be an expression with only 1 element (suppose we in advance remove the sequences that are identical with another one). When considering further smaller t, no more new superpattern with different expressions will be introduced, and the number of superpatterns will not grow. The number of the *compressed parameters*, the regression coefficients for the superpatterns, will therefore



Training Cases

i	x_1	x_2	x_3
1	1	2	1
2	2	1	2
3	1	1	2

Figure 3.4: A picture showing that the interaction patterns in logistic sequence prediction models can be grouped, illustrated with binary sequences of length $O = 3$, based on 3 training cases shown in the upper-right box. $E_{\mathcal{P}}$ is the expression of the pattern (or superpattern) $\mathcal{P}$ — the indices of the training cases in which the $\mathcal{P}$ is expressed, with ϕ meaning the empty set. We group the patterns with the same expression together, re-represented collectively by a "superpattern", written as $[0\cdots 0A_b\cdots A_O]_f$, meaning $\bigcup_{t=b}^{f}[0\cdots 0A_t\cdots A_O]$, where $1 \le b \le f \le O+1$, and in particular when $t = O+1$, $[0\cdots 0A_t\cdots A_O] = [0\cdots 0]$. We also remove the patterns not expressed by any of the training cases. Only 5 superpatterns with unique expressions are left in the lower picture.



```
INPUTS:
    N: number of training cases
    O: length of sequences
    X: training data, N×O matrix
OUTPUTS:
    LSP: list of superpatterns
    LE:  list of expressions
-----------------------------------------------
INPUTS of ‘DIVERGE’(shown on the right):
    E:  expression, subset of 1, ... ,N
    SP: superpattern,  structure with members
        −−− P: pattern, vector of length O
        −−− f : index of  fixed state  in P
-----------------------------------------------
ALGORITHM:
    E = {1, ... ,N}
    SP.f = O + 1,  SP.P=(0 , ... ,0)
    LSP = NULL
    LE = NULL
    DIVERGE(E,SP)
    RETURN LE and LSP

DIVERGE(E, SP)
{
    b = SP.f − 1
    while ( # of unique values in  X[E][b] is 1 & b >0)
    {   SP.P [b]  = the unique value in X[E] [b]
        b = b − 1
    }
    Add SP to LSP
    Add E to LE
    if( b > 0 )
    {   for(  x in unique values in X[E][b] )
        {   SubE = {i in E : X[i][b] = x}
            Set NSP = SP
            NSP.f = b , NSP.P[b] = x
            DIVERGE(SubE, NSP)
        }
    }
}
```

Figure 3.5: The algorithm for grouping parameters of Bayesian logistic sequence prediction models. To group parameters, we call function “DIVERGE” with the initial values of expression $E = \{1, \ldots, N\}$ and superpattern $SP = [0 \ldots 0]_{O+1}$, as shown in above picture, resulting in two lists of the same length, LE and LSP, respectively storing the expressions and the corresponding superpatterns. Note that the first index of an array is assumed to be 1, and that the $X[E][b]$ means a 1-dimension subarray of X in which the row indices are in E and the column index equals b.

not grow after the time t.

In contrast, after the time t when each interaction pattern is expressed by only 1 training case, if the order is increased by 1, the number of interaction patterns is increased by the number of training cases. The regression coefficients associated with these original interaction patterns, called *the original parameters* thereafter, will grow linearly with the order considered. Note that these original parameters do not include the regression coefficients for those interaction patterns not expressed by any training case. The total number of regression coefficients defined by the model grows exponentially with the order considered.

3.4.2 Making Prediction for a Test Case

Given $\beta^{(1)}, \ldots, \boldsymbol{\beta}^{(K)}$, the predictive probability for the next state $\boldsymbol{x}^*_{O+1}$ of a test case for which we know the historic sequence $\boldsymbol{x}^*_{1:O}$ can be computed using equation (3.1), applied to $\boldsymbol{x}^*_{1:O}$. A Monte Carlo estimate of $P(x^*_{O+1} = k \mid \boldsymbol{x}^*_{1:O}, \mathcal{D})$ can be obtained by averaging (3.1) over the Markov chain samples from the posterior distribution of $\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(K)}$.

Each of the $O+1$ patterns expressed by the test case $\boldsymbol{x}^*_{1:O}$ is either expressed by some training case (and therefore belongs to one of the superpatterns), or is a new pattern (not expressed by any training case). Suppose we have found γ superpatterns. The $O+1$ β's in the linear function $l(\boldsymbol{x}^*_{1:O}, \beta^{(k)})$ can accordingly be divided into $\gamma+1$ groups (some groups may be empty). The function $l(\boldsymbol{x}^*_{1:O}, \beta^{(k)})$ can be written as the sum of the sums of the β's over these $\gamma+1$ groups. Consequently, $P(x^*_{O+1} = k \mid \boldsymbol{x}^*_{1:O})$ can be written in the form of (3.23). As discussed in Section 3.3.4, we need to only split the sum of the β's associated with a superpattern, i.e., a compressed parameter s_g, into two parts, such that one of them is the sum of those β expressed by the test case $\boldsymbol{x}^*_{1:O}$, using the splitting distribution (3.25).

It is easy to identify the patterns that are also expressed by $\boldsymbol{x}^*_{1:O}$ from a superpattern $[0 \cdots A_b \cdots A_O]_f$. If $(x^*_f, \ldots, x^*_O) \neq (A_f, \ldots, A_O)$, none of the patterns in $[0 \cdots A_b \cdots A_O]_f$ are expressed by $\boldsymbol{x}^*_{1:O}$, otherwise, if $(x^*_{b'}, \ldots, x^*_O) = (A_{b'}, \ldots, A_O)$ for some b' $(b \leq b' \leq f)$, all patterns in $[0 \cdots A_{b'} \cdots A_O]_f$ are expressed by $\boldsymbol{x}^*_{1:O}$.

3.4.3 Experiments with a Hidden Markov Model

In this section we apply Bayesian logistic sequence prediction modeling, with or without our compression method, to data sets generated using a Hidden Markov model, to demonstrate our method for compressing parameters. The experiments show that when the considered length of the sequence O is increased, the number of compressed parameters will converge to a fixed number, whereas the number of original parameters will increase linearly. Our compression method also improves the quality of Markov chain sampling in terms of autocorrelation. We therefore obtain good predictive performances in a small amount of time using long historic sequences.

The Hidden Markov Model Used to Generate the Data

Hidden markov models (HMM) are applied widely in many areas, for example, speech recognition (Baker 1975), image analysis (Romberg et.al. 2001), computational biology (Sun 2006). In a simple hidden Markov model, the observable sequence $\{x_t \mid t = 1, 2, \ldots\}$ is modeled as a noisy representation of a hidden sequence $\{h_t \mid t = 1, 2, \ldots\}$ that has the Markov property (the distribution of h_t given h_{t-1} is independent with the previous states before h_{t-1}). Figure 3.6 displays the hidden Markov model used to generate our data sets, showing the transitions of three successive states. The hidden sequence h_t is an Markov chain with state space $\{1, \ldots, 8\}$, whose dominating transition probabilities are shown by the arrows in Figure 3.6, each of which is 0.95. However, the hidden Markov chain can move from any state to any other state as well, with some small probabilities. If h_t is an even number, x_t will be equal to 1 with probability 0.95 and 2 with probability 0.05, otherwise, x_t will be equal to 2 with probability 0.95 and 1 with probability 0.05. The sequence $\{x_t \mid t = 1, 2, \ldots\}$ generated by this exhibits high-order dependency, though the hidden sequence is only a Markov chain.

We can see this by looking at the transitions of observable x_t in Figure 3.6. For example, if $x_1 = 1$ (rectangle) and $x_2 = 2$ (oval), it is most likely to be generated by $h_1 = 2$ and $h_2 = 3$, since this is the only strong connection from the rectangle to the oval, consequently, $h_3 = 8$ is most likely to to be the next, and x_3 is therefore most likely to be 1 (rectangle).

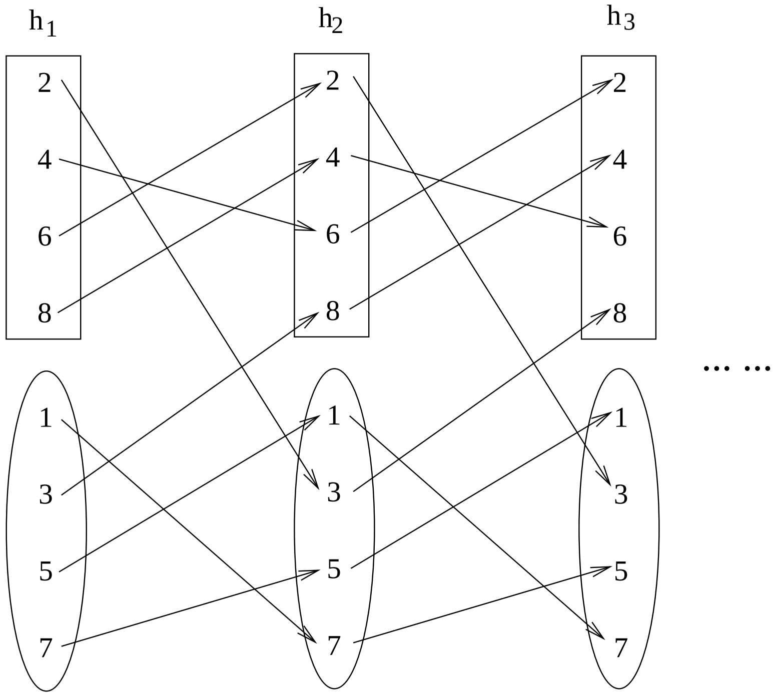


Figure 3.6: A picture showing a Hidden Markov Model, which is used to generate sequences to demonstrate Bayesian logistic sequence prediction models. Only the dominating transition probabilities of 0.95 are shown using arrows in the above graph, while from any state the hidden Markov chain can also move to any other state with a small probability. When h_t is in a rectangle, x_t is equal to 1 with probability 0.95, and 2 with probability 0.05, otherwise, when h_t is in an oval, x_t is equal to 2 with probability 0.95, and 1 with probability 0.05.

3.4.4 Specifications of the Priors and Computation Methods

The Priors for the Hyperprameters

We fix σ_0 at 5 for the Cauchy models and 10 for the Gaussian models. For $o > 0$, the prior for σ_o is Inverse Gamma$(\alpha_o, (\alpha_o + 1)w_o)$, where α_o and w_o are:

$$\alpha_o = 0.25, \quad w_o = 0.1/o, \quad \text{for } o = 1, \ldots, O \tag{3.38}$$

The quantiles of Inverse-Gamma$(0.25, 1.25 \times 0.1)$, the prior of σ_1, are shown as follows:

p	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.99
q	0.05	0.17	0.34	0.67	1.33	2.86	7.13	22.76	115.65	1851.83	1.85×10^7

The quantiles of other σ_o can be obtained by multiplying those of σ_1 by $1/o$.

The Markov Chain Sampling Method

We use Gibbs sampling to sample for both the s_g's (or the β_{gk}'s when not applying our compression method) and the hyperparameters, σ_o. These 1-dimensional conditional distributions are sampled using the slice sampling method (Neal 2003), summarized as follows. In order to sample from a 1-dimensional distribution with density $f(x)$, we can draw points (x, y) from the uniform distribution over the set $\{(x, y) \mid 0 < y < f(x)\}$, i.e., the region of the 2-dimensional plane between the x-axis and the curve of $f(x)$. One can show that the marginal distribution of x drawn this way is $f(x)$. We can use Gibbs sampling scheme to sample from the uniform distribution over $\{(x, y) \mid 0 < y < f(x)\}$. Given x, we can draw y from the uniform distribution over $\{y \mid 0 < y < f(x)\}$. Given y, we need to draw x from the uniform distribution over the "slice", $S = \{x \mid f(x) > y\}$. However, it is generally infeasible to draw a point directly from the uniform distribution over S. (Neal 2003) devises several Markov chain sampling schemes that leave this uniform distribution over S invariant. One can show that this updating of x along with the previous updating of y leaves $f(x)$ invariant. Particularly we chose the "stepping out" plus "shrinkage" procedures. The "stepping out" scheme first steps out from the point in the previous iteration, say x_0, which is in S, by expanding an initial interval, I, of size w around x_0 on both sides with intervals of size w, until the ends of I are outside S, or the number of steps has reached a pre-specified number, m. To guarantee correctness, the initial interval I is positioned randomly around x_0, and

m is randomly aportioned for the times of stepping right and stepping left. We then keep drawing a point uniformly from I until obtaining an x in S. To facilitate the process of obtaining an x in S, we shrink the interval I if we obtain an x not in S by cutting off the left part or right part of I depending on whether $x < x_0$ or $x > x_0$.

We set $w = 20$ when sampling for β's if we use Cauchy priors, considering that there might be two modes in this case, and set $w = 10$ if we use Gaussian priors. We set $w = 1$ when sampling for σ_o. The value of m is 50 for all cases. We trained the Bayesian logistic sequence model, with the compressed or the original parameters, by running the Markov chain 2000 iterations, each updating the β's 1 time, and updating the σ's 10 times, both using slice sampling. The first 750 iterations were discarded, and every 5th iteration afterward was used to predict for the test cases. The number of 750 was chosen empirically after looking at many trial runs of Markov chains for many different circumstances.

The above specification of Markov chain sampling and the priors for the hyperparameters will be used for all experiments in this chapter, including the experiments on classification models discussed in Section 3.5.

Experiment Results

We used the HMM in Figure 3.6 to generate 5500 sequences with length 21. We used 5000 sequences as test cases, and the remaining 500 as the training cases. We tested the prediction methods by predicting x_{21} based on varying numbers of preceding states, O, chosen from the set $\{1, 2, 3, 4, 5, 7, 12, 15, 17, 20\}$.

Figure 3.7 compares the number of parameters and the times used to train the model, with and without our compression method. It is clear that our method for compressing parameters reduces greatly the number of parameters. The ratio of the number of compressed parameters to the number of the original ones decreases with the number of preceding states, O. For example, the ratio reaches 0.207 when $O = 20$. This ratio will reduce to 0 when considering even bigger O, since the number of original parameters will grow with O while the number of compressed parameters will converge to a finite number, as discussed in Section 3.4.1. There are similar reductions for the training times with our compression method. But the training time with compressed parameters will not converge to a finite amount, since the time used to update the hyperparameters (σ_o's) grows with order, O. Figure 3.7 also shows the prediction times for 5000 training cases. The small prediction times show that the methods for splitting Gaussian and Cauchy variables are very fast. The prediction times grow with O because the time used to identify the patterns in a superpattern expressed by a test case grows with O. The prediction times with the original parameters are not shown in Figure 3.7, since we do not claim that our compression method saves prediction time. (If we used the time-optimal programming method for each method, the prediction times with compressed parameters should be more than without compressing parameters since the method with compression should include times for identifying the patterns from the superpattern for test cases. With our software, however, prediction times with compression are less than without compression, which is not shown in Figure 3.7, because the method without compression needs to repeatedly read a huge number of the original parameters into memory from disk.)

Compressing parameters also improves the quality of Markov chain sampling. Figure 3.8 shows the autocorrelation plots of the hyperparameters σ_o, for $o = 10, 12, 15, 17, 20$, when the length of the preceding sequence, O, is 20. It is clear that the autocorrelation decreases more rapidly with lag when we compress the parameters. This results from the compressed parameters capturing the important directions of the likelihood function (i.e. the directions where a small change can result in large a change of the likelihood). We did not take the time reduction from compressing parameters into consideration in this comparison. If we rescaled the lags in the autocorrelation plots according to the computation time, the reduction of autocorrelation of Markov chains with the compressed parameters would be much more pronounced.

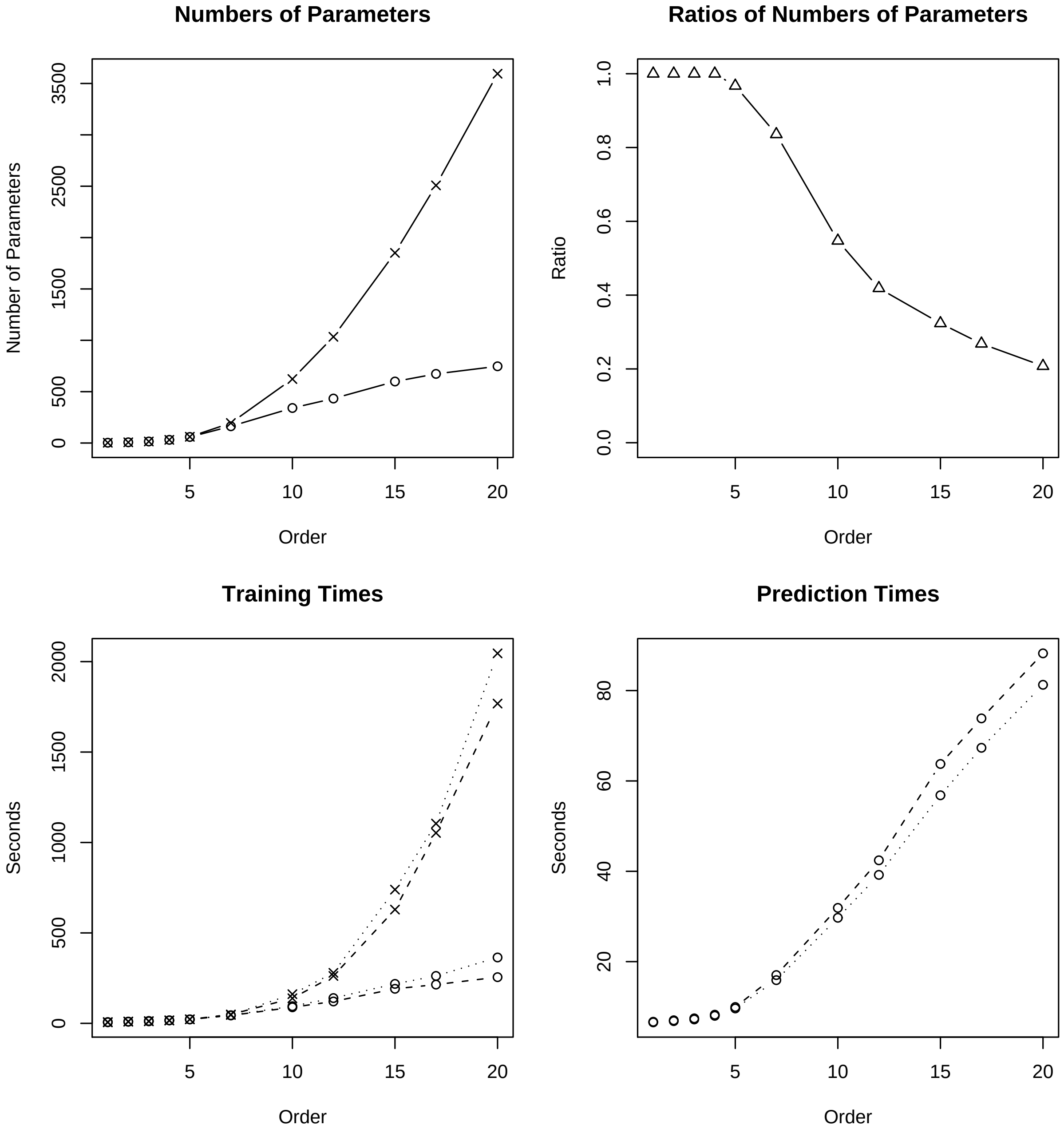


Figure 3.7: Plots showing the reductions of the number of parameters and the training time with our compression method using the experiments on a data set generated by a HMM. The upper-left plot shows the number of the compressed and the original parameters based on 500 training sequences for $O = 1, 2, 3, 4, 5, 7, 10, 12, 15, 17, 20$, their ratios are shown in the upper-right plot. In the above plots, the lines with $\circ$ are for the methods with parameters compressed, the lines with $\times$ are for the methods without parameters compressed, the dashed lines are for the methods with Gaussian priors, and the dotted lines are for the methods with Cauchy priors. The lower-left plot shows the training times for the methods with and without parameters compressed. The lower-right plot shows the prediction time only for the methods with parameters compressed.

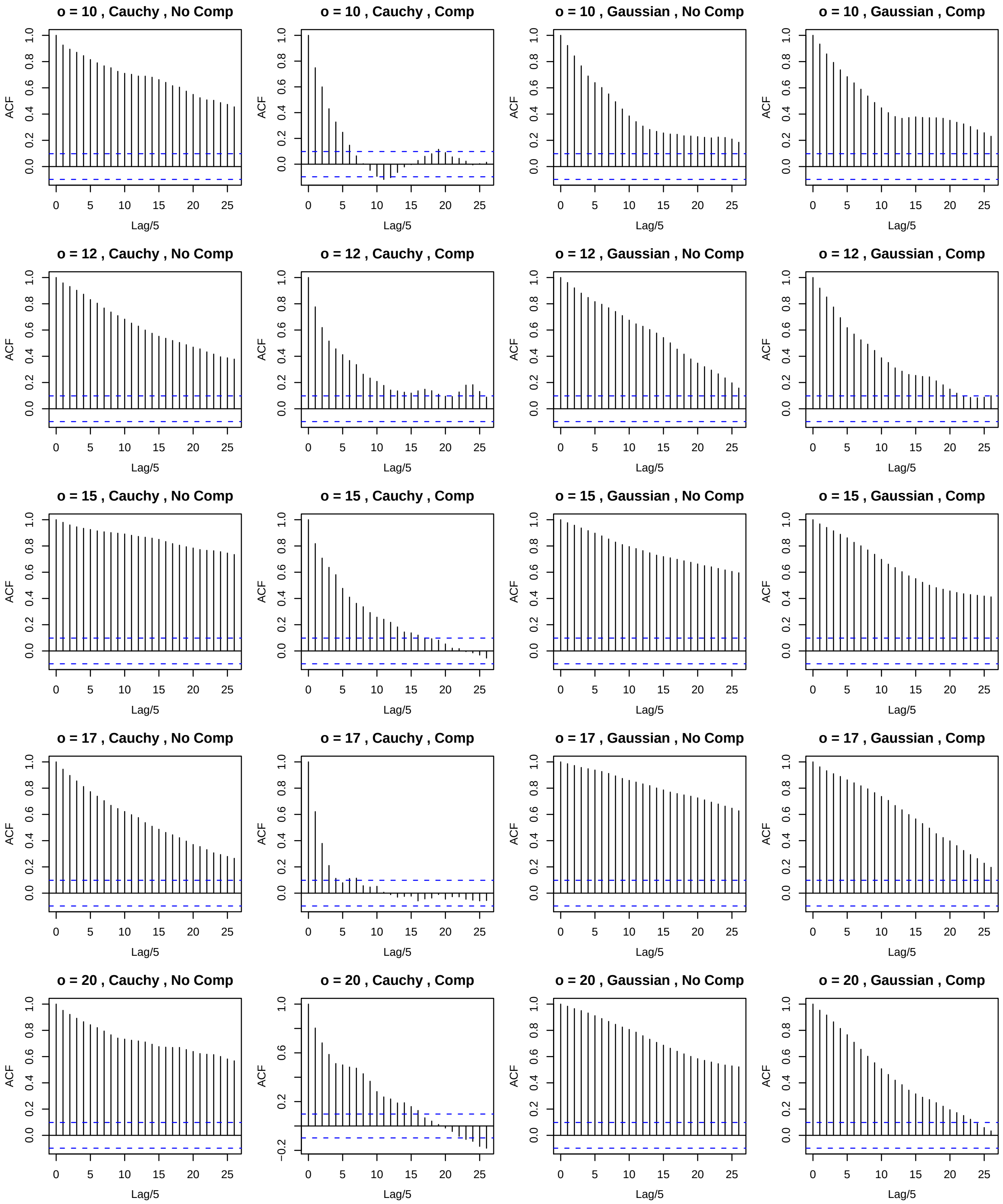


Figure 3.8: The autocorrelation plots of σ_o's for the experiments on a data set generated by a HMM, when the length of the preceding sequence $O = 20$. We show the autocorrelations of σ_o, for $o = 10, 12, 15, 17, 20$. In the above plots, "Gaussian" in the titles indicates the methods with Gaussian priors, "Cauchy" indicates with Cauchy priors, "comp" indicates with parameters compressed, "no comp" indicates without parameters compressed.

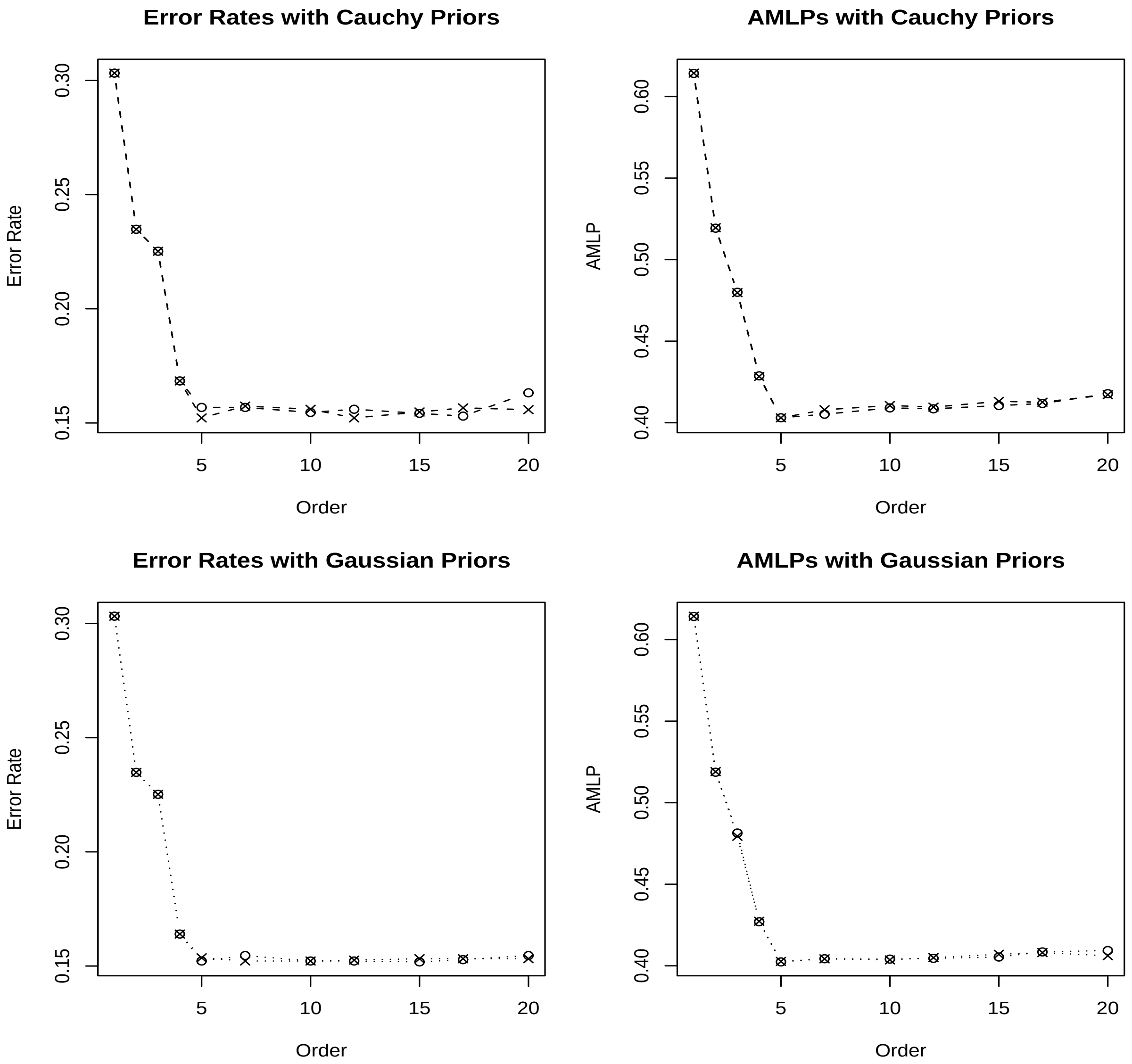


Figure 3.9: Plots showing the predictive performance using the experiments on a data set generated by a HMM. The left plots show the error rates and the right plots show the average minus log probabilities of the true responses in the test cases. The upper plots show the results when using the Cauchy priors and the lower plots shows the results when using the Gaussian priors. In all plots, the lines with $\circ$ are for the methods with parameters compressed, the lines with $\times$ are for the methods without parameters compressed. The numbers of the training and test cases are respectively 500 and 5000. The number of classes of the response is 2.

Finally, we evaluated the predictive performance in terms of error rate (the fraction of wrong predictions in test cases), and the average minus log probability (AMLP) of observing the true response in a test case based on the predictive probability for different classes. The performance of with and without compressing parameters are the same, as should be the case in theory, and will be in practice when the Markov chains for the two methods converge to the same modes. Performance of methods with Cauchy and Gaussian priors is also similar for this example. The predictive performance is improved when O goes from 1 to 5. When $O > 5$ the predictions are slightly worse than with $O = 5$ in terms of AMLP. The error rates for $O > 5$ are almost the same as for $O = 5$. This shows that the Bayesian models can perform reasonably well even when we consider a very high order, as they avoid the overfitting problem in using complex models. We therefore do not need to restrict the order of the Bayesian sequence prediction models to a very small number, especially after applying our method for compressing parameters.

3.4.5 Experiments with English Text

We also tested our method using a data set created from an online article from the website of the Department of Statistics, University of Toronto. In creating the data set, we encoded each character as 1 for vowel letters (a,e,i,o,u), 2 for consonant letters, and 3 for all other characters, such as space, numbers, special symbols, and we then collapsed multiple occurrences of "3" into only 1 occurrence. The length of the whole sequence is 3930. Using it we created a data set with 3910 overlaped sequences of length 21, and used the first 1000 as training data.

The experiments were similar to those in Section 3.4.3, with the same priors and the same computational specifications for Markov chain sampling. Figures 3.10, 3.11, 3.12, and 3.13 show the results. All the conclusions drawn from the experiments in Section 3.4.3 are confirmed in this example, with some differences in details. In summary, our compression

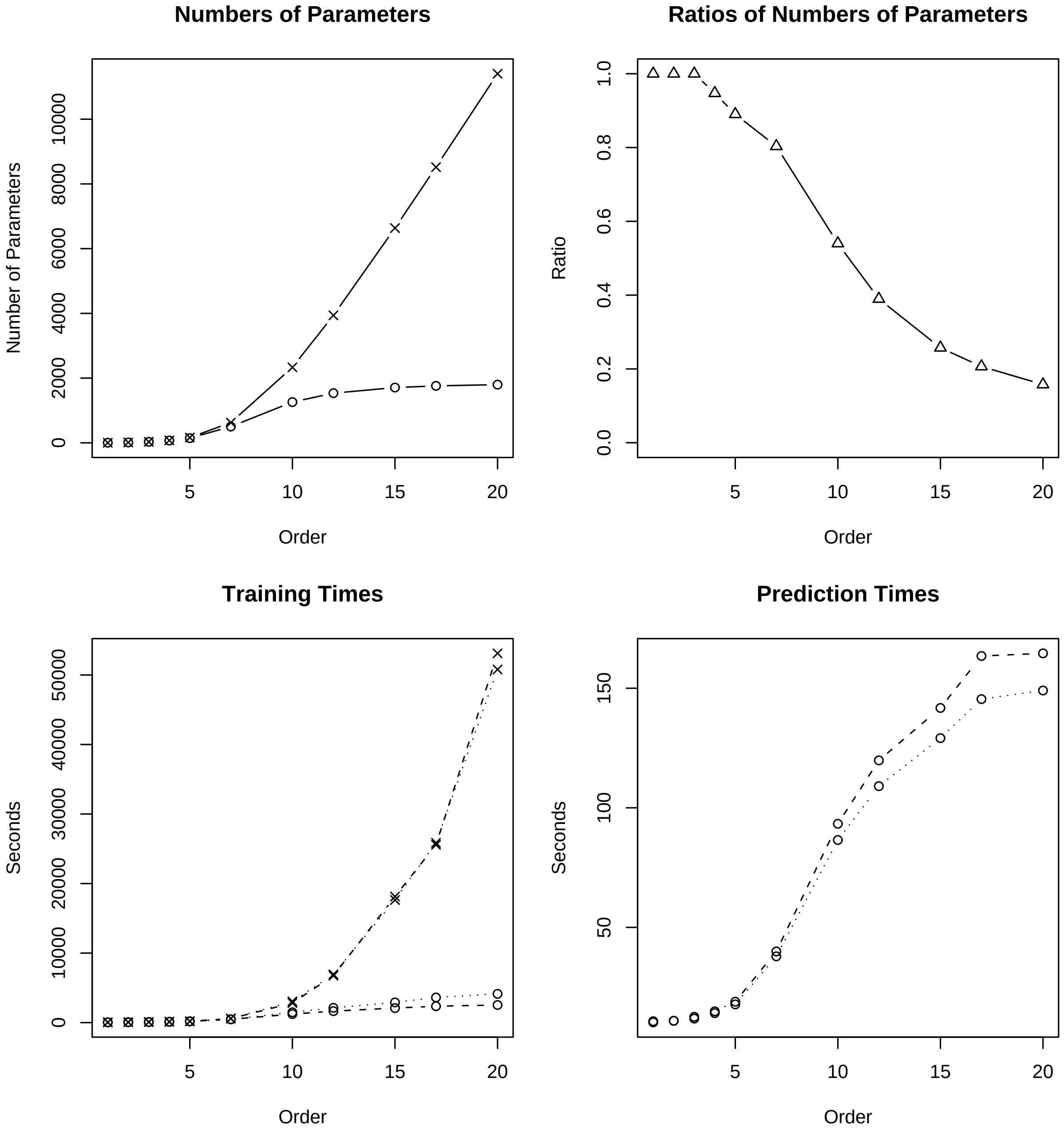
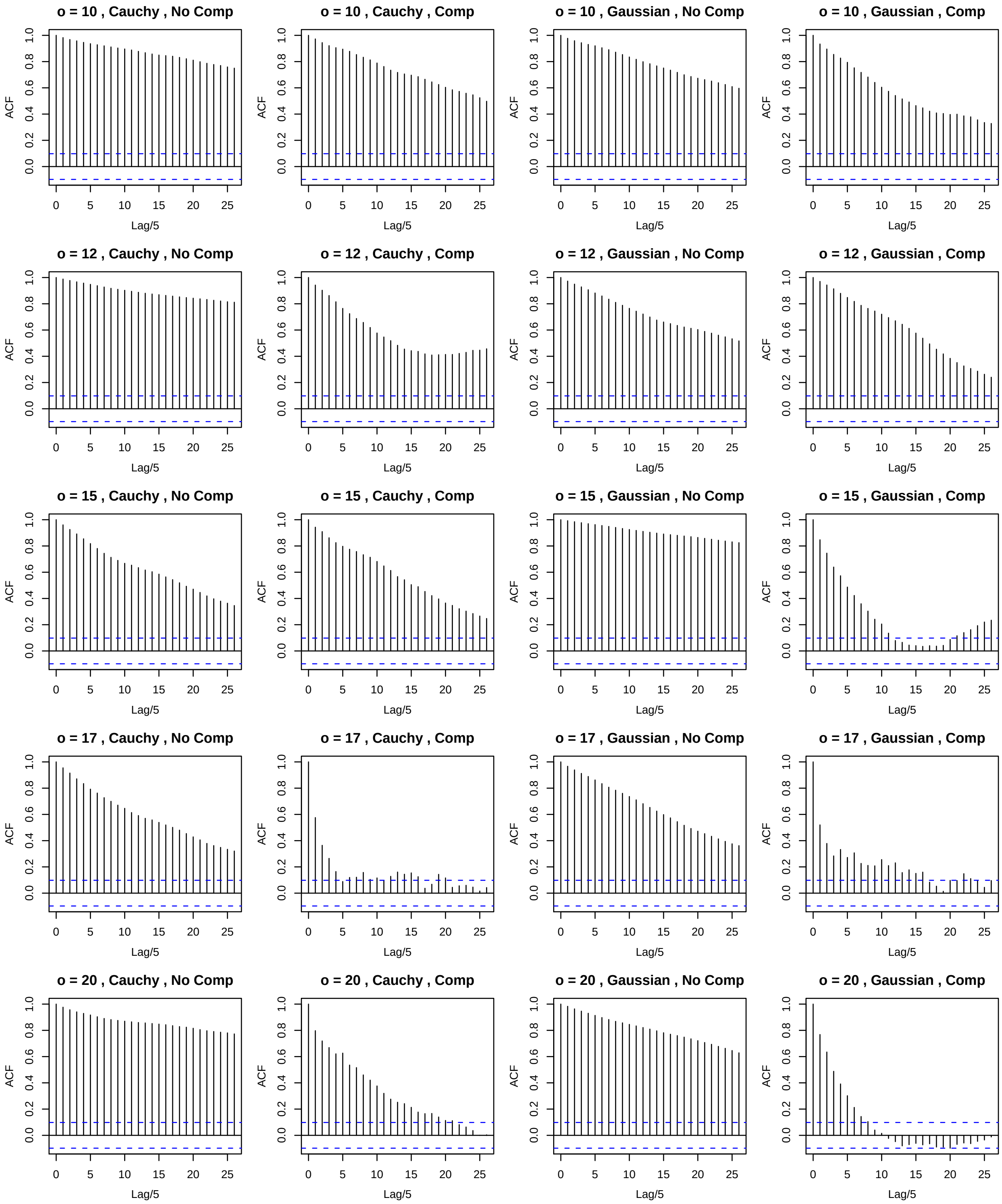


Figure 3.10: Plots showing the reductions of the number of parameters and the training and prediction time with our compression method using the experiments on English text. The upper-left plot shows the number of the compressed and the original parameters based on 500 training sequences for $O = 1, 2, 3, 4, 5, 7, 10, 12, 15, 17, 20$, their ratios are shown in the upper-right plot. In the above plots, the lines with $\circ$ are for the methods with parameters compressed, the lines with × are for the methods without parameters compressed, the dashed lines are for the methods with Gaussian priors, and the dotted lines are for the methods with Cauchy priors. The lower-left plot shows the training times for the methods with and without parameters compressed. The lower-right plot shows the prediction time only for the methods with parameters compressed.


o = 10 , Cauchy , No Comp

o = 10 , Cauchy , Comp

o = 10 , Gaussian , No Comp

o = 10 , Gaussian , Comp

o = 12 , Cauchy , No Comp

o = 12 , Cauchy , Comp

o = 12 , Gaussian , No Comp

o = 12 , Gaussian , Comp

o = 15 , Cauchy , No Comp

o = 15 , Cauchy , Comp

o = 15 , Gaussian , No Comp

o = 15 , Gaussian , Comp

o = 17 , Cauchy , No Comp

o = 17 , Cauchy , Comp

o = 17 , Gaussian , No Comp

o = 17 , Gaussian , Comp

o = 20 , Cauchy , No Comp

o = 20 , Cauchy , Comp

o = 20 , Gaussian , No Comp

o = 20 , Gaussian , Comp

Figure 3.11: The autocorrelation plots of the σ_o's for the experiments on English text data, when the length of the preceding sequence $O = 20$. We show the autocorrelation plot of σ_o, for $o = 10, 12, 15, 17, 20$. In the above plots, "Gaussian" in the titles indicates the methods with Gaussian priors, "Cauchy" indicates with Cauchy priors, "comp" indicates with parameters compressed, "no comp" indicates without parameters compressed.

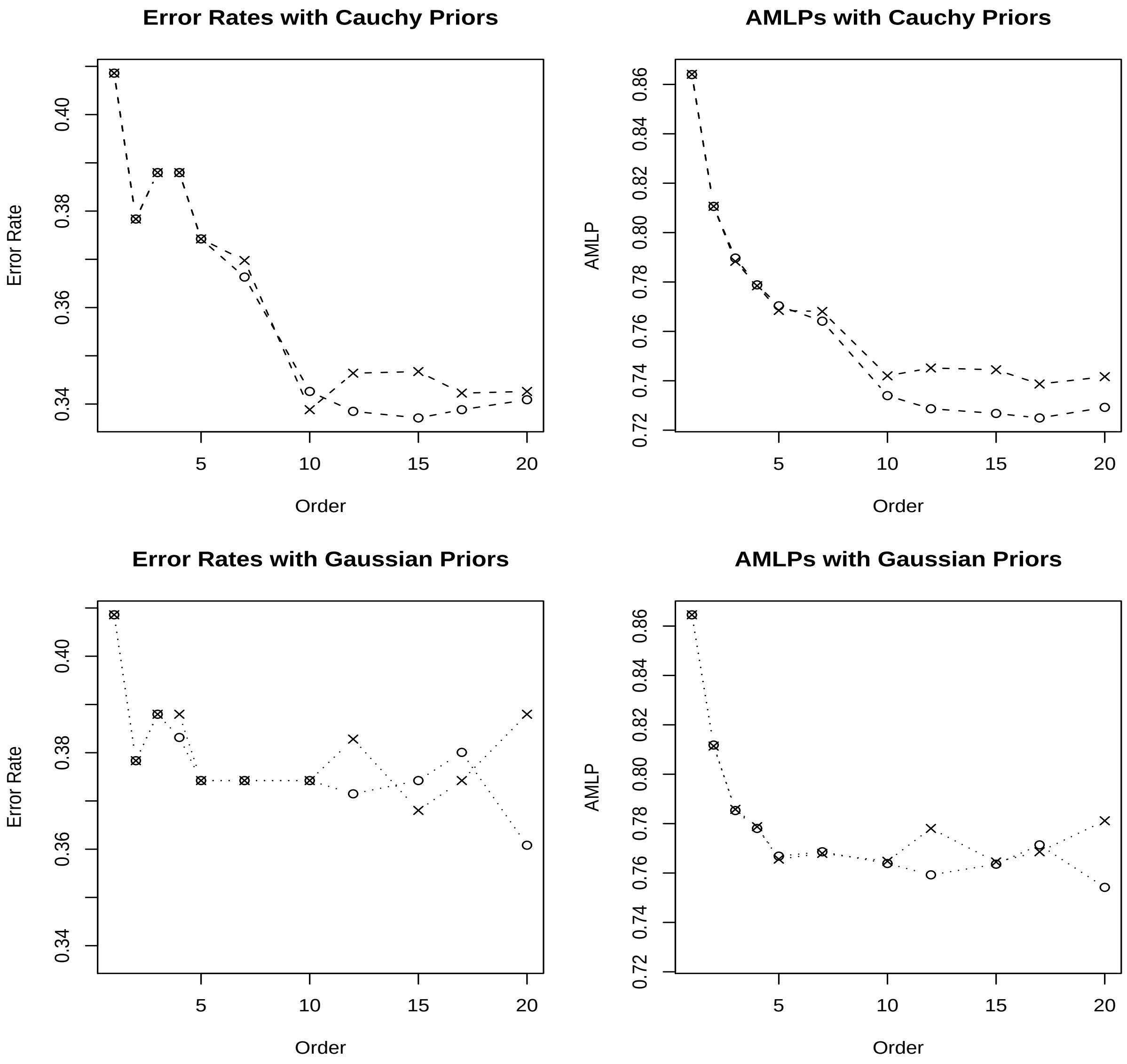


Figure 3.12: Plots showing the predictive performance using the experiments on English text data. The left plots show the error rate and the right plots show the average minus log probability of the true response in a test case. The upper plots show the results when using the Cauchy priors and the lower plots shows the results when using the Gaussian priors. In all plots, the lines with ∘ are for the methods with parameters compressed, the lines with × are for the methods without parameters compressed. The numbers of the training and test cases are respectively 1000 and 2910. The number of classes of the response is 3.

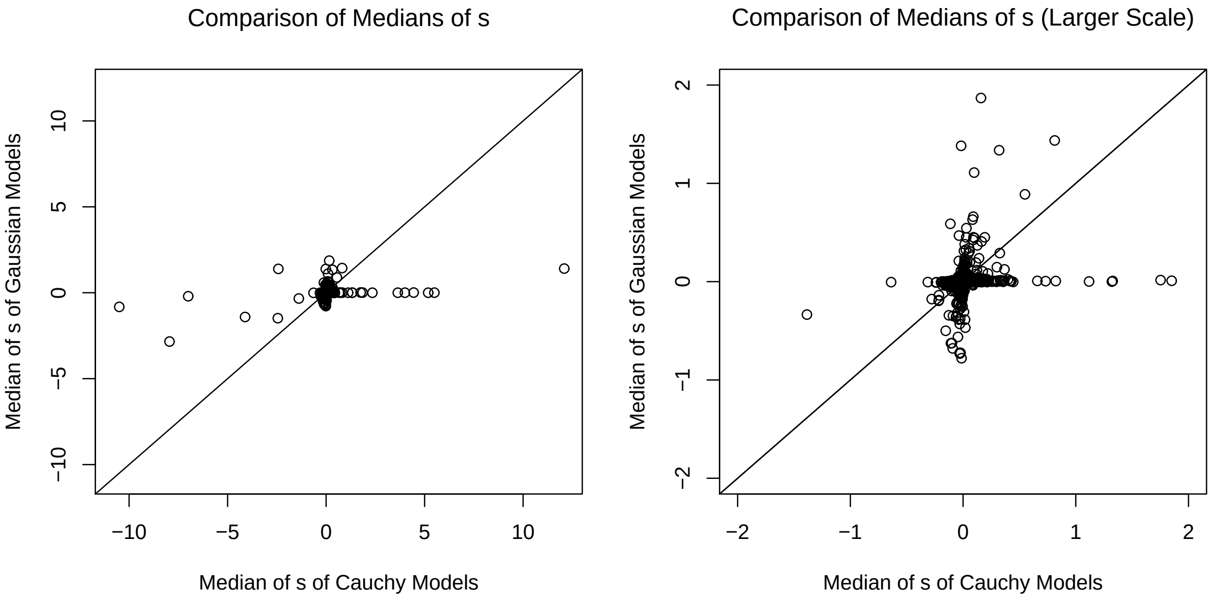


Figure 3.13: Scatterplots of medians of all compressed parameters, s, of Markov chain samples in the last 1250 iterations, for the models with Cauchy and Gaussian priors, fitted with English text data, with the length of preceding sequence $O = 10$, and with the parameters compressed. The right plot shows in a larger scale the rectangle $(-2, 2) \times (-2, 2)$.

method reduces greatly the number of parameters, and therefore shortens the training process greatly. The quality of Markov chain sampling is improved by compressing parameters. Prediction is very fast using our splitting methods. The predictions on the test cases are improved by considering higher order interactions. From Figure 3.12, at least some order 10 interactions are useful in predicting the next character.

In this example we also see that when Cauchy priors are used Markov chain sampling with the original parameters may have been trapped in a local mode, resulting in slightly worse predictions on test cases than with the compressed parameters, even though the models used are identical.

We also see that the models with Cauchy priors result in better predictions than those with Gaussian priors for this data set, as seen from the plots of error rates and AMLPs. To investigate the difference of using Gaussian and Cauchy priors, we first plotted the medians

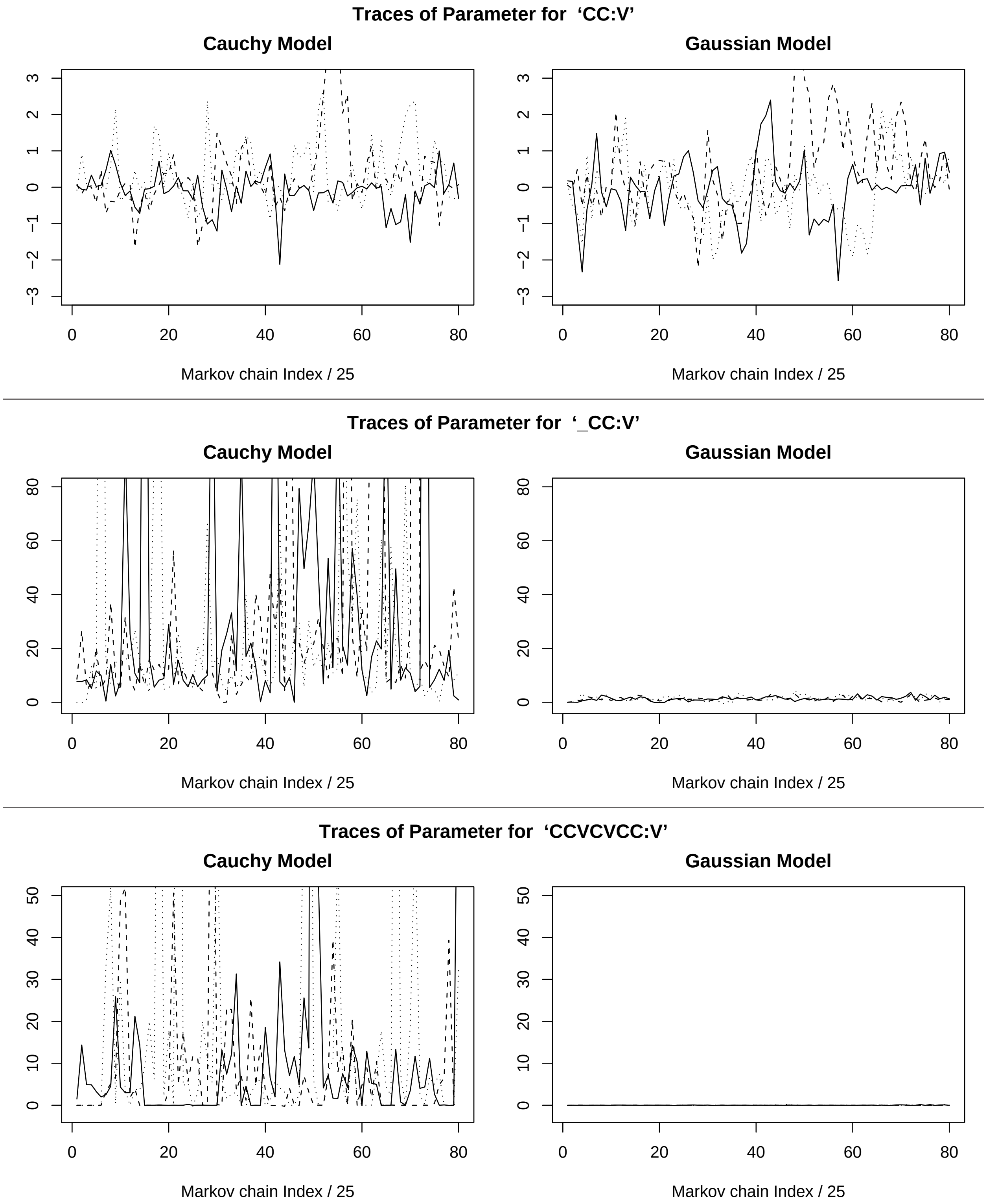


Figure 3.14: Plots of Markov chain traces of three compressed parameters (each contains only one β) from experiments on English text with 10 preceding states, with Cauchy or Gaussian priors. In each plot, three different lines show three indepedent runs. The parameters are annotated by their original meanings in English sequence. For example, '__CC:V' stands for the parameter for predicting that the next character is a "vowel" given preceding three characters are "space, consonant, consonant".

of Markov chains samples (in the last 1250 iterations) of all compressed parameters, s, for the model with $O = 10$, shown in Figure 3.13, where the right plot shows in a larger scale the rectangle $(-2, 2) \times (-2, 2)$. This figure shows that a few β with large medians in the Cauchy model have very small corresponding medians in the Gaussian model.

We also looked at the traces of some compressed parameters, as shown in Figure 3.14. The three compressed parameters shown all contain only a single β. The plots on the top are for the β for “CC:V”, used for predicting whether the next character is a vowel given the preceding two characters are consonants; the plots in the middle are for “␣CC:V”, where “␣” denotes a space or special symbol; the plots on the bottom are for “CCVCVCC:V”, which had the largest median among all compressed parameters in the Cauchy model, as shown by Figure 3.13. The regression coefficient β for “CC:V” should be close to 0 by our common sense, since two consonants can be followed by any of three types of characters. We can very commonly see “CCV”, such as “the”, and “CC␣”, such as “with␣”, and not uncommonly see “CCC”, such as “technique”,“world”, etc. The Markov chain trace of this β with a Cauchy prior moves in a smaller region around 0 than with a Gaussian prior. But if we look back one more character, things are different. The regression coefficient β for “␣CC:V” is fairly large, which is not surprising. The two consonants in “␣CC:V” stand for two letters in the beginning of a word. We rarely see a word starting with three consonants or a word consisting of only two consonants. The posterior distribution of this θ for both Cauchy and Gaussian models favor positive values, but the Markov chain trace for the Cauchy model can move to much larger values than for the Gaussian model. As for the high-order pattern “CCVCVCC”, it matches words like “statistics” or “statistical”, which repeatedly appear in an article introducing a statistics department. Again, the Markov chain trace of this β for the Cauchy model can move to much larger values than for Gaussian model, but sometimes it is close to 0, indicating that there might be two modes for its posterior distribution.

The above investigation reveals that a Cauchy model allows some useful β to be much larger in absolute value than others while keeping the useless β in a smaller region around 0 than a Gaussian model. In other words, Cauchy models are more powerful in finding the information from the many possible high-order interactions than Gaussian models, due to the heavy two-sided tails of Cauchy distributions.

3.5 Application to Logistic Classification Models

3.5.1 Grouping Parameters of Classification Models

As we have seen in sequence prediction models, the regression coefficients for the patterns that are expressed by the same training cases can be compressed into a single parameter. We present an algorithm to find such groups of patterns. Our algorithm may not the only one possible and may not be the best.

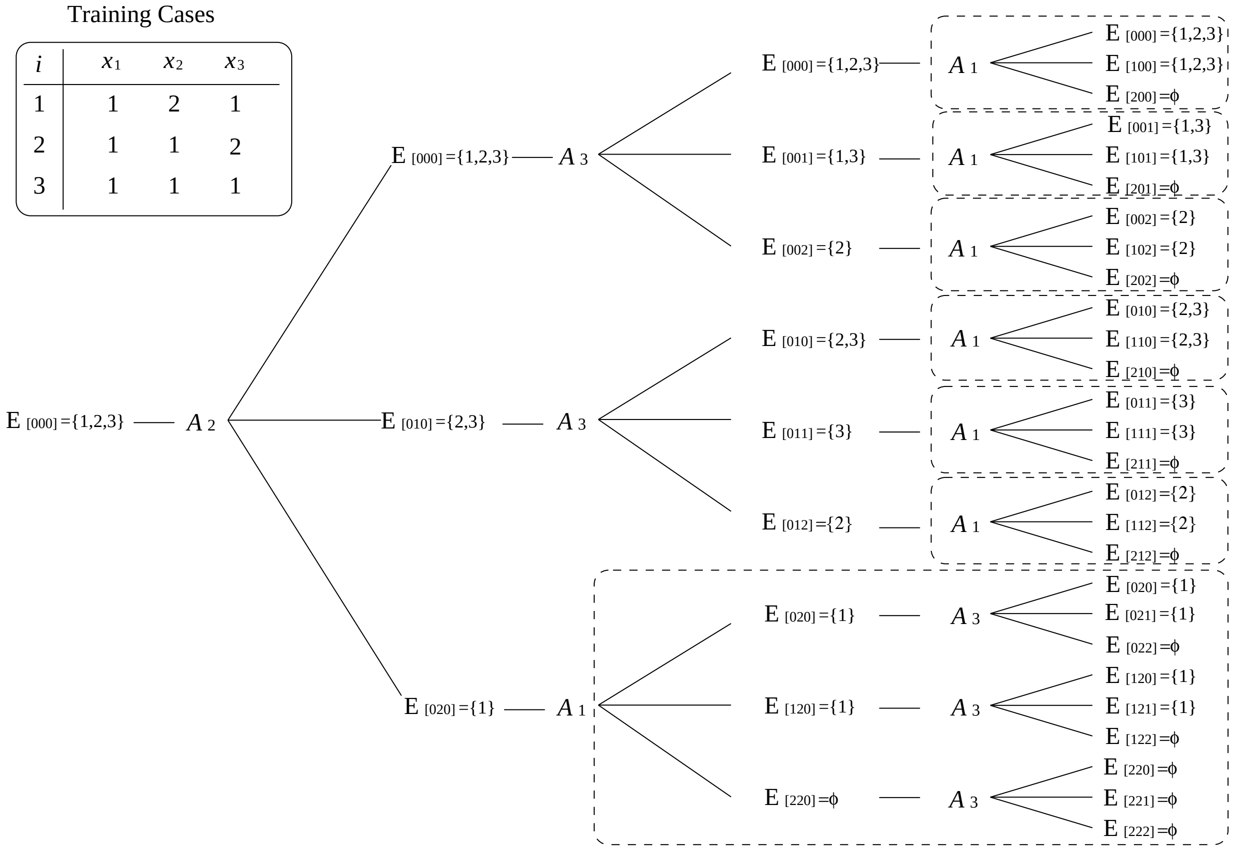
Our algorithm uses a "superpattern", SP, to represent a set of patterns with some common property, written as $\left(\begin{array}{c} A_1 \ldots A_p \\ I_1 \ldots I_p \end{array} \right)^o_f$, where A_t is the pattern value for position t (an integer from 0 to K_t), I_t is binary (0/1) with 1 indicating A_t is fixed for this superpattern, f is the number of fixed positions (ie $f = \sum_{t=1}^{p} I_t$), and o indicates the smallest order of all patterns in this superpattern, equal to the sum of nonzero values of those A_t with $I_t = 1$ (i.e. $o = \sum_{t=1} I(I_t = 1, A_t \neq 0)$). Such a superpattern represents the union of all patterns with order not greater than O, with values at the fixed positions (with $I_t = 1$) being A_t, but the pattern values at unfixed positions (with $I_t = 0$) being either 0 or A_t. For example, if $O = 3$, the superpattern $\left(\begin{array}{c} 12304 \\ 10010 \end{array} \right)^1_2$ is composed of $\left(\begin{array}{c} 3 \\ 0 \end{array} \right)$, $\left(\begin{array}{c} 3 \\ 1 \end{array} \right)$, and $\left(\begin{array}{c} 3 \\ 2 \end{array} \right)$ patterns respectively of order $1, 2$ and 3, listed as follows:

$$\begin{aligned} &\text{order } 1: \quad [10000] \\ &\text{order } 2: \quad [12000], \quad [10300], \quad [10004] \\ &\text{order } 3: \quad [10304], \quad [12004], \quad [12300] \end{aligned} \tag{3.39}$$

The algorithm is inspired by the display of all interaction patterns in Figure 3.2. It starts with the expression $\{1,\ldots,N\}$ for the superpattern $\left(\begin{array}{c} 00\ldots0 \\ 11\ldots1 \end{array}\right)_p^0$, and the unconsidered features $(1,\ldots,p)$. After choosing a feature x_t from the unconsidered features by the way described below, the current expression is split for each value of the x_t, as done by the algorithm for sequence prediction models, additionally the whole expression is also passed down for the pattern with $A_t = 0$. When we see that we can not split the expression by any of the unconsidered features, all the following patterns can be grouped together and represented with a superpattern. In Figure 3.15, we use training data with 3 binary features and only 3 cases to illustrate the algorithm. We give the algorithm in a C-like language in Figure 3.16, which uses a recursive function.

In choosing a feature for splitting a expression, we look at the diversity of the values of the unconsidered features restricted on the current expression. By this way the expression is split into more expressions but each may be smaller. We therefore more rapidly reach the expression that can not be split further. The diversity of a feature x_t restricted on the current expression is measured with the entropy of the relative frequency of the values of x_t, i.e., $-\sum_i p_i \log(p_i)$, where p_i's are the relative frequency of the possible values of the feature restricted on the expression. When two features have the same entropy value, we choose the one with smaller index t. Note that the entropy is always positive unless all the values of the feature x_t restricted on the expression are the same. The resulting expressions found by this algorithm are not unique for each superpattern.

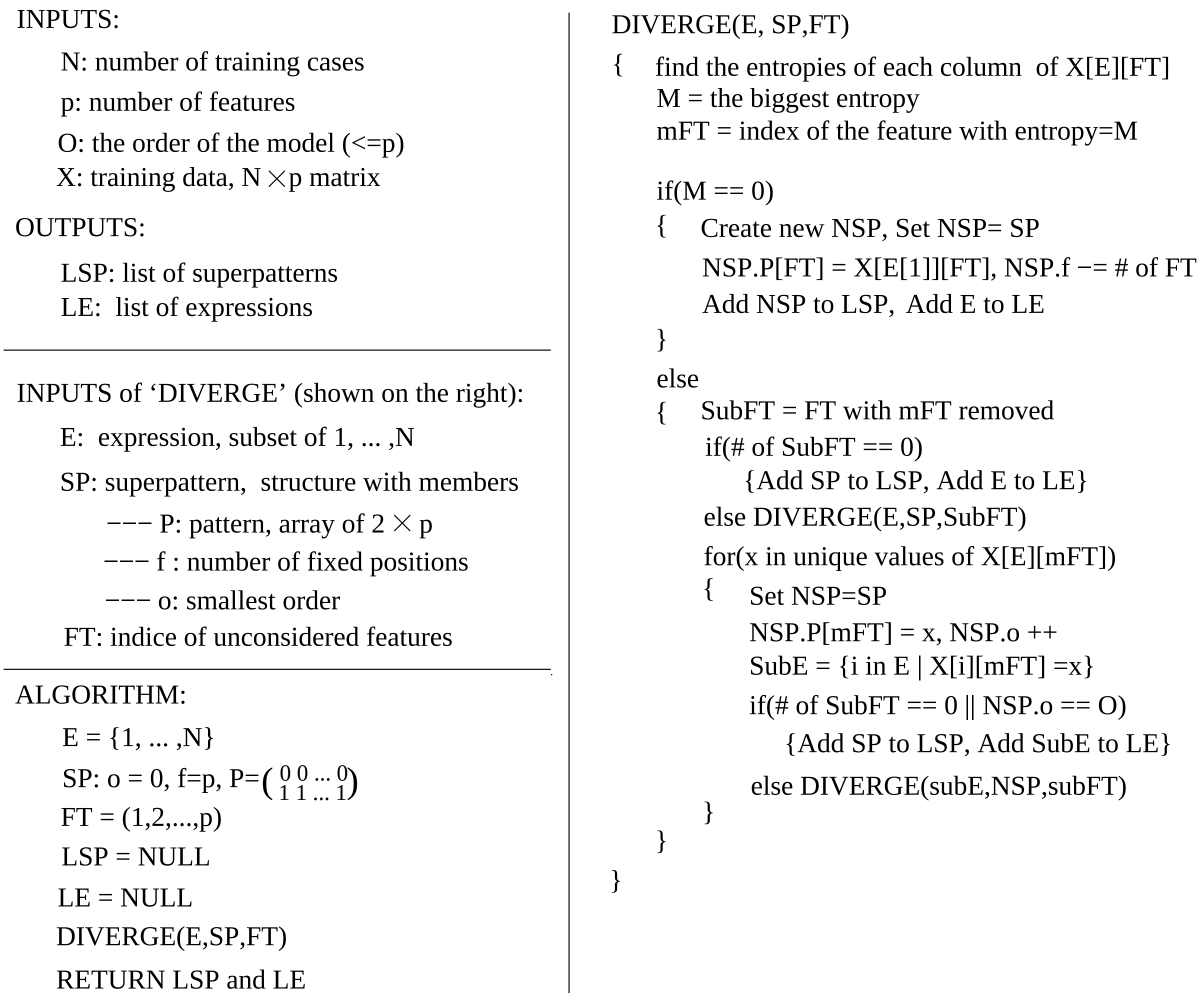
In training the model, we need to compute the width of a parameter associated with a superpattern given the values of the hyperparameters σ_o's. For Cauchy models, the width is equal to the sum of the hyperparameters of all patterns in the superpattern. For Gaussian models, the width is the square roots of the sum of the squares of the hyperparameters of all patterns in the superpattern. We therefore need only know the number of the patterns in the superpattern belonging to each order from 0 to O. For a superpattern $\left(\begin{array}{c} A_1\ldots A_p \\ I_1\ldots I_p \end{array}\right)_f^o$,



Training Cases

i	x_1	x_2	x_3
1	1	2	1
2	1	1	2
3	1	1	1

Figure 3.15: This picture illustrates the algorithm for grouping the patterns of classification models using a training data of 3 binary features and 3 cases shown on the left-top corner. Starting from the expression for the intercept and all features in the unconsidered features set, we recursively split the current expression by the values of the feature with the biggest entropy (diversity) among the remaining unconsidered features. When the values of all remaining unconsidered features are the same for all training cases, for example, when the expression contains only one training case, all the following patterns can be grouped together. After grouping, all the expressions in dashed box are removed and grouped into their parent expressions, with the group of patterns being represented using a superpattern.



```
INPUTS:
    N: number of training cases
    p: number of features
    O: the order of the model (<=p)
    X: training data, N ×p matrix
OUTPUTS:
    LSP: list of superpatterns
    LE:  list of expressions
--------------------------------------------------
INPUTS of ‘DIVERGE’ (shown on the right):
    E:  expression, subset of 1, ... ,N
    SP: superpattern,  structure with members
        −−− P: pattern, array of 2 × p
        −−− f : number of fixed positions
        −−− o: smallest order
    FT: indice of unconsidered features
--------------------------------------------------
ALGORITHM:
    E = {1, ... ,N}
    SP: o = 0, f=p, P=$\binom{0\ 0\ ...\ 0}{1\ 1\ ...\ 1}$
    FT = (1,2,...,p)
    LSP = NULL
    LE = NULL
    DIVERGE(E,SP,FT)
    RETURN LSP and LE
```

```
DIVERGE(E, SP,FT)
{   find the entropies of each column  of X[E][FT]
    M = the biggest entropy
    mFT = index of the feature with entropy=M

    if(M == 0)
    {   Create new NSP, Set NSP= SP
        NSP.P[FT] = X[E[1]][FT], NSP.f −= # of FT
        Add NSP to LSP,  Add E to LE
    }
    else
    {   SubFT = FT with mFT removed
        if(# of SubFT == 0)
            {Add SP to LSP, Add E to LE}
        else DIVERGE(E,SP,SubFT)
        for(x in unique values of X[E][mFT])
        {   Set NSP=SP
            NSP.P[mFT] = x, NSP.o ++
            SubE = {i in E | X[i][mFT] =x}
            if(# of SubFT == 0 || NSP.o == O)
                {Add SP to LSP, Add SubE to LE}
            else DIVERGE(subE,NSP,subFT)
        }
    }
}
```

Figure 3.16: The algorithm for grouping the patterns of Bayesian logistic classification models. To do grouping, we call the function “DIVERGE” with the initial values of expression E, superpattern SP and the unconsidered features FT shown as above, resulting in two lists of the same length, LE and LSP, respectively storing the expressions and the corresponding superpatterns. Note that the first index of an array are assumed to be 1, and that$X[E][FT]$ means sub matrix of X by restricting the rows in E and columns in FT. 3) The resulting expressions are not unique for each superpattern in LSP. We still need to merge those superpatterns with the same expression by directly comparing the expressions.

they are given as follows:

$$\#(\text{patterns of order } o+d) = \begin{cases} \left(\begin{array}{c} p-f \\ d \end{array} \right), & \text{for } d = 0, \ldots, \min(O-o, p-f) \\ 0, & \text{Otherwise} \end{cases} \tag{3.40}$$

In predicting for a test case $\boldsymbol{x}^*$, we need to identify the patterns in a superpattern $\left(\begin{array}{c} A_1 \ldots A_p \\ I_1 \ldots I_p \end{array} \right)_f^o$ that is also expressed by $\boldsymbol{x}^*$. If $A_t \neq x_t^*$ for any t with $I_t = 1$ and $A_t \neq 0$, none of the patterns in the superpatterns are expressed by $\boldsymbol{x}^*$. Otherwise, if $x_t^* \neq A_t$ for any unfixed positions (with $I_t = 0$) then the A_t is set to 0. This results in a smaller superpattern, from which we can count the number of patterns belonging to each order using the formula in (3.40).

For a fixed very large number of features, p, and a fixed number of cases, N, the number of the compressed parameters may have converged before considering the highest possible order, p. This can be verified by regarding the interaction patterns of a classification model as the union (non-exclusive) of the interaction patterns of $p!$ sequence models from permutating the indice of features. As shown for sequence models in Section 3.4.1, there is a certain order O_j, for $j = 1, \ldots, p!$, for each of the $p!$ sequence models, the number of superpatterns with unique expressions will not grow after considering order higher than O_j. The number of the superpatterns with unique expressions for all $p!$ sequence models will not grow after we consider the order higher than the maximum value of O_j, for $j = 1, \ldots, p!$. If this maximum value is smaller than p, the number of the *compressed parameters* converges before considering the highest possible order, p. On the contrary, the number of the *original parameters* (the regression coefficients for those interaction patterns expressed by some training case) will keep growing until considering the highest order, p.

3.5.2 Experiments Demonstrating Parameter Reduction

In this section, we use simulated data sets to illustrate our compression method. We apply our compression method to many training data sets with different properties to demonstrate how the rate of parameters reduction and the number of compressed parameters depend on the properties of the data, but without running MCMC to train the models and assessing the predictive performance with test cases.

We generated data sets with varying dimension p, and number of possibilities of each feature, K, the same for all p features. We consider varying order, O. In all datasets, the values of features are drawn uniformly from the set $\{1, \ldots, K\}$, with the number of training cases $N = 200$. We did three experiments, in each of which two of the three values p, K and O are fixed, with the remaining one varied to see how the performance of the compression method changes. The values of p, K and O and the results are shown in Figure 3.17.

From Figure 3.17, we can informally assess the performance of our compression method in different situations. First, when p and O are fixed, as shown by the top plots, the number of compressed parameters decreases with increasing K, but the number of the original parameters does not, showing that our compression method is more useful when K is large, in other words, when K is larger, more patterns that do not need to be represented explicitly will exist. Note, however, that this does not mean the predictive performance for large K is better. On the contrary, when K is larger, each pattern will be expressed by fewer training cases, possibly leading to worse predictive performance. Second, as shown by the middle plots where O and K are fixed, the numbers of both the original parameters and the compressed parameters increase very quickly with p, but their ratio decreases with p. In the bottom plots with fixed p and K, the number of the original parameters increases with O but at a much slower rate than with p. The number of compressed parameters have converged when $O = 4$, earlier than $O = 7$.

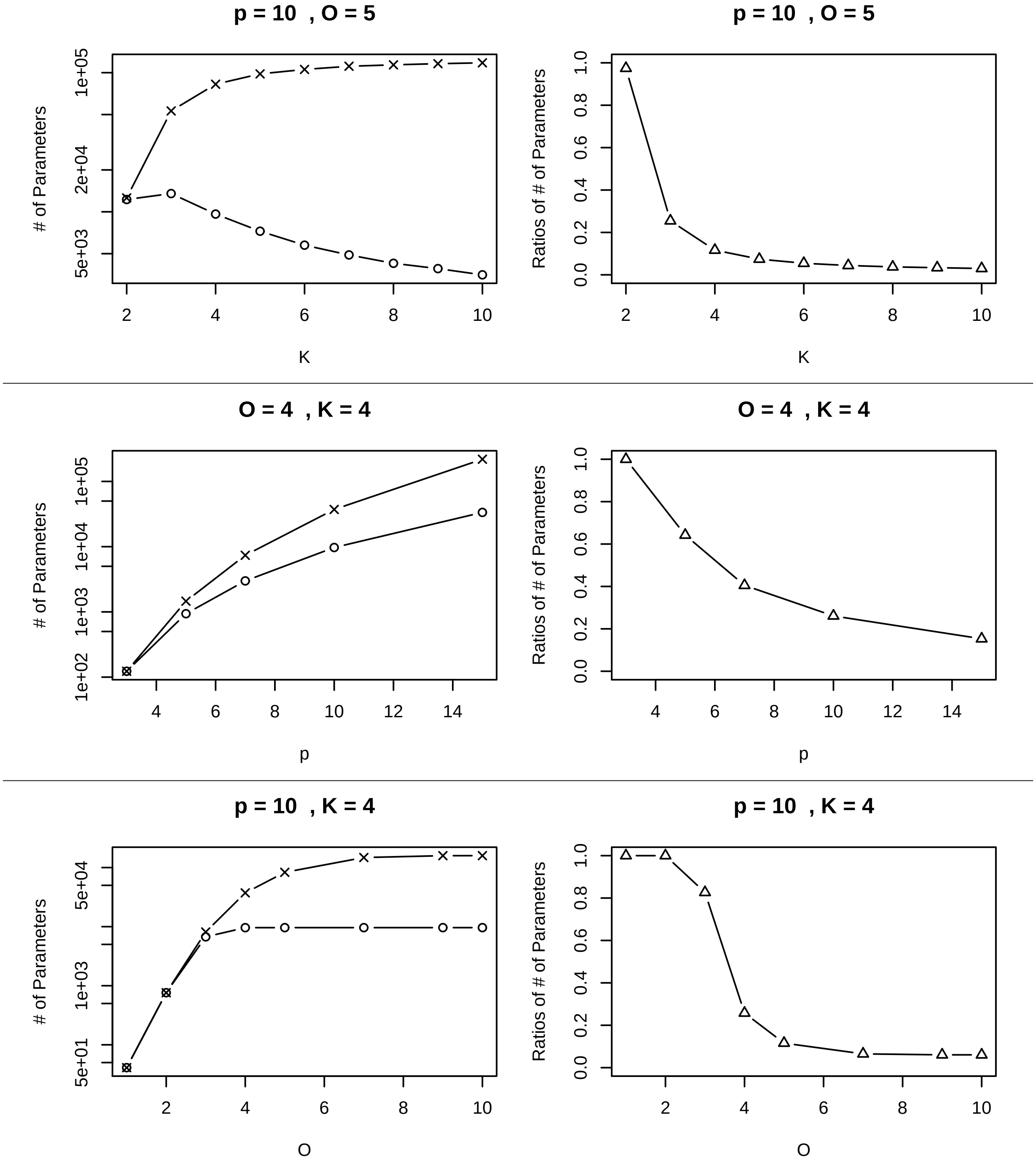


Figure 3.17: Plots of the number of the compressed parameters (the lines with ∘) and the original parameters (the lines with ×), in log scale, their ratios (the lines with △) for Bayesian logistic classification models. The number of training cases is 200 for all data sets. The titles and the horizontal axis labels indicates the values of p, K and O in compressing parameters, where p is the number of features, K is the number of possibilities of each feature, and O is the order of interactions considered .

3.5.3 Experiments with Data from Cauchy Models

We tested the Bayesian logistic classification models on a data set generated using the true model defined in Section 3.2.3. The number of features, p, is 7, and each feature was drawn uniformly from $\{1, 2, 3\}$. For generating the responses, we consider only the interactions from order 0 to order 4, we let $\sigma_o = 1/o$, for $o = 1, \ldots, 4$, then generated regression coefficients β's from Cauchy distributions, as shown by (3.3), except fixing the intercept at 0. We generated 5500 cases, of which 500 were used as training cases and the remainder as test cases. We did experiments with and without the parameters compressed, for order $O = 1, \ldots, 7$.

From these experiments, we see that our compression method can reduce greatly the number of parameters and therefore saves a huge amount of time for training the models with MCMC. The number of the compressed parameters does not grow any more after considering $O = 4$, earlier than $O = 7$. After compressing the parameters, the quality of Markov chain sampling is improved, seen from Figure 3.19, where 5 out of 6 experiments show that the autocorrelation of the σ_o decreases more rapidly with lag after compressing parameters. The predictive performance with and without the parameters compressed are similar, with the optimal performance obtained from the Cauchy model with order $O = 4$, as is expected since this is the true model generating the data set. From the left plot of Figure 3.21, we see that a Cauchy model allows some β to be much larger than others, whereas a Gaussian model keeps all of β in a small region. For those truly small β, Cauchy priors can keep them smaller than Gaussian priors can, as shown by the right plot of Figure 3.21.

3.6 Conclusion and Discussion

In this chapter, we have proposed a method to effectively reduce the number of parameters of Bayesian regression and classification models with high-order interactions, using a compressed parameter to represent the sum of all the regression coefficients for the predictor

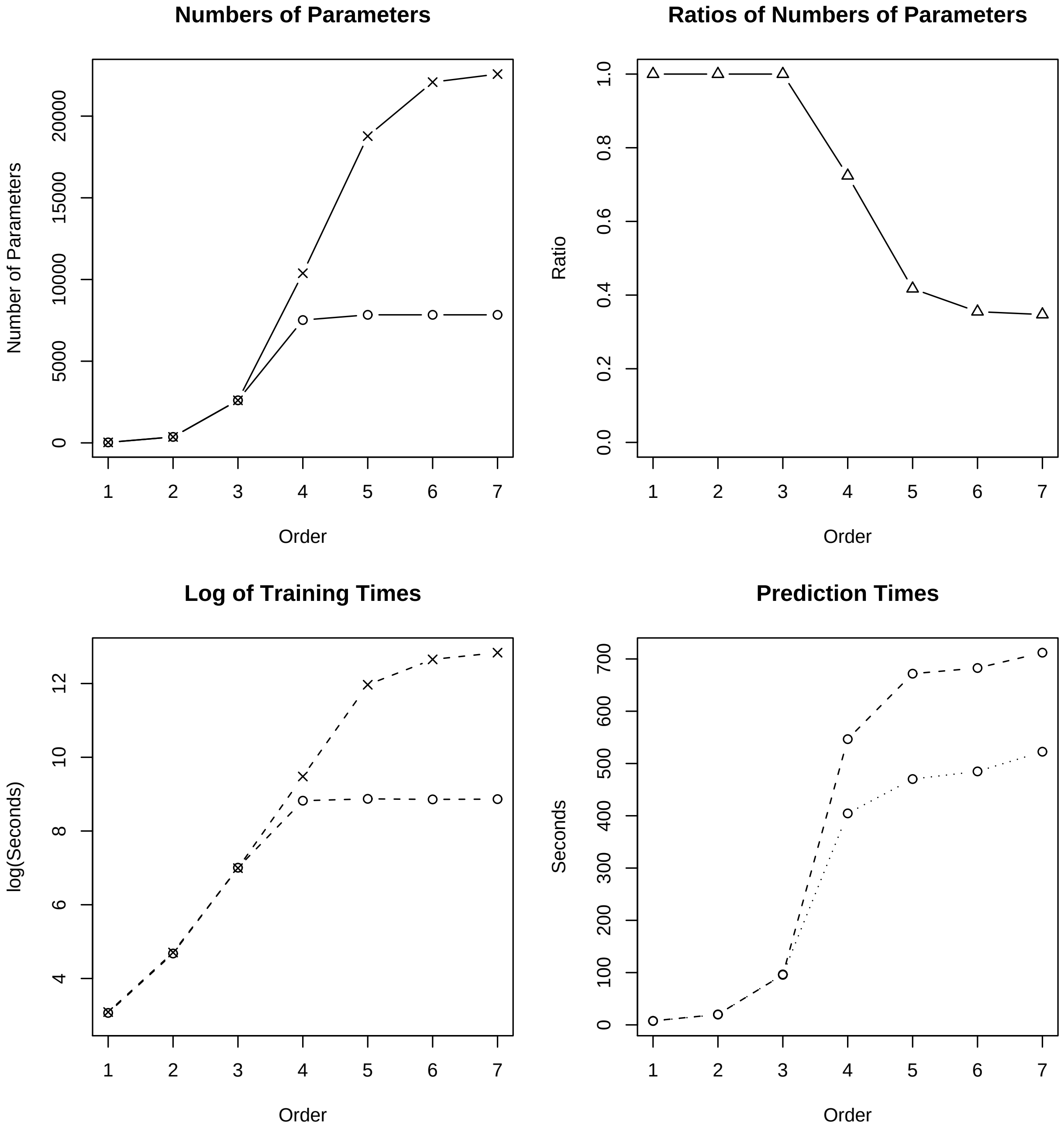


Figure 3.18: Plots showing the reductions of the number of parameters and the training time with our compression method using the experiments on a data from a Cauchy model. The upper-left plot shows the numbers of the compressed and the original parameters based on 500 training sequences for $O = 1, 2, \ldots, 7$, their ratios are shown in the upper-right plot. In the above plots, the lines with $\circ$ are for the methods with parameters compressed, the lines with $\times$ are for the methods without parameters compressed, the dashed lines are for the methods with Gaussian priors, and the dotted lines are for the methods with Cauchy priors. The lower-left plot shows the training times for the methods with and without parameters compressed. The lower-right plot shows the prediction time only for the methods with parameters compressed.

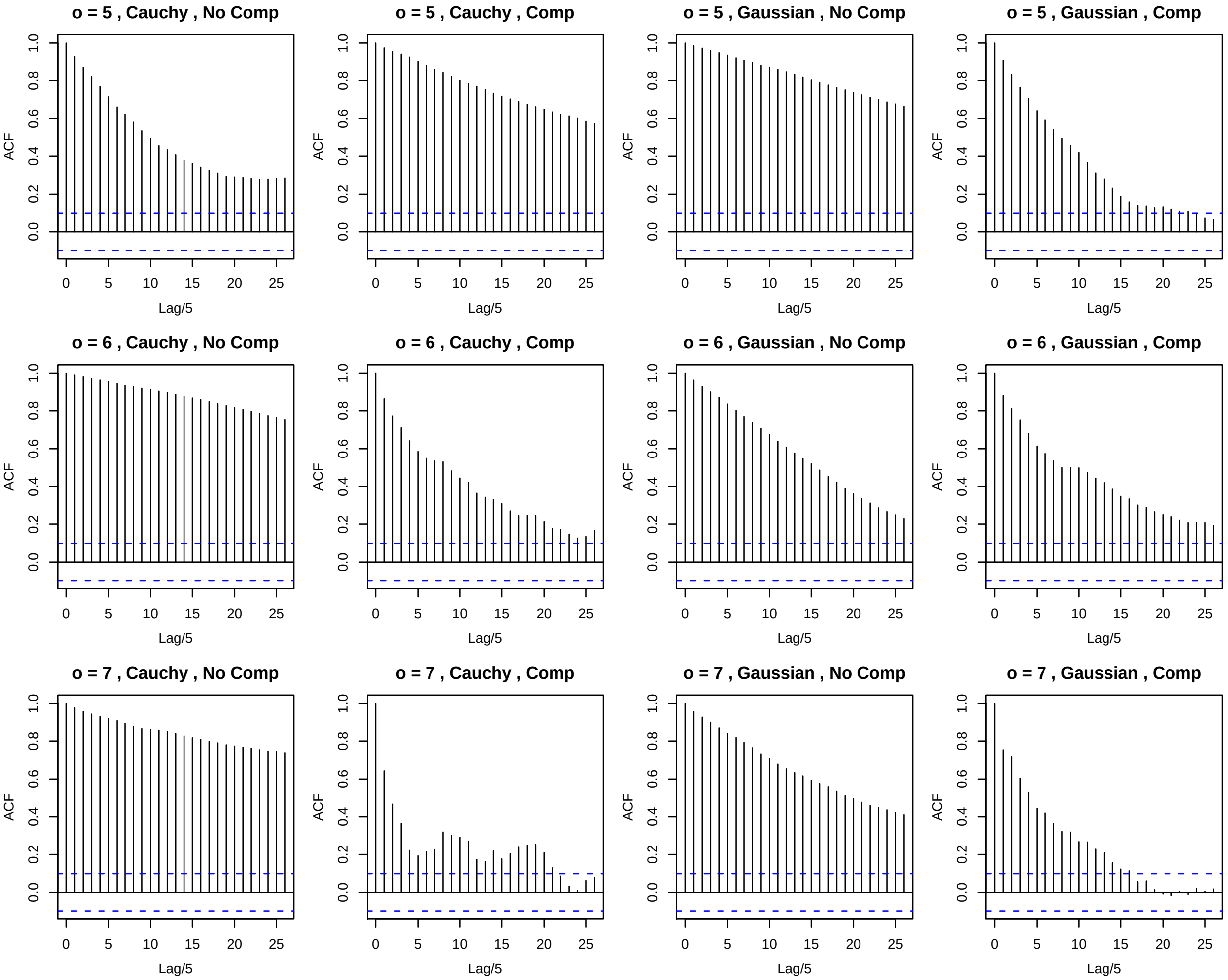


Figure 3.19: The autocorrelation plots of σ_o's for the experiments on a data from a Cauchy model, when the order $O = 7$. We show the autocorrelations of σ_o, for $o = 5, 6, 7$. In the above plots, “Gaussian” in the titles indicates the methods with Gaussian priors, “Cauchy” indicates with Cauchy priors, “comp” indicates with parameters compressed, “no comp” indicates without compressing parameters.

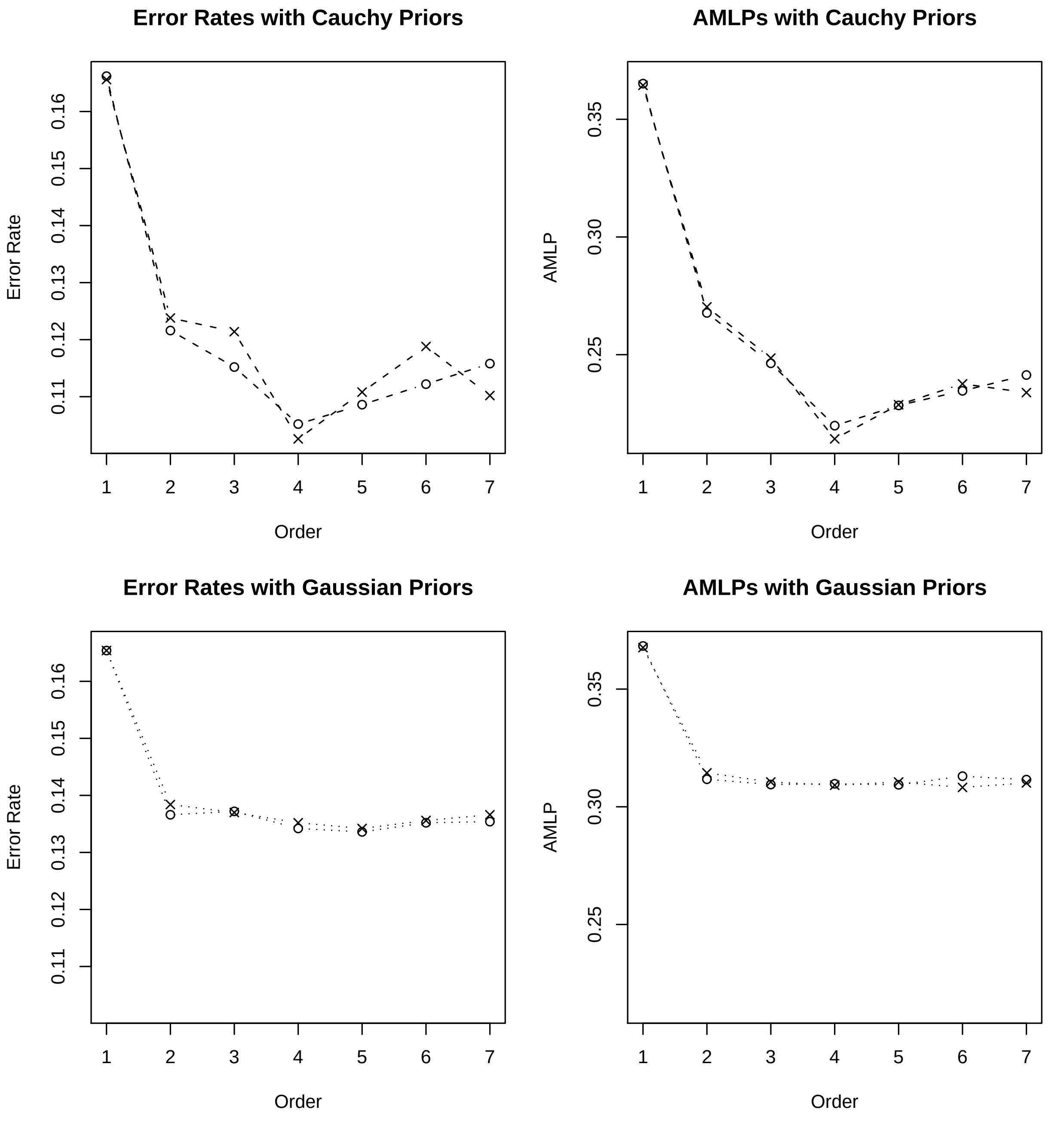


Figure 3.20: Plots showing the predictive performance using the experiments on data from a Cauchy model. The left plots show the error rates and the right plots show the average minus log probabilities of the true responses in the test cases. The upper plots show the results when using the Cauchy priors and the lower plots shows the results when using the Gaussian priors. In all plots, the lines with ∘ are for the methods that compress parameters, the lines with × are for the methods do not compress parameters. The number of the training and test cases are respectively 500 and 5000. The number of classes of the response is 2.

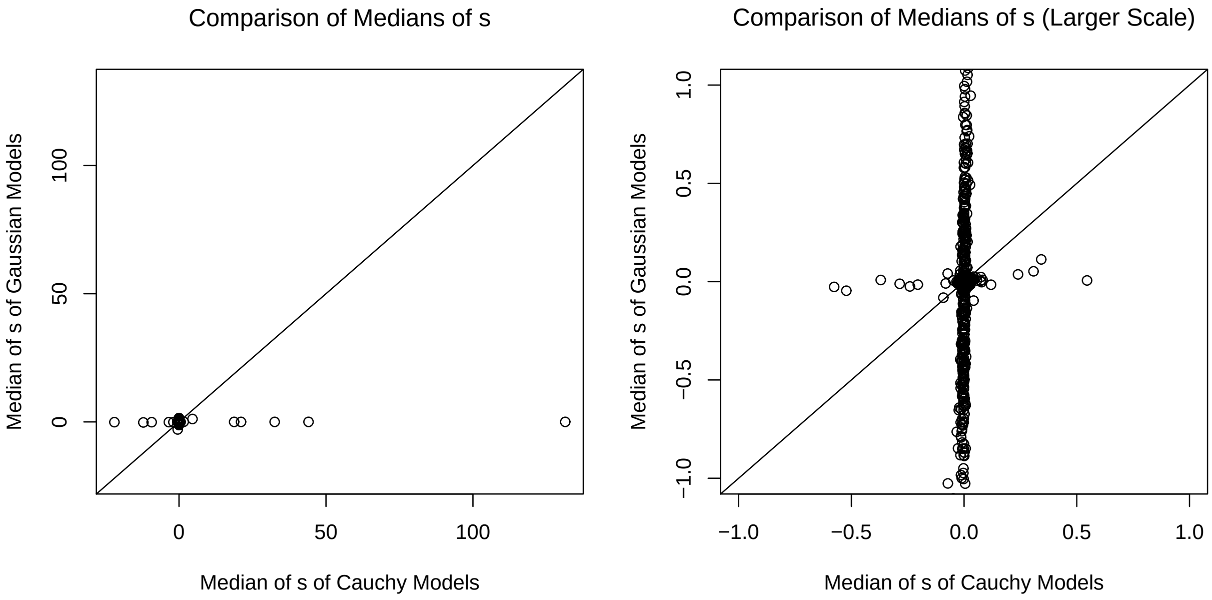


Figure 3.21: Scatterplots of medians of all β of the last 1250 iterations of Markov chain samples, for the models with Cauchy and Gaussian priors, from the experiment with data from a Cauchy model, with the order $O = 4$, and with the parameters compressed. The right plot shows in a larger scale the rectangle $(-1, 1) \times (-1, 1)$.

variables that have the same values for all the training cases. Working with these compressed parameters, we greatly shorten the training time with MCMC. These compressed parameters can later be split into the original parameters efficiently. We have demonstrated, theoretically and empirically, that given a data set with fixed number of cases, the number of compressed parameters will have converged before considering the highest possible order. Applying Bayesian methods to regression and classification models with high-order interactions therefore become much easier after compressing the parameters, as shown by our experiments with simulated and real data. The predictive performance will be improved by considering high-order interactions if some useful high-order interactions do exist in the data.

We have devised schemes for compressing parameters of Bayesian logistic sequence prediction models and Bayesian logistic classification models, as described in Section 3.4 and 3.5.

The algorithm for sequence prediction models is efficient. The resulting groups of interaction patterns have unique expressions. In contrast, the algorithm for classification models works well for problems of moderate size, but will be slow for problems with a large number of cases and with very high order. The resulting groups of interaction patterns may not have unique expressions, requiring extra work to merge the groups with the same expression afterward. A better algorithm that can compress the parameters in shorter time and have a more concise representation of the group of parameters may be found for classification models, though the improvement will not shorten the training time.

We have also empirically demonstrated that Cauchy distributions with location parameter 0, which have heavy two-sided tails, are more appropriate than Gaussian distributions in capturing the prior belief that most of the parameters in a large group are very close to 0 but a few of them may be much larger in absolute value, as we may often think appropriate for the regression coefficients of a high order.

We have implemented the compression method only for classification models in which the response and the features are both discrete. Without any difficulty, the compression method can be used in regression models in which the response is continuous but the features are discrete, for which we need only use another distribution to model the continuous response variable, for example, a Gaussian distribution. Unless one converts the continuous features into discrete values, it is not clear how to apply the method described in this thesis to continuous features. However it seems possible to apply the more general idea that we need to work only with those parameters that matter in likelihood function when training models with MCMC, probably by transforming the original parameters.

We have implemented the compression method only for classification models in which the response and the features are both discrete. Without any difficulty, the compression method can be used in regression models in which the response is continuous but the features are discrete, for which we need only use another distribution to model the continuous response

variable, for example, a Gaussian distribution. Unless one converts the continuous features into discrete values, it is not clear how to apply the method described in this thesis to continuous features. However it seems possible to apply the more general idea that we need to work only with those parameters that matter in likelihood function when training models with MCMC, probably by transforming the original parameters.

We have implemented the compression method only for classification models in which the response and the features are both discrete. Without any difficulty, the compression method can be used in regression models in which the response is continuous but the features are discrete, for which we need only use another distribution to model the continuous response variable, for example, a Gaussian distribution. Unless one converts the continuous features into discrete values, it is not clear how to apply the method described in this thesis to continuous features. However it seems possible to apply the more general idea that we need to work only with those parameters that matter in likelihood function when training models with MCMC, probably by transforming the original parameters.

We have implemented the compression method only for classification models in which the response and the features are both discrete. Without any difficulty, the compression method can be used in regression models in which the response is continuous but the features are discrete, for which we need only use another distribution to model the continuous response variable, for example, a Gaussian distribution. Unless one converts the continuous features into discrete values, it is not clear how to apply the method described in this thesis to continuous features. However it seems possible to apply the more general idea that we need to work only with those parameters that matter in likelihood function when training models with MCMC, probably by transforming the original parameters.

We have implemented the compression method only for classification models in which the response and the features are both discrete. Without any difficulty, the compression method can be used in regression models in which the response is continuous but the features are

discrete, for which we need only use another distribution to model the continuous response variable, for example, a Gaussian distribution. Unless one converts the continuous features into discrete values, it is not clear how to apply the method described in this thesis to continuous features. However it seems possible to apply the more general idea that we need to work only with those parameters that matter in likelihood function when training models with MCMC, probably by transforming the original parameters.

We have implemented the compression method only for classification models in which the response and the features are both discrete. Without any difficulty, the compression method can be used in regression models in which the response is continuous but the features are discrete, for which we need only use another distribution to model the continuous response variable, for example, a Gaussian distribution. Unless one converts the continuous features into discrete values, it is not clear how to apply the method described in this thesis to continuous features. However it seems possible to apply the more general idea that we need to work only with those parameters that matter in likelihood function when training models with MCMC, probably by transforming the original parameters.

We have implemented the compression method only for classification models in which the response and the features are both discrete. Without any difficulty, the compression method can be used in regression models in which the response is continuous but the features are discrete, for which we need only use another distribution to model the continuous response variable, for example, a Gaussian distribution. Unless one converts the continuous features into discrete values, it is not clear how to apply the method described in this thesis to continuous features. However it seems possible to apply the more general idea that we need to work only with those parameters that matter in likelihood function when training models with MCMC, probably by transforming the original parameters.

Bibliography

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999) "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays", *Proceedings of the National Academy of Sciences (USA)*, vol. 96, pp. 6745-6750.

Ambroise, C. and McLachlan, G. J. (2002) "Selection Bias in Gene Extraction on the Basis of Microarray Gene-expression Data", *PNAS*, volumn 99, number 10, pages 6562-6566. Available from http://www.pnas.org/cgi/content/abstract/99/10/6562,

Bishop, C. M. (2006) *Pattern Recognition and Machine Learning*, Springer.

Baker, J. K. (1975). "The Dragon system - an overview", *IEEE Transactions on. Acoustic Speech Signal Processing* ASSP-23(1): 24-29.

Bell, T. C., Cleary, J. G., and Witten, I. H. (1990) *Text Compression*, Prentice-Hall

Cowles, M. K. and Carlin, B. P. (1996) "Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review", *Journal of the American Statistical Association*, Vol. 91, Pages 883–904

Dawid, A. P. (1982) "The well-calibrated Bayesian", *Journal of the American Statistical Association*, vol. 77, no. 379, pp. 605-610.

Everitt, B. S. and Hand, D. J. (1981) *Finite Mixture Distributions*, London: Chapman and Hall.

Eyheramendy, S., Lewis, D. and Madigan, D. (2003) On the Naive Bayes Model for Text Categorization. *Artificial Intelligence and Statistics.*

Feller, W. (1966) "An Introduction to Probability Theory and its Applications", Volume II, New York: John Wiley

Friedman, J.H. (1998) "Regularized Discriminant Analysis", *Journal of the American Statistical Association*, volume 84, number 405, pp. 165–175

Gelfand, A.E. and Smith, A.F.M. (1990) "Sampling-Based Approaches to Calculating Marginal Densities". *Journal of American Statistical Association*, 85:398-409.

Gelman, A., Bois, F. Y., and Jiang, J. (1996) Physiological pharmacokinetic analysis using population modeling and informative prior distributions. *Journal of the American Statistical Association* 91, 1400–1412.

Geman, S. and Geman, D. (1984) "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721-741.

Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A. (2006) *Feature Extraction: Foundations and Applications* (edited volume), Studies in Fuzziness and Soft Computing, Volume 207, Springer.

Hastie, T., Tibshirani, R., and Friedman, J.H. (2001) *The Elements of Statistical Learning*, Springer

Hastings, W.K. (1970) "Monte Carlo Sampling Methods Using Markov Chains and Their Applications", *Biometrika*, Vol. 57, No. 1., pp. 97-109.

Jacques, I. and Judd, C. (1987) *Numerical Analysis*, Chapman and Hall.

Jelinek, F. (1998) *Statistical Methods for Speech Recognition*, The MIT Press. Available from http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=7447

Khan, J., Wei, J.S., Ringnér, M., Saal, L.H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C.R., Peterson, C., (2001) “Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks”, *Nature Medicine*, vol 7, pp. 673–679

Lee, C., Landgrebe, D.A., and National Aeronautics and Space Administration and United States (1993), “Feature Extraction and Classification Algorithms for High Dimensional Data”, *School of Electrical Engineering, Purdue University; National Aeronautics and Space Administration; National Technical Information Service, distributor*

Lecocke, M. L. and Hess K. (2004) “An Empirical Study of Optimism and Selection Bias in Binary Classification with Microarray Data”. UT MD Anderson Cancer Center Department of Biostatistics Working Paper Series. Working Paper 3. Available from http://www.bepress.com/mdandersonbiostat/paper3

Li, L., Zhang, J., and Neal, R. M. (2007) “A Method for Avoiding Bias from Feature Selection with Application to Naive Bayes Classification Models”, *Technical Report No. 0705, Department of Statistics, University of Toronto.*
Available from http://www.cs.toronto.edu/∼radford/selbias.abstract.html.

Li, Y. H. and Jain, A. K. (1998) “Classification of Text Documents”, *The Computer Journal* 41(8): 537-546.

Liu, J. S. (2001) *Monte Carlo Strategies in Scientific Computing*, Springer-Verlag.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E. (1953) “Equation of State Calculations by Fast Computing Machines”, *The Journal of Chemical Physics*, Vol. 21, No. 6, pp. 1087-1092.

McLachlan, G. J. and Basford, K. E. (1988) *Mixture Models: Inference and Applications to Clustering*, New York: Springer-Verlag.

Neal, R. M. (1992) “Bayesian mixture modeling”, in C. R. Smith, G. J. Erickson, and P. O. Neudorfer (editors) *Maximum Entropy and Bayesian Methods: Proceedings of the 11th International Workshop on Maximum Entropy and Bayesian Methods of Statistical Analysis, Seattle, 1991*, p. 197-211, Dordrecht: Kluwer Academic Publishers.

Neal, R. M. (1993) *Probabilistic Inference Using Markov Chain Monte Carlo Methods*, Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 140 pages. Available from `http://www.cs.utoronto.ca/∼radford/`.

Neal, R. M. (1996) *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics No. 118, New York: Springer-Verlag.

Neal, R. M. (2003) “Slice Sampling”, *Annals of Statistics*, vol. 31, p. 705-767

Raudys, S., Baumgartner, R. and Somorjai, R. (2005) “On Understanding and Assessing Feature Selection Bias”, *Artificial Intelligence in Medicine*, Page 468-472, Springer. Available from http://www.springerlink.com/content/8e41e3wncj7yqhx3

Ritchie, M. D., Hahn, L. W., Roodi, N., Bailey, L. R., Dupont,W. D., Parl,F. F., and Moore, J.H. (2001) “Multifactor-Dimensionality Reduction Reveals High-Order Interactions among Estrogen-Metabolism Genes in Sporadic Breast Cancer”, *The American Journal of Human Genetics*, volume 69, pages 138-147

Roberts, G.O., and Rosenthal, J.S. (2004) “General state space Markov chains and MCMC algorithms”, *Probability Surveys* 1:20-71

Romberg, J., Choi, H. and Baraniuk, R. (2001) “Bayesian tree-structured image modeling using wavelet-domain hidden Markov models”, *IEEE Transactions on image processing* 10(7): 1056-1068.

Rosenthal, J. S. (1995) Minorization Conditions and Convergence Rates for Markov Chain Monte Carlo, *Journal of the American Statistical Association* 90:558-566

Singhi, S. K. and Liu, H. (2006) “Feature Subset Selection Bias for Classification Learning”, *Proceedings of the 23rd International Conference on Machine Learning.* Available from http://imls.engr.oregonstate.edu/www/htdocs/proceedings/icml2006/ 107_Feature_Subset_Selec.pdf

Sun, S. (2006) “Haplotype Inference Using a Hidden Markov Model with Efficient Markov Chain Sampling”, PhD Thesis, University of Toronto

Tadjudin, S. and Landgrebe, D. A. (1998) “Classification of High Dimensional Data with Limited Training Samples”, *TR-ECE 98-8, School of Electrical and Computer Engineering Purdue University.*
Available from http://cobweb.ecn.purdue.edu/∼landgreb/Saldju_TR.pdf

Tadjudin, S. and Landgrebe, D. A. (1999) “Covariance estimation with limited training samples”, *Geoscience and Remote Sensing, IEEE Transactions on*, volume 37,number 4, pp. 2113–2118

Tierney, L. (1994). “Markov chains for exploring posterior distributions.” *Annals of Statistics*, 22: 1701-1762.

Titterington, D. M., Smith, A. F. M., and Makov, U. E. (1985) *Statistical Analysis of Finite Mixture Distributions*, Chichester, New York: Wiley.

Thisted, R. A. (1988) *Elements of Statistical Computing*, Chapman and Hall.

Vaithyanathan, S., Mao, J. C., and Dom, B. (2000) “Hierarchical Bayes for Text Classification”, *PRICAI Workshop on Text and Web Mining*, pages 36–43