Bayesian Analysis (2008) **3**, Number 4, pp. 793–822

Compressing Parameters in Bayesian High-order Models with Application to Logistic Sequence Models

Longhai Li* and Radford M. Neal†

Abstract. Bayesian classification and regression with high-order interactions is largely infeasible because Markov chain Monte Carlo (MCMC) would need to be applied with a great many parameters, whose number increases rapidly with the order. In this paper 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. 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 models. Experiments on both simulated and real data demonstrate that a huge number of parameters can indeed be reduced by our compression method.

Keywords: compressing parameters, high-order models, Markov chain Monte Carlo, logistic models, interaction

1 Introduction

In many classification and regression problems, the response variable y depends on high-order interactions of "features" (also called "covariates", "inputs", "predictor variables", or "explanatory variables"). For example, 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 also improved by using a large number of preceding characters (Bell, Cleary, and Witten 1990). Many biological sequences (eg, genomes) have long-memory properties (see eg Durbin et al. 1999, and references therein). Interaction among gene products (under the name of epistasis) seems to be ubiquitous (see eg Cheverud and Routman 1995; Wright 1980, and references therein).

When the features are discrete, we can employ high-order interactions in classi-

*Department of Mathematics and Statistics, University of Saskatchewan, Saskatoon, SK, Canada, mailto:longhai@math.usask.ca

†Department of Statistics, University of Toronto,, Toronto, Ontario, Canada, mailto:radford@utstat.toronto.edu

 DOI:10.1214/08-BA330

fication and regression 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 paper 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 (ie, all possible combinations) of k binary features is 2^k; accordingly we will have 2^k predictor variables. Models with interactions of even a moderate order are infeasible in real applications, primarily for computational reasons. People are often forced to use a model with very small order, say only 1 or 2, even though this may omit useful higher-order predictor variables.

Besides the computational considerations, classification and regression 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 centred 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 interactions of high order are close to 0, with a very small number of exceptions. Additionally, the priors used for the widths of these Gaussian or Cauchy distributions should favor small values for higher order interactions. 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 by inferring unnecessarily complex relationships.

However, the computational difficulty with a huge number of parameters is even more pronounced for a Bayesian approach than other approaches. We will likely need 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 also need more iterations to converge, or get trapped more easily in local modes. Applying Markov chain Monte Carlo methods to classification and regression models with high-order interactions therefore seems infeasible.

In this paper, we show how these problems can be solved by effectively reducing the number of parameters in a Bayesian model with high-order interactions. Our method is based on the fact that in a model using all interaction patterns, from a low order to a high order, there are many sets of predictor variables that have the same values for all training cases. For example, if an interaction pattern occurs in only one training case, that pattern and all interaction patterns of higher order contained in it will occur in only that case, and hence 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. This allows us to 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 interactions in a reasonable amount of time.

This paper will be organized as follows. In Section 2 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. There, we will describe the specific schemes for compressing parameters for the sequence prediction models, and use simulated data and real data to demonstrate our method. We draw conclusions and discuss future work in Section 5.

A software package for the method described in this paper, which is written in R, is available from http://math.usask.ca/~longhai.

2 Our Method for Compressing Parameters

In this section we describe in general terms how to use compressed parameters during MCMC sampling and afterward recover the original parameters as needed for making predictions. We also discuss why this procedure saves computation.

2.1 Compressing Parameters

Our method for compressing parameters is applicable when we can divide the regression coefficients used in the likelihood function into a number of groups such that the likelihood is a function only of the sums over these groups. The groups will depend on the particular training data set. An example of such a group is the regression coefficients for a set of predictor variables that have the same values for all training cases. It may not be easy to find an efficient scheme for grouping the parameters of a specific model. We will describe how to group the parameters for sequence prediction models in Section 3. Suppose the number of such groups is G. The parameters in group g are denoted by $\beta_{g1}, \ldots, \beta_{g,n_g}$, and their sum is denoted by s_g:

$$s_g = \sum_{j=1}^{n_g} \beta_{gj}, \qquad \text{for } g = 1, \ldots, G, \tag{1}$$

We assume that the likelihood function can be written as:

$$\begin{aligned} & L^{\beta}(\beta_{11},\ldots,\beta_{1,n_1},\ \ldots\ ,\beta_{G1},\ldots,\beta_{G,n_G}) \\ = \ & L\left(\sum_{j=1}^{n_1}\beta_{1j},\ \ldots,\ \sum_{j=1}^{n_G}\beta_{Gj}\right) \\ = \ & L(s_1,\ \ldots\ ,s_G). \end{aligned} \tag{2}$$

Note that 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 additional regression coefficients. We will assign prior distributions for all the coefficients assuming that they are mutually independent (possibly conditional on some hyperparameters). The posterior distributions of these additional coefficients are therefore equal to their priors (given the relavant hyperparameters if there are any).

We need to define priors for the β_{gj} in a way that lets us easily find the priors of the s_g. For this purpose, we assign each β_{gj} a symmetric stable distribution centred at 0 with width parameter σ_{gj}. 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 (up to location and width) 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_{gj} has a Gaussian or Cauchy distribution with location parameter 0 and width parameter σ_{gj}, and they are independent given the σ_{gj}'s:

$$\beta_{gj} \sim N(0,\sigma_{gj}^2) \quad \text{or} \quad \beta_{gj} \sim \text{Cauchy}(0,\sigma_{gj}). \tag{3}$$

Some σ_{gj} may be common for different β_{gj}, but for the moment we denote them individually. We might also treat the σ_{gj}'s as unknown hyperparameters, but again we assume them fixed for the moment.

If the prior distributions for the β_{gj}'s are as in (3), the prior distribution of s_g can be found using the property of symmetric stable distributions:

$$s_g \sim N\left(0,\ \sum_{j=1}^{n_g}\sigma_{gj}^2\right) \quad \text{or} \quad s_g \sim \text{Cauchy}\left(0,\ \sum_{j=1}^{n_g}\sigma_{gj}\right). \tag{4}$$

Let us denote the density of s_g in (4) 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:

$$P(\boldsymbol{s} \mid \mathcal{D}) = \frac{1}{c(\mathcal{D})} L(s_1, \ \ldots \ , s_G) \ P_1^s(s_1) \ \cdots \ P_G^s(s_G), \tag{5}$$

where $\mathcal{D}$ denotes the response variables in the training data, and $c(\mathcal{D})$ is the marginal probability or probability density 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 will need to use some Markov chain sampling method to sample for $\boldsymbol{s}$ from distribution (5).

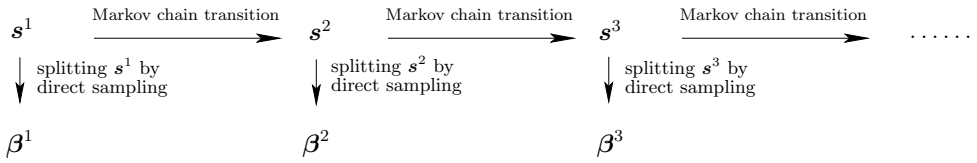
2.2 Splitting Compressed Parameters to Make Predictions

After we have obtained samples of s_g, probably using some Markov chain sampling method, 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. In other words, the splitting distribution is just the conditional distribution of $\beta_{g1}, \ldots, \beta_{g,n_g}$ given $\sum_{j=1}^{n_g} \beta_{gj} = s_g$, whose density function is:

$$P(\beta_{g1}, \ldots, \beta_{g,n_g-1} \mid s_g) \quad = \quad \frac{\left[\prod_{j=1}^{n_g-1} \ P_{gj}(\beta_{gj})\right] \ P_{g,n_g}\left(s_g - \sum_{j=1}^{n_g-1} \beta_{gj}\right)}{P_g^s(s_g)}, \tag{6}$$

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

As will be discussed in the Section 2.4, sampling from (6) 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 $\boldsymbol{s}$. The sampling procedure when using compressed parameters can be diagrammed as follows:



Here, $\boldsymbol{\beta}$ is a collective representation of β_{gj}, for $g = 1, \ldots, G, j = 1, \ldots, n_g - 1$, and the superscripts indicate the index in Markov chain. When we consider high-order interactions, the number of groups, G, will be much smaller than the number of β_{gj}'s. This procedure is therefore much more efficient than applying Markov chain sampling methods to all the original β_{gj} parameters.

If we sampled for all the β_{gj}'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. Furthermore, when making predictions for a particular test case, we actually do not need to sample from the distribution (6), of dimension $n_g - 1$, but only from a derived 1-dimensional distribution.

As will be seen in Section 3, a prediction for a particular test case, for example the probability that this test case is in a certain class, depends only on the sums of subsets of β_{gj}'s in groups. After re-indexing the β_{gj}'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:

$$s_g^t \quad = \quad \sum_{j=1}^{t_g} \beta_{gj} \,, \text{ for } g = 1, \ldots, G, \tag{7}$$

Note that 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 any training case. Suppose these 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 sums of these variables that are needed for making a prediction for the test case are:

$$s_z^* \quad = \quad \sum_{j=1}^{n_z^*} \beta^*_{zj} \,, \text{ for } z = 1, \ldots, Z. \tag{8}$$

In terms of these, the prediction for a test case if the parameters were known can be written as the following function:

$$a\left(\sum_{j=1}^{t_1} \beta_{1j}, \ \ldots \ , \sum_{j=1}^{t_G} \beta_{Gj}, \ \sum_{j=1}^{n_1^*} \beta^*_{1j}, \ \ldots \ , \sum_{j=1}^{n_Z^*} \beta^*_{Zj}\right) \quad = \quad a(s_1^t, \ \ldots \ , s_G^t, s_1^*, \ \ldots \ , s_Z^*). \tag{9}$$

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 based on the posterior parameter distribution is

$$\int \ a(\boldsymbol{s}^t, \boldsymbol{s}^*) \ P(\boldsymbol{s}^*) \ P(\boldsymbol{s} \mid \mathcal{D}) \ \prod_{g=1}^{G} \ P(s_g^t \mid s_g) \ d\boldsymbol{s} \ d\boldsymbol{s}^t d\boldsymbol{s}^*. \tag{10}$$

The integral over $\boldsymbol{s}$ above is done by averaging over iterations from an MCMC run. The integral over $\boldsymbol{s}^*$ is also done by Monte Carlo, sampling from $P(\boldsymbol{s}^*)$, which is the prior distribution of $\boldsymbol{s}^*$. Typically, $P(\boldsymbol{s}^*)$ depends on hyperparameters, which are integrated over by taking them from the same MCMC run. Finally, we need to sample from $P(s_g^t \mid s_g)$, which from (6) is as follows:

$$P(s_g^t \mid s_g) = P_g^{(1)}(s_g^t) \ P_g^{(2)}(s_g - s_g^t) \,/\, P_g^s(s_g). \tag{11}$$

Here, $P_g^{(1)}$ and $P_g^{(2)}$ are the priors (either Gaussian or Cauchy) of $\sum_1^{t_g} \beta_{gj}$ and $\sum_{t_g+1}^{n_g} \beta_{gj}$, respectively.

Before discussing how to sample from (6) or (11), we first phrase this compressing-splitting procedure more formally in the next section to show its correctness.

2.3 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_{gj} to a new set of parameters containing s_g's, as defined in (1), in light of the training data. The posterior distribution (5) of $\boldsymbol{s}$ and the splitting distribution (6) 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 $g = 1, \ldots, G$,

$$\begin{array}{ccc} (\beta_{g1}, \ldots, \beta_{g,n_g-1}, \beta_{g,n_g}) & \Longrightarrow & (\beta_{g1}, \ldots, \beta_{g,n_g-1}, \sum_{j=1}^{n_g} \beta_{gj}) \\ & & \| \\ & & (\beta_{g1}, \ldots, \beta_{g,n_g-1}, s_g) \end{array} \tag{12}$$

In words, the first $n_g - 1$ original parameters β_{gj}'s are mapped to themselves (we might use other symbols to denote the new parameters, but we will use the same ones for simplicity of presentation while making no confusion), and the sum of all $\beta_{g,j}$'s, is mapped to s_g. Let us denote the new parameters β_{gj}, for $g = 1, \ldots, G, j = 1, \ldots, n_g - 1$, 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 $\beta_{g,n_g} = s_g - \sum_{j=1}^{n_g-1} \beta_{gj}$ to obtain the samples of β_{g,n_g}.)

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

$$P(\beta_{11}, \ \ldots \ , \beta_{G,n_G} \mid \mathcal{D}) \ = \ \frac{1}{c(\mathcal{D})} L \left(\sum_{j=1}^{n_1} \beta_{1j}, \ \ldots, \ \sum_{j=1}^{n_G} \beta_{Gj} \right) \prod_{g=1}^{G} \prod_{j=1}^{n_g} P_{gj}(\beta_{gj}). \tag{13}$$

By applying the standard formula for the density function of transformed random variables, we can obtain from (13) the posterior distribution of the $\boldsymbol{s}$ and $\boldsymbol{\beta}$:

$$\begin{aligned} & P(\boldsymbol{s}, \boldsymbol{\beta} \mid \mathcal{D}) \\ & = \frac{1}{c(\mathcal{D})} L\,(s_1, \ \ldots, \ s_G) \prod_{g=1}^{G} \left[\prod_{j=1}^{n_g-1} P_{gj}(\beta_{gj}) \right] P_{g,n_g} \left(s_g - \sum_{j=1}^{n_g-1} \beta_{gj} \right) \Big/ |\det(J)|, \end{aligned} \tag{14}$$

where $|\det(J)|$ is absolute value of the determinant of the Jacobian matrix, J, of the mapping (12), which can be shown to be 1.

Using the additive property of symmetric stable distributions, which is stated in section 2.1, we can analytically integrate out $\boldsymbol{\beta}$ in $P(\boldsymbol{s}, \boldsymbol{\beta} \mid \mathcal{D})$, resulting in the marginal

distribution $P(\boldsymbol{s} \mid \mathcal{D})$:

$$\begin{aligned} P(\boldsymbol{s} \mid \mathcal{D}) &= \int P(\boldsymbol{s}, \boldsymbol{\beta} \mid \mathcal{D})\, d\boldsymbol{\beta} \\ &= \frac{1}{c(\mathcal{D})} L\left(s_1, \ldots, s_G\right) \times \\ &\quad \prod_{g=1}^{G} \int \cdots \int \left[\prod_{j=1}^{n_g - 1} P_{gj}(\beta_{gj}) \right] P_{g,n_g} \left(s_g - \sum_{j=1}^{n_g - 1} \beta_{gj} \right) d\beta_{g1} \cdots d\beta_{g,n_g - 1} \quad (15) \\ &= \frac{1}{c(\mathcal{D})} L\left(s_1, \ldots, s_G\right)\ P_1^s(s_1)\ \cdots\ P_G^s(s_G). \quad (16) \end{aligned}$$

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

$$\begin{aligned} P(\boldsymbol{\beta} \mid \boldsymbol{s}, \mathcal{D}) &= P(\boldsymbol{s}, \boldsymbol{\beta} \mid \mathcal{D}) \,/\, P(\boldsymbol{s} \mid \mathcal{D}) \\ &= \prod_{g=1}^{G} \left[\prod_{j=1}^{n_g - 1} P_{gj}(\beta_{gj}) \right] P_{g,n_g} \left(s_g - \sum_{j=1}^{n_g - 1} \beta_{gj} \right) \Big/\ P_g^s(s_g). \quad (17) \end{aligned}$$

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. Note that equation (6) gives this distribution only for one group g. In other words, conditional on $\boldsymbol{s}$, the data $\mathcal{D}$ and $\boldsymbol{\beta}$ are independent. Some authors term this as that $\boldsymbol{s}$ are the *sufficient parameters* to the distribution of data, and that $\boldsymbol{\beta}$ is unidentifiable from the data, also that the data is uninformative for $\boldsymbol{\beta}$ given $\boldsymbol{s}$ (Barankin 1961; Dawid 1979; Poirier 1998). This transformation over parameters facilitates the separation of the sufficient parameters from the unidentifiable parameters. Finally, we want to clarify that the sufficient parameters will change for different data sets, ie, they are not fixed for a model.

In an analogous manner, we can obtain (11), which splits s_g into two components, from (6) — 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.

2.4 Sampling from the Splitting Distribution

In this section, we discuss how to sample from the splitting distribution (11) to make predictions for test cases after we have obtained samples of $s_1, \ldots, s_G$ from a Markov chain simulation.

When the priors for the β_{gj}'s are Gaussian distributions, the distribution (11) is also a Gaussian distribution, given as follows:

$$s_g^t \mid s_g \ \sim\ N\left(s_g \frac{\Sigma_1^2}{\Sigma_1^2 + \Sigma_2^2}\ ,\ \Sigma_1^2 \left(1 \ -\ \frac{\Sigma_1^2}{\Sigma_1^2 + \Sigma_2^2} \right) \right), \quad (18)$$

where $\Sigma_1^2 = \sum_{j=1}^{t_g} \sigma_{gj}^2$ and $\Sigma_2^2 = \sum_{t_g+1}^{n_g} \sigma_{gj}^2$. We can sample from this Gaussian distribution by standard methods.

When we use Cauchy distributions as the priors for the β_{gj}'s, the density function of (11) is:

$$P(s_g^t \mid s_g) = \frac{1}{C} \frac{1}{\Sigma_1^2 + (s_g^t)^2} \frac{1}{\Sigma_2^2 + (s_g^t - s_g)^2}, \tag{19}$$

where $\Sigma_1 = \sum_{j=1}^{t_g} \sigma_{gj}$, $\Sigma_2 = \sum_{t_g+1}^{n_g} \sigma_{gj}$, and C is the normalizing constant given below by (21).

When $s_g = 0$ and $\Sigma_1 = \Sigma_2$, the distribution (19) 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 (19) can be shown to be:

$$\begin{aligned} F(s_g^t;\, s_g, \Sigma_1, \Sigma_2) \;=\; & \frac{1}{C} \left[r \log \left(\frac{(s_g^t)^2 + \Sigma_1^2}{(s_g^t - s_g)^2 + \Sigma_2^2} \right) \right. \\ & + p_0 \left(\arctan\left(\frac{s_g^t}{\Sigma_1} \right) + \frac{\pi}{2} \right) \\ & \left. + p_s \left(\arctan\left(\frac{s_g^t - s_g}{\Sigma_2} \right) + \frac{\pi}{2} \right) \right], \end{aligned} \tag{20}$$

where

$$C = \frac{\pi\,(\Sigma_1 + \Sigma_2)}{\Sigma_1 \Sigma_2\,(s_g^2 + (\Sigma_1 + \Sigma_2)^2)}, \tag{21}$$

$$r = \frac{s_g}{s_g^4 + 2\,(\Sigma_1^2 + \Sigma_2^2)\; s_g^2 + (\Sigma_1^2 - \Sigma_2^2)^2}, \tag{22}$$

$$p_0 = \frac{1}{\Sigma_1} \frac{s_g^2 - (\Sigma_1^2 - \Sigma_2^2)}{s_g^4 + 2\,(\Sigma_1^2 + \Sigma_2^2)\; s_g^2 + (\Sigma_1^2 - \Sigma_2^2)^2}, \tag{23}$$

$$p_s = \frac{1}{\Sigma_2} \frac{s_g^2 + (\Sigma_1^2 - \Sigma_2^2)}{s_g^4 + 2\,(\Sigma_1^2 + \Sigma_2^2)\; s_g^2 + (\Sigma_1^2 - \Sigma_2^2)^2}. \tag{24}$$

When $s_g \neq 0$, the derivation of (20) uses the equations below from (25) to (27) as follows, where $p = (a^2 - c)/b$, $q = b + q$, $r = pc - a^2 q$, and we assume $4c - b^2 > 0$,

$$\frac{1}{x^2 + a^2} \frac{1}{x^2 + bx + c} = \frac{1}{r} \left(\frac{x + p}{x^2 + a^2} - \frac{x + q}{x^2 + bx + c} \right), \tag{25}$$

$$\int_{-\infty}^{x} \frac{u + p}{u^2 + a^2} du = \frac{1}{2} \log(x^2 + a^2) + \frac{p}{a} \arctan\left(\frac{x}{a} \right) + \frac{\pi}{2}, \tag{26}$$

$$\int_{-\infty}^{x} \frac{u + q}{u^2 + bu + c} du = \frac{1}{2} \log(x^2 + bx + c) + \frac{2q - b}{\sqrt{4c - b^2}} \arctan\left(\frac{2x + b}{\sqrt{4c - b^2}} \right) + \frac{\pi}{2}. \tag{27}$$

When $s_g = 0$, the derivation of (20) uses the following equations:

$$\frac{1}{x^2+a^2}\frac{1}{x^2+b^2} = \frac{1}{b^2-a^2}\left(\frac{1}{x^2+a^2} - \frac{1}{x^2+b^2}\right), \tag{28}$$

$$\int_{-\infty}^{x} \frac{1}{u^2+a^2}\,du = \frac{1}{a}\left(\arctan\left(\frac{x}{a}\right) + \frac{\pi}{2}\right). \tag{29}$$

Since we can compute the CDF of (19) with (20) explicitly, we can use the inversion method to sample from (19), with the inverse CDF computed by some numerical method. We chose the Illinois method (Thisted 1988, Page 171), which is robust and fairly fast.

Sampling for $s_1^t, \ldots, s_G^t$ temporarily for each test case may not be desirable when we need to make predictions for a huge number of test cases. Then we can apply the above method splitting 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 Application to Sequence Prediction Models

In this section, we show how to group parameters of logistic sequence prediction models in which states of a sequence are discrete. We will first define this class of models, then present our scheme for grouping the parameters, and finally give details of a suitable MCMC method for inference. Section 4 contains experimental demonstrations of these models and methods.

3.1 Bayesian Logistic Sequence Prediction Models

We write a sequence of length $O+1$ as $x_1, \ldots, x_O, x_{O+1}$, where x_t takes values from 1 to K_t, for $t = 1, \ldots, O$, and x_{O+1} takes values from 1 to K. We call $x_1, \ldots, x_O = \boldsymbol{x}_{1:O}$ a 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, $\beta_{[0 \ldots 0]}$ is the intercept term. A logistic model assigns to 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., $\{\beta_{\mathcal{P}}^{(k)} \mid \mathcal{P} \in \boldsymbol{\mathcal{S}}\}$, collectively as $\boldsymbol{\beta}^{(k)}$. Figure 1 displays $\boldsymbol{\beta}^{(k)}$ for a 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

$$P(x_{O+1} = k \mid \boldsymbol{x}_{1:O}, \boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(K)}) \;=\; \frac{\exp(l\,(\boldsymbol{x}_{1:O}, \boldsymbol{\beta}^{(k)}))}{\sum_{j=1}^{K} \exp(l\,(\boldsymbol{x}_{1:O}, \boldsymbol{\beta}^{(j)}))}, \tag{30}$$

where

$$l\,(\boldsymbol{x}_{1:O}, \boldsymbol{\beta}^{(k)}) \;=\; \sum_{\mathcal{P} \in \boldsymbol{\mathcal{S}}} \beta_{\mathcal{P}}^{(k)}\; I(\boldsymbol{x}_{1:O} \in \mathcal{P}) \;=\; \beta_{[0\cdots 0]}^{(k)} + \sum_{t=1}^{O} \beta_{[0\cdots x_t \cdots x_O]}^{(k)}. \tag{31}$$

In Figure 1, the linear function is shown by the lines from $\beta_{(x_1,x_2,x_3)}$ to $\beta_{(0,0,0)}$ linking the terms of (31).

The prior for each $\beta_{\mathcal{P}}^{(k)}$ is a Gaussian or Cauchy distribution centred 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, allowing their values to be determined by the data. We set the prior for $\log(\sigma_o)$ to be Gaussian, with mean w_o and standard deviation s_o. In summary, the hierarchy of the priors is:

$$\beta_{\mathcal{P}}^{(k)} \mid \sigma_{o(\mathcal{P})} \;\sim\; \text{Cauchy}(0, \sigma_{o(\mathcal{P})}) \;\text{ or }\; N(0, \sigma_{o(\mathcal{P})}^2), \quad \text{for } \mathcal{P} \in \boldsymbol{\mathcal{S}}, \tag{32}$$

$$\log(\sigma_o) \;\sim\; N(w_o, s_o^2), \quad \text{for } o = 0, \ldots, O. \tag{33}$$

Cauchy distributions have heavy two-sided tails. The absolute value of a Cauchy random variable has infinite mean. When a Cauchy distribution with centre 0 and a small width is used as the prior for a group of parameters, such as the β's for interaction patterns with some order in (32), a few parameters may be much larger in absolute value than others in this group. As priors for the coefficients of high-order interaction patterns, Cauchy distributions can therefore express more accurately than 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 $k = 1, \ldots, K$, since only the differences between $\boldsymbol{\beta}^{(k)}$ matter in (30). 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 set $\boldsymbol{\beta}^{(1)}$ to zero, as there is no asymmetry problem.

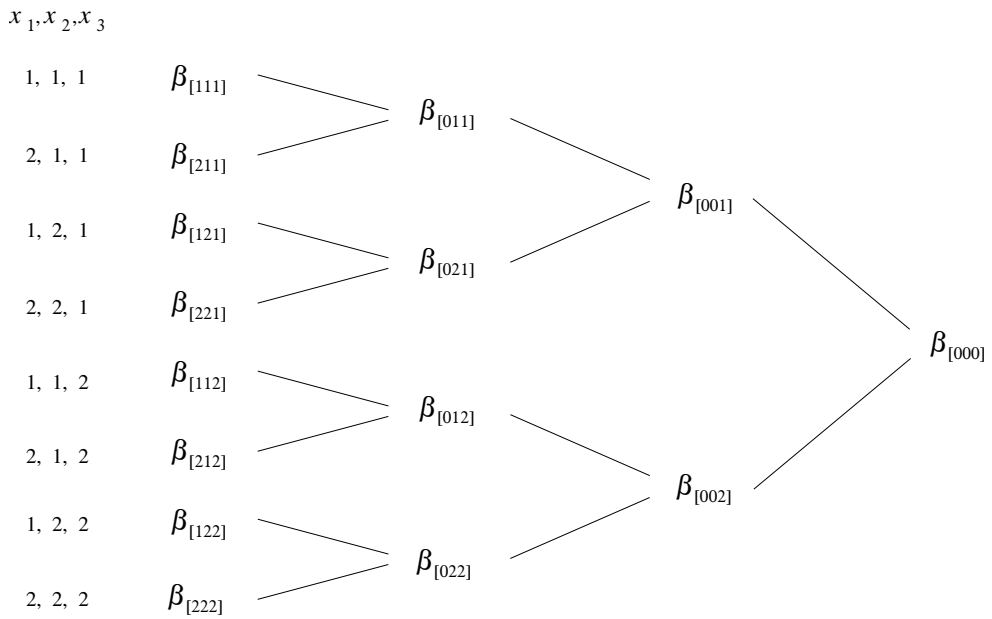


Figure 1: A picture of the coefficients, $\boldsymbol{\beta}$, for all patterns in binary sequences of length $O = 3$. $\beta_{[A_1 A_2 A_3]}$ is associated with the pattern written as $[A_1 A_2 A_3]$, with $A_t = 0$ meaning that x_t is allowed to be either 1 or 2, in other words, x_t is ignored in defining this pattern. For 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 the lines from $\beta_{[x_1 x_2 x_3]}$ to $\beta_{[000]}$.

Inclusion of $\beta_{\mathcal{P}}$ other than the highest order is also a redundancy, which facilitates the expression of appropriate prior beliefs. For similar historic sequences $x_{1:O}$, the prior distributions of $l\left(x_{1:O}, \boldsymbol{\beta}\right)$ are positively correlated, since they share some common β's. For example, in the model displayed by Figure 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+1} 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, thereby avoiding unreasonably extreme conclusions based on a small number of replications. We therefore are not forced to use a model of very low order (i.e., truncating the sequence), since the complexity of the relationship that is inferred will be automatically adjusted by the data.

3.2 Grouping Parameters of Sequence Prediction Models

To apply our method for compressing parameters, we need to divide 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 (2). The likelihood function is the product of prob-

abilities in (30) applied to the training cases, $\boldsymbol{x}^{(i)}_{1:O}, x^{(i)}_{O+1}$, for $i = 1, \ldots, N$ (collectively denoted by $\mathcal{D}$). It can be written as a function of the $\boldsymbol{\beta}^{(k)}$ as follows:

$$L^{\beta}(\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(K)} \mid \mathcal{D}) = \prod_{i=1}^{N} \frac{\exp(l\,(\boldsymbol{x}^{(i)}_{1:O}, \boldsymbol{\beta}^{(x^{(i)}_{O+1})}))}{\sum_{j=1}^{K} \exp(l\,(\boldsymbol{x}^{(i)}_{1:O}, \boldsymbol{\beta}^{(j)}))}. \tag{34}$$

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

Since the linear functions with different k's have the same form except the superscript, the way we divide $\boldsymbol{\beta}^{(k)}$ into groups is the same for all k. In the following discussion, $\boldsymbol{\beta}^{(k)}$ will therefore be written as $\boldsymbol{\beta}$, omitting k.

As shown by (31), 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 the interaction patterns in a group are expressed by the same training cases, the associated β's will appear *simultaneously* in the same factors of (34). The likelihood function (34) therefore depends only on the sum of these β's, rather than the individual ones. Our task is therefore 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, $E_{[0\cdots0]} = \{1, \ldots, N\}$. We can display $E_{\mathcal{P}}$ in a tree-shape, as we displayed $\beta_{\mathcal{P}}$. The upper part of Figure 2 shows such expressions for each pattern in a binary sequence of length $O = 3$, based on the $N = 3$ training cases shown. Note 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 2 can be grouped together. All such patterns must be linked by lines, and can be represented collectively with a "superpattern" SP, written as $[0\cdots0A_b\cdots A_O]_f = \bigcup_{t=b}^{f} [0\cdots0A_t\cdots A_O]$, where $1 \le b \le f \le O+1$. (Note that when $t = O+1$, $[0\cdots0A_t\cdots A_O] = [0\cdots0]$.) One can easily translate the above discussion into a computer algorithm. Figure 3 shows an algorithm for grouping parameters of Bayesian logistic sequence prediction models, using a recursive function.

3.3 Convergence of the Number of Compressed Parameters

With our method for compressing parameters of sequence prediction models, the number of superpatterns with unique expressions, and hence the number of compressed parameters, will converge to a finite number as O increases, even when the past history is infinite. This happens because as we keep splitting the expressions following the tree shown in Figure 2, at a certain time, say t, the expression of every superpattern will

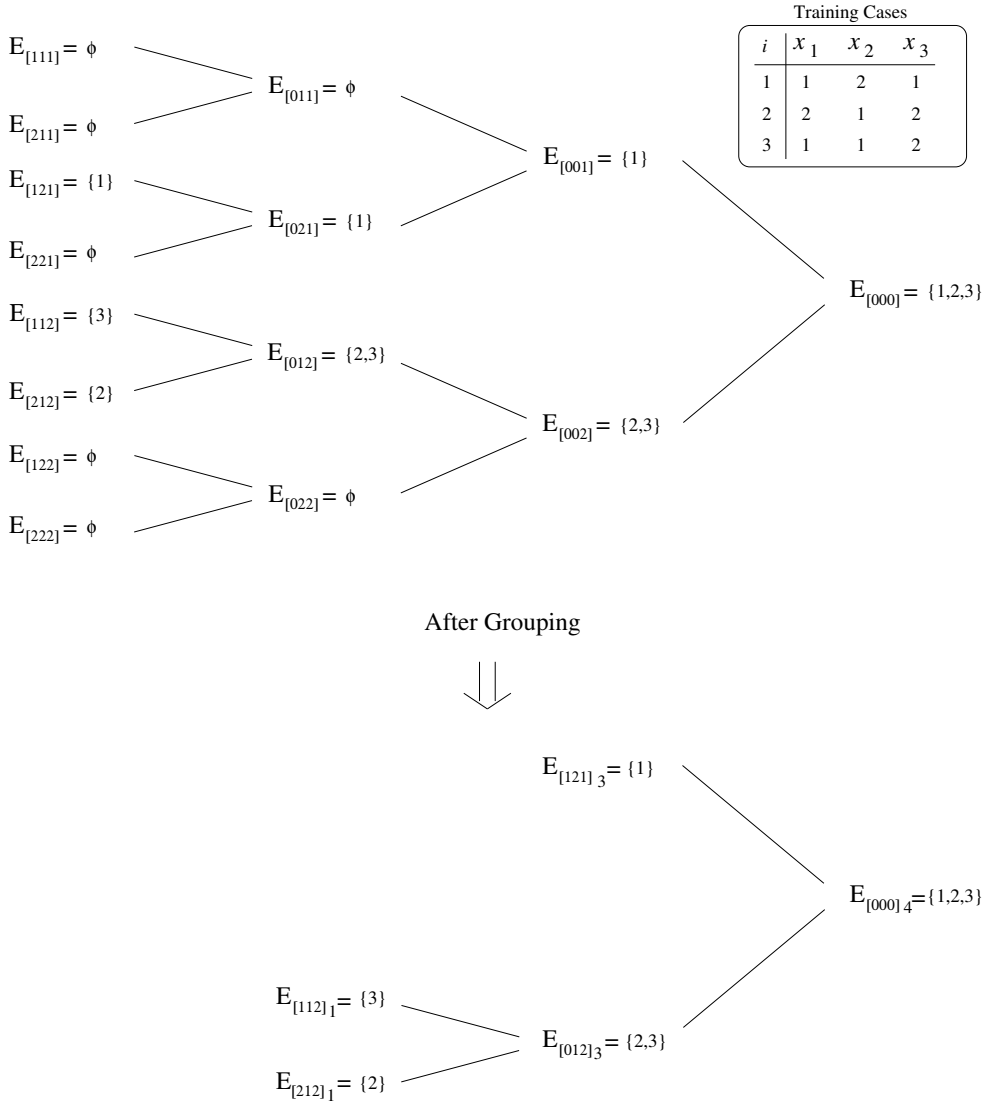


Figure 2: How the interaction patterns in a logistic sequence prediction model can be grouped, illustrated with binary sequences of length $O = 3$, based on the 3 training cases shown in the upper-right box. $E_{\mathcal{P}}$ is the expression of the pattern (or superpattern) $\mathcal{P}$ — the indices of training cases in which $\mathcal{P}$ is expressed, with ϕ being the empty set. Patterns with the same expression are grouped 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 $1 \le b \le f \le O+1$. Patterns not expressed by any of the training cases are removed. 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 X 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 ( b > 0  && # of unique values in  X[E][b] == 1)
    {   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: Algorithm for grouping parameters of Bayesian logistic sequence prediction models. The output is two lists: LSP contains a list of superpatterns, and LE contains the expressions of these superpatterns. The recursive DIVERGE function does the work, taking a subset of training cases (E) and a superpattern (SP) as inputs, and appending new elements to LSP and LE. Note that the index of the first element in a 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.

consist of only a single training case, or a set of identical training cases. When considering times earlier than t, no more new superpatterns with different expressions will be introduced, and the number of superpatterns will not grow. The number of *compressed parameters*, the regression coefficients for the superpatterns, will therefore not grow as O increases to include times before t.

In contrast, if no parameter compression is done, as O increases to include times before this time t, each increase of O by one increases the number of patterns by the number of distinct training cases. The number of regression coefficients associated with these original interaction patterns, called the *original parameters* hereafter, therefore grows linearly with the maximum order. Note that these original parameters do not include 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 maximum order.

3.4 Markov Chain Sampling for the Sequence Prediction Models

Since logistic models are not analytically tractable, we use Markov chain Monte Carlo (Neal 1993, and the references therein) to sample from the posterior distribution of the compressed parameters. Many MCMC methods could be used for this problem. We describe here a simple approach using slice sampling (Neal 2003). We use the same approach to implement the model without parameter compression, for comparison purposes.

Slice sampling uses the fact that one can sample from a one-dimensional distribution with density $f(x)$ by sampling uniformly over the set $\{(x, y) \mid 0 < y < f(x)\}$, i.e., the region of the two-dimensional plane between the x-axis and the curve of $f(x)$. Since it is usually infeasible to sample directly from this region, slice sampling methods use Gibbs sampling (Gelfand and Smith 1990) instead — given x, we draw y from the uniform distribution over $\{y \mid 0 < y < f(x)\}$, then given y, we draw x from the uniform distribution over the "slice", $S = \{x \mid f(x) > y\}$. Directly drawing x uniformly from S is usually infeasible, however, so a Markov chain update is used instead, which leaves this uniform distribution over S invariant. For this problem, we use the "stepping out" plus "shrinkage" procedures described in Neal (2003). 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 steps 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 apportioned for the times of stepping right and stepping left. We then keep drawing a point uniformly from I until we obtain an x in S. To facilitate obtaining an x in S, we shrink the interval I when x is not in S by cutting off the left part or right part of I (depending on whether $x < x_0$ or $x > x_0$). Suitable values for w and m must be chosen, but performance is not highly sensitive to these choices.

We use this slice sampling procedure for each s_g in turn (or for the β_{gj}'s, when not compressing parameters), using the conditional distribution of that s_g given all other parameters and hyperparameters. We then perform some number of slice sampling updates for each of the $\log(\sigma)$ hyperparameters, again based on their conditional distributions given the other parameters.

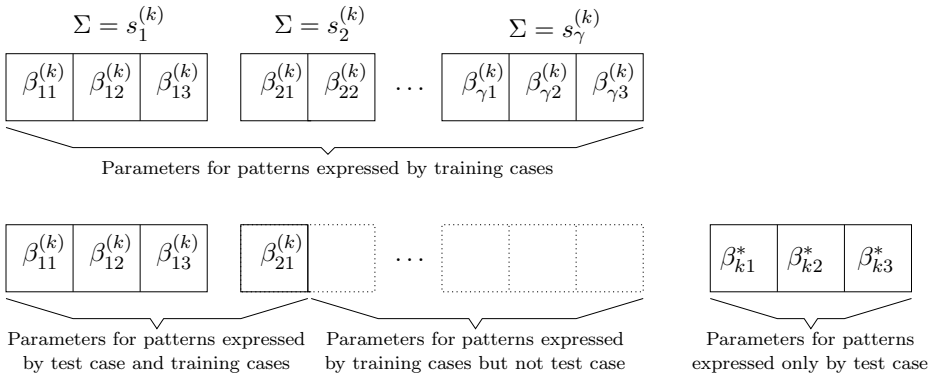
3.5 Making Predictions for Test Cases

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 (30), 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 (30) over 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). The patterns in the test case that are also expressed in one or more training cases are on one path leftwards from the root in the tree

of superpatterns shown in Figure 2. The patterns in a superpattern, $[0 \cdots A_b \cdots A_O]_f$, that are expressed by $\boldsymbol{x}^*_{1:O}$ can be identified as follows. If $(x^*_b, \ldots, x^*_O) = (A_b, \ldots, A_O)$, all the patterns are expressed by $\boldsymbol{x}^*_{1:O}$. This is true for some number of patterns proceeding leftwards from the root. If $(x^*_{b'}, \ldots, x^*_O) = (A_{b'}, \ldots, A_O)$ for some b' with $b < b' < f$, the patterns in $[0 \cdots A_{b'} \cdots A_O]_f$ are expressed by $\boldsymbol{x}^*_{1:O}$. This will be true for one at most one superpattern in the tree, and only for this superpattern will splitting (using (11)) be necessary. Finally, 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 $\boldsymbol{x}^*_{1:O}$.

All of the patterns in the test case of higher order than any contained in a superpattern from the training cases will form a single new superpattern expressed only by the test case. If there are γ superpatterns in the training data, $P(x^*_{O+1} = k \mid \boldsymbol{x}^*_{1:O})$ can be written in the form of (9), with $G = \gamma K$ (though some groups may be empty) and $Z = K$. For prediction in a test case x^*, the parameters for $x^*_{O+1} = k$ are shown below, with dashed boxes standing for unused parameters:



The sum of the β^* parameters above can be drawn from the prior distribution, conditional on the σ_o hyperparameters.

4 Experiments with Logistic Sequence Prediction Models

To demonstrate use of parameter compression with logistic sequence prediction models, we apply them to an artificial binary data set generated using a hidden Markov model, and to a data set created from English text, in which each state has 3 possibilities (consonant, vowel, or other). These experiments will show that our compression method produces a large reduction in the number of parameters needed for training a model of higher order (for which the prediction for the next state of a sequence is based on a long preceding sequence). We also show that good predictions on test cases result from being able to use a high-order model.

4.1 Prior Distributions and Settings of MCMC Parameters

For both sets of experiments, we used prior distributions for parameters and hyperparameters of the form given in (32) and (33). For $o = 0$, we fixed $\log(\sigma_o)$ to 5 in Gaussian models, and to 0.5 in Cauchy models. For $o > 0$, in Gaussian models, we set the Gaussian prior for $\log(\sigma_o)$ to have standard deviations $s_o = 2$, and the means w_o decreasing from 0 to -4 linearly as o increases from 1 to O, in Cauchy models, $s_o = 0.5$, and w_o decreasing from -2 to -6 linearly as o increases from 1 to O.

MCMC was done as described in Section 3.4, with slice sampling for s_g or β, and for $\log(\sigma_o)$. Slice sampling for s_g or β was done with a stepsize of $w = 5$ and a limit on stepping of $m = 10$. Slice sampling for updating $\log(\sigma)$ has $w = 0.5$ and $m = 10$. In all cases, we ran the Markov chain for 4000 iterations (each iteration updating each s_g or β_{gj} parameter once, and each σ_o hyperparameter 50 times). The first 1500 iterations were discarded, and every 10th iteration afterward was used to make predictions for the test cases.

Though these choices were adequate for these experiments, different choices might be necessary for other problems.

4.2 Experiments Using Data from a Hidden Markov Model

In this section, we demonstrate our method for compressing parameters by applying Bayesian logistic sequence prediction models, with or without our compression method, to data sets generated using a Hidden Markov model.

Hidden Markov models (HMM) are applied widely in many areas, for example, speech recognition (Baker 1975), image analysis (Romberg et al. 2001), and computational biology (Sun 2006). In hidden Markov models, the observed sequence, $x_1, x_2, \ldots$, is modeled as being stochastically related to a hidden sequence, $h_1, h_2, \ldots$, that has the Markov property (i.e., given the value of h_{t-1}, h_t is independent of states before h_{t-1}).

Figure 4 displays the hidden Markov model used to generate our data sets, showing the transitions of three successive states. The hidden sequence h_t is a Markov chain with state space $\{1, \ldots, 8\}$, with uniform initial state distribution, and whose dominant transition probabilities (of 0.95) are shown by the arrows in Figure 4. The hidden Markov chain can move from any state to any other state as well, with small probability. 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 observed sequence, $x_1, x_2, x_3, \ldots$, generated by this model exhibits high-order dependency, even though the hidden sequence is a Markov chain. For example, observations of $x_1 = 1$ (rectangle) and $x_2 = 2$ (oval), are 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 be the next hidden state, and the next observation, x_3, is therefore most likely to be 1 (rectangle).

We used the HMM in Figure 4 to generate 5500 sequences of length 21, and used

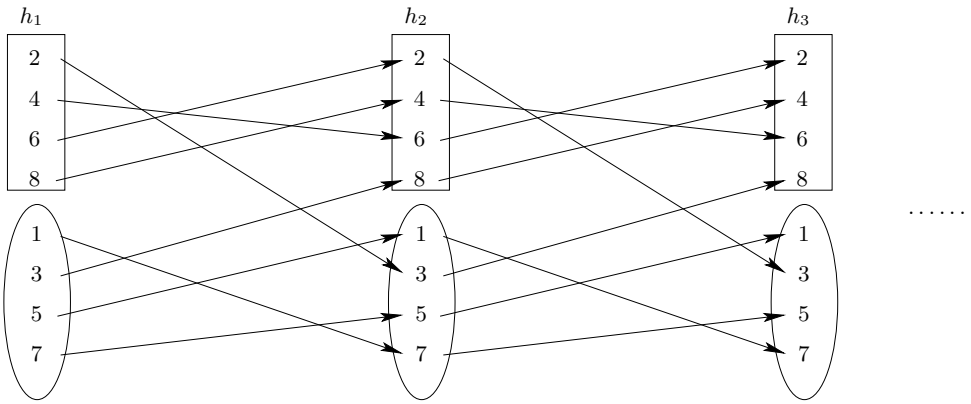


Figure 4: The hidden Markov model used to generate sequences to demonstrate Bayesian logistic sequence prediction models. Only the dominant transition probabilities of 0.95 are shown using arrows in the above graph. Rectangles enclose states that are likely to produce the observation 1; ovals enclose states likely to produce the observation 2.

5000 sequences as test cases, and the remaining 500 as the training cases. We tested the methods by predicting x_{21} based on O preceding states, with O set to 1, 2, 3, 4, 5, 7, 12, 15, 17, and 20.

Figure 5 compares the number of parameters and the computation time for training 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 parameters decreases with the number, O, of preceding states used, reaching 0.207 when $O = 20$. This ratio will go to zero for even larger values of O, as discussed in Section 3.3. Computation time per Markov chain iteration during training is also smaller with our compression method, but it will not converge to a finite amount as O increases, since the time used to update the hyperparameters $\log(\sigma_o)$'s grows with O, even when the number of compressed parameters has converged after a certain order. (However, if this were to become a practical issue, it would probably be possible to compress the σ_o parameters as well.)

Figure 5 also shows the prediction times with and without compression for 5000 test cases. In order to save space, we re-split compressed parameters for each test case if necessary, as described in Section 3.5. For each test case, we need to split at most one compressed parameter, into two parts. Splitting is necessary for only a fraction of test cases — for example when the order $O = 20$, this fraction is 0.6572. The time for prediction with parameter compression is smaller than when using the original parameters, partly because the time for splitting is small, and partly because more summation is needed when parameters are not compressed. Even with compression, predictions times grow with the order, O, because the time used to identify the patterns in a superpattern expressed by a test case grows with O.

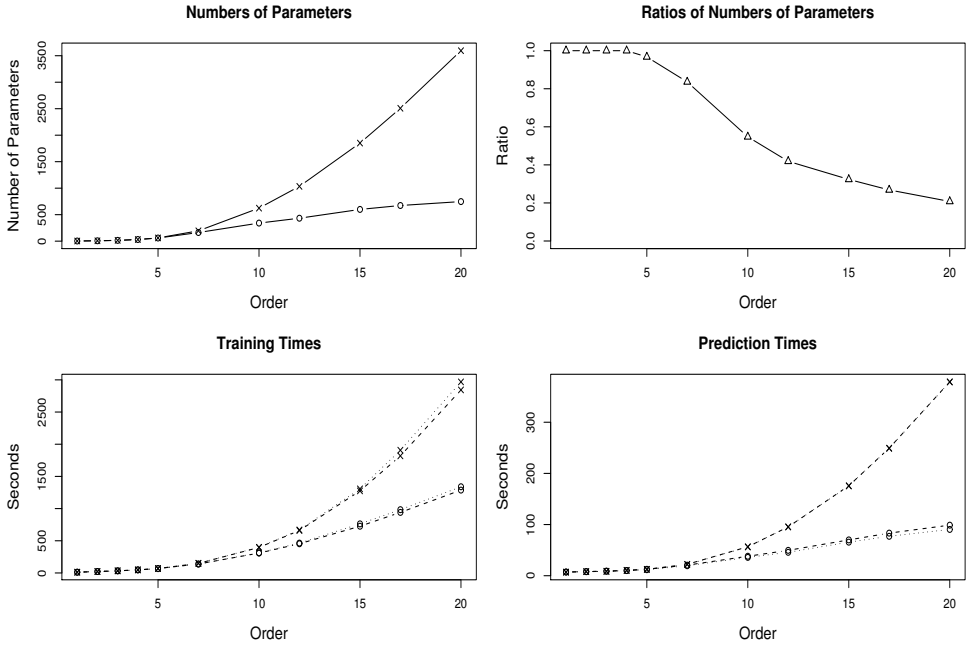


Figure 5: Reduction in the number of parameters, and in training and prediction times using our compression method applied to the HMM data. The upper-left plot shows how the number of compressed ($\circ$) and original ($\times$) parameters based on 500 training sequences varies with the order, O. The ratios of these numbers are shown in the upper-right plot. The lower plots show computation times for training and prediction. Dashed lines are with Cauchy priors; dotted lines are with Gaussian priors. Note that the curves of prediction times with original parameters for Cauchy and Gaussian priors almost overlap.

Compressing parameters also improves the quality of Markov chain sampling. Figure 7 shows autocorrelation plots of the σ_o hyperparameters, for $o = 10, 12, 15, 17, 20$, when the length of the preceding sequence is $O = 20$. Autocorrelation decreases more rapidly with lag when we compress the parameters. If we rescaled the lags in the autocorrelation plots to take account of the lower computation time per iteration with compression, the reduction of autocorrelation with the compressed parameters would be much more pronounced.

Finally, Figure 6 shows predictive performance in terms of error rate (the fraction of wrong predictions in test cases), and the average minus log probability (AMLP) of the true response in a test case based on the predictive probability for different classes. Predictive performance with and without compressing parameters is the same, as should be the case in theory, and will be in practice when the Markov chains for the two methods both converge and sample well. Performance using Cauchy and Gaussian priors is also

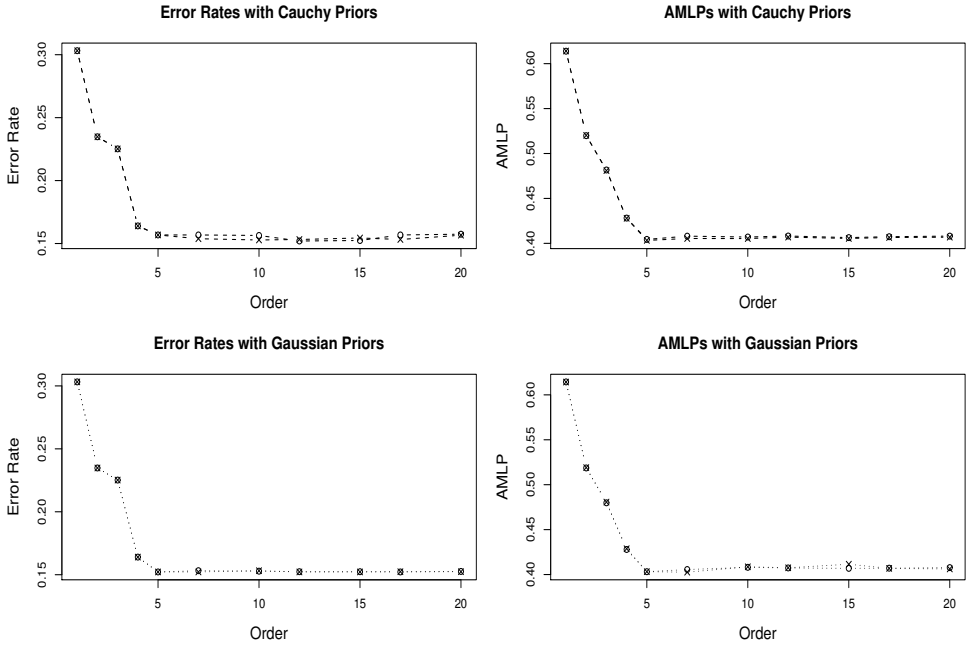


Figure 6: Predictive performance on the HMM data, with parameter compression (∘) and without compression (×). "AMLP" is the average of minus the log probability of the test symbols.

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, though 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, avoiding any overfitting problem from using a complex model. We therefore do not need to restrict the order of the Bayesian sequence prediction models to a very small number for statistical reasons, and with our method for compressing parameters, restricting the order for computational reasons is also unnecessary.

4.3 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, and punctuation. For example, character sequence "with me. They" was encoded as "212232132212". We then collapsed multiple occurrences of 3 into only one occurrence. The length of the whole sequence is 3930. From it, we created a data set with 3910 overlapped sequences (ie, the first

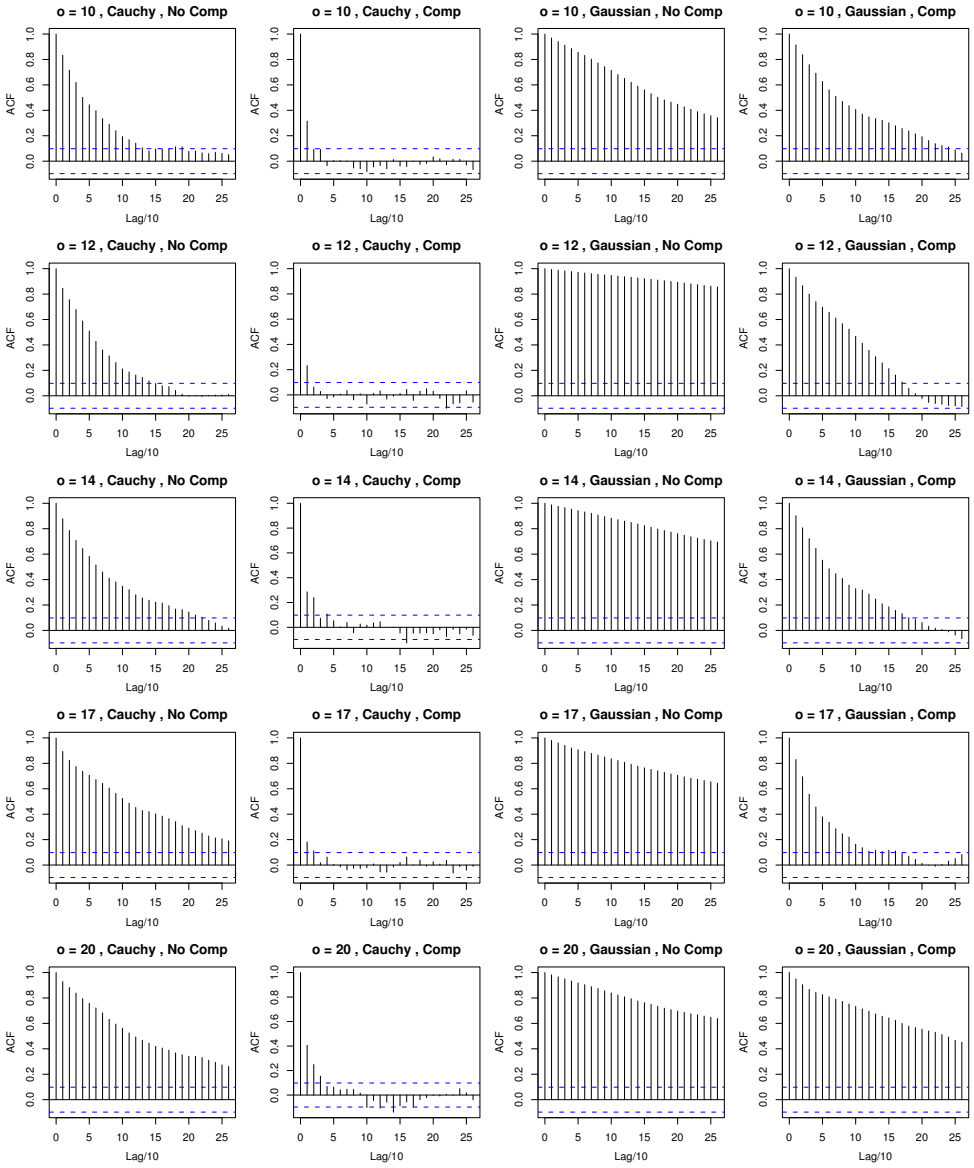


Figure 7: The autocorrelation plots of σ_o's for the experiments on a data set generated by a HMM, when the length of the preceding sequence $O = 20$. We show the autocorrelations of σ_o, for $o = 10, 12, 14, 17, 20$. In the above plots, "Gaussian" in the titles indicates the methods with Gaussian priors, "Cauchy" indicates with Cauchy priors, "comp" indicates with parameters compressed, "no comp" indicates without parameters compressed.

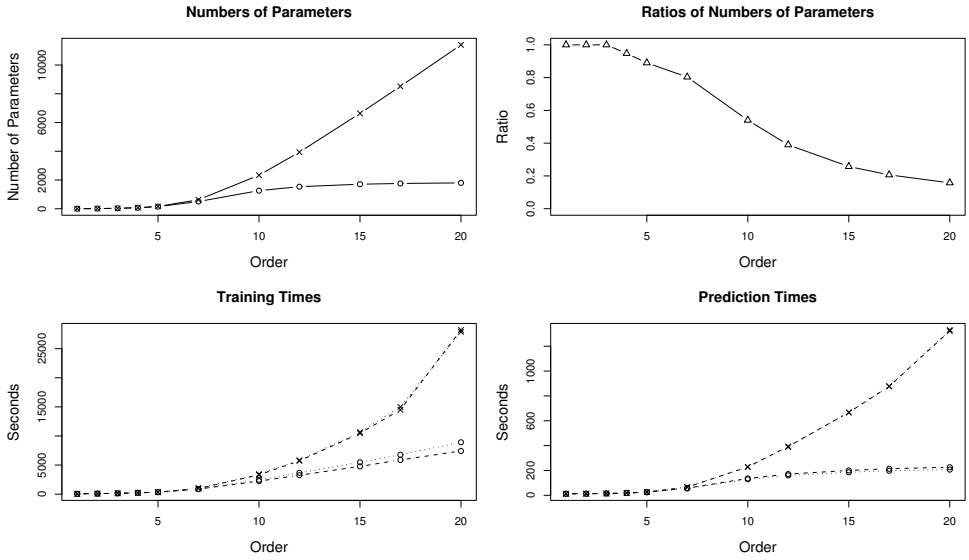


Figure 8: Reduction in the number of parameters, and in training and prediction times using our compression method applied to the English text data. The upper-left plot shows how the number of compressed ($\circ$) and original ($\times$) parameters based on 500 training sequences varies with the order, O. The ratios of these numbers are shown in the upper-right plot. The lower plots show computation times for training and prediction. Dashed lines are with Cauchy priors; dotted lines are with Gaussian priors.

charater of the second sequence is the second character of the first sequence) of length 21, and used the first 1000 as training data, and the remaining 2910 as test cases. The priors and computational parameters used were as described in Section 4.1.

The results are shown in Figures 8 through 11. The conclusions drawn from the experiments in Section 4.2 are confirmed in this example, with some differences in details. As there, our compression method greatly reduces the number of parameters, and therefore greatly reduces the computation time needed for training. The speed 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. There is reason to think that very high order interactions are of significance with English text, and from Figure 9, it appears that at least some interactions with order between 7 and 10 are indeed useful in predicting the next character.

In terms of error rates and AMLPs, the Cauchy models and the Gaussian models perform similarly for this example, as seen in Figure 9. But it is interesting to investigate the difference of using Gaussian and Cauchy priors. We first plotted the medians of Markov chains samples (in the last 2500 iterations) of all compressed parameters, s, for

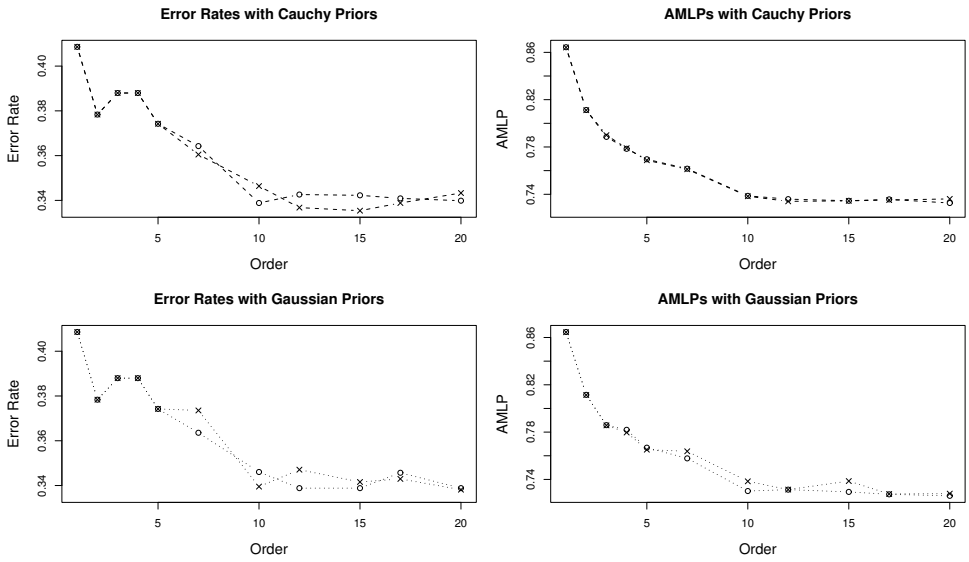


Figure 9: Predictive performance on the English text data, with parameter compression ($\circ$) and without compression ($\times$). "AMLP" is the average of minus the log probability of the test symbols.

the model with $O = 10$. These are shown in Figure 10, where the right plot shows in a larger scale the rectangle $(-2, 2) \times (-2, 2)$. A few parameters with large medians in the Cauchy model have very small medians in the Gaussian model, but on the other hand, as shown by the right plot, most β close to 0 have smaller medians in the Cauchy model, illustrating that posterior distributions based on Cauchy priors concentrate more around 0 than those based on Gaussian priors.

This difference is demonstrated further by the Markov chain traces of some particular compressed parameters that are shown in Figure 11. The three compressed parameters shown there 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 that 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 displayed in Figure 10.

The regression coefficient for "CC:V" should not be far from zero by our common sense, since two consonants are often followed by any of the 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" or "world". The Markov chain trace of this β with a Cauchy prior moves in a smaller region around zero than with a Gaussian prior. But if we look back one more character, things are different. The regression

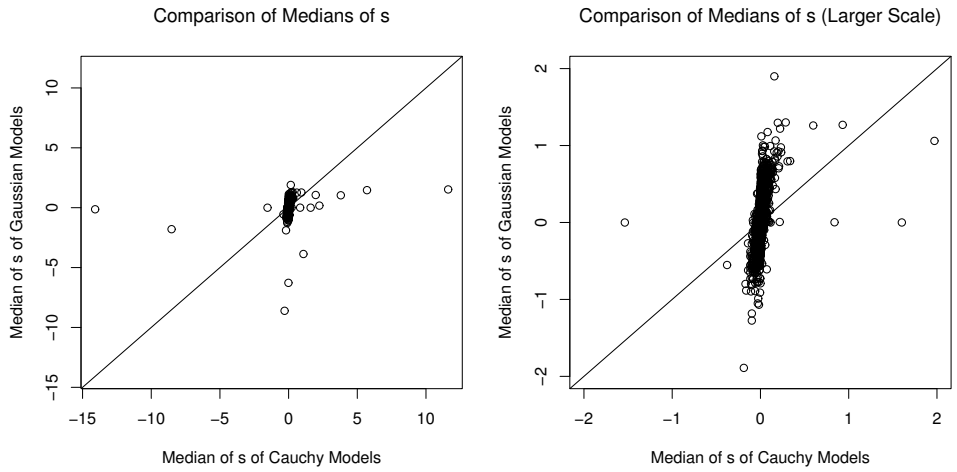


Figure 10: Scatterplots of medians of all compressed parameters, s, of Markov chain samples in the last 1250 iterations, for the models with Cauchy and Gaussian priors, fitted with English text data, with the length of preceding sequence $O = 10$, and with the parameters compressed. The right plot shows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

coefficient for "_CC:V" is fairly large, which is not surprising. The two consonants in "_CC:V" suggest two letters in the beginning of a word. Words starting with three consonants are fairly uncommon (though they exist, e.g. "strong"), and words consisting only of just two consonants are very uncommon. The posterior distribution of this β 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 the first seven letters of words like "statistics" and "statistical", which appear repeatedly 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 the Gaussian model, but sometimes it is close to zero, indicating that there might be two modes for its posterior distribution.

The above investigation reveals that a Cauchy model allows some useful regression coefficients to be much larger in magnitude than others while keeping the useless coefficients in a smaller region around zero than a Gaussian model. In other words, Cauchy models are more powerful than Gaussian models in finding the few useful interactions from the many possible in a high-order model, due to the heavy two-sided tails of Cauchy distributions. However, this difference may make little effect on the predictive performance in logistic classification models, presumably because the predictive performance of a logistic classification model is not very sensitive to the magnitude of coefficients provided that they are moderately large (e.g. models with $\beta = 100$, $\beta = 20$, and $\beta = 5$ for a binary covariate and the same other coefficients would predict similarly).

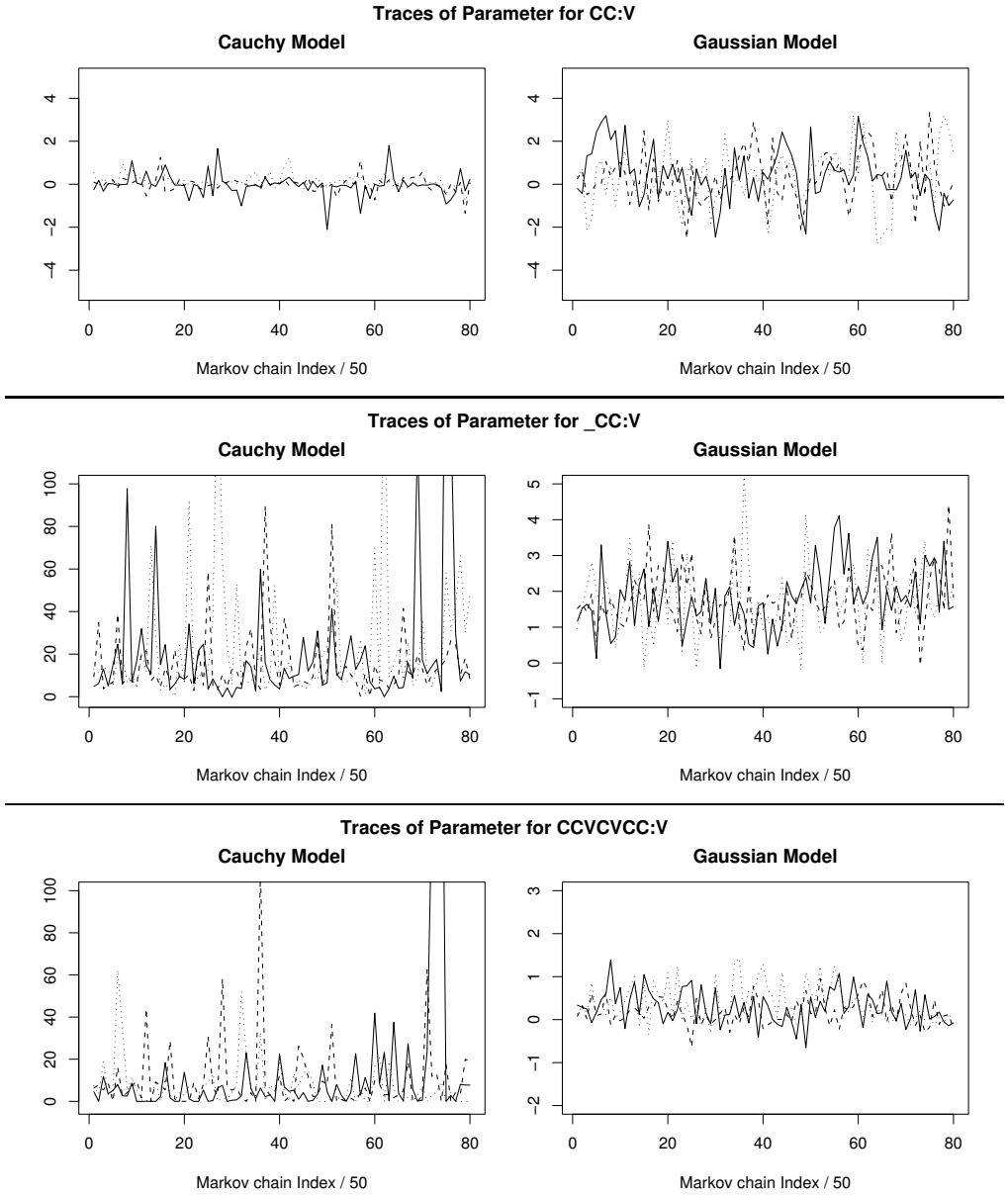


Figure 11: Markov chain traces of three compressed parameters (each containing only one β) from the experiments on English text. Traces from three independent Markov chains are shown, for models with Cauchy priors and with Gaussian priors. The parameters are annotated by their original meanings in the English text. 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".

We anticipate that the Cauchy models will more often perform better than Gaussian models for regression problems with continuous responses, though such problems are not illustrated here.

It is worthy of mentioning distributions other than the Cauchy that have heavy tails, such as the Laplace distribution and the Student's t distributions with small degrees of freedom. Unfortunately, Laplace and general t distributions (other than those with 1 degree of freedom, which is Cauchy) are not symmetric stable distributions. We therefore cannot compress the parameters for Laplace and general t models as we do for Gaussian and Cauchy models. But if time is not an issue, they are also appropriate as priors for high-order models. In fact, after examining our data set, we have found that sequences "_CC" and "CCVCVCC" are all followed by vowel letters, so their likelihood functions actually favour infinite parameter values, with consequence of that their Markov chain traces sometimes visit some very large (and a little bit undesirable) values in Cauchy models, as allowed by the heavy tails of Cauchy distributions. If the Cauchy models were fit with a larger data set, where sequences "_CC" and "CCVCVCC" are not all be followed by vowel letters, these undesirable large values would not be possible, but prior distributions with lighter tails than the Cauchy (but heavier than the Gaussian) would still be interesting. Symmetric stable distributions with index between 1 (Cauchy) and 2 (Gaussian) are an attractive possibility, which would fit with our compression scheme, except that we have not found a way of dealing with the difficulty of evaluating their probability density functions.

5 Conclusion and Discussion

In this paper, we have proposed a method to effectively reduce the number of parameters of Bayesian classification and regression models with high-order interactions, using a compressed parameter to represent the sum of all the regression coefficients for predictor 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 efficiently split into the original parameters, using direct sampling methods. We have demonstrated that, for a data set with a fixed number of cases, the number of compressed parameters will converge to a limiting value regardless of how large the maximum order of the model is. 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. Predictive performance will be improved by considering high-order interactions, if some high-order interactions do exist, which there is often good reason to suspect.

We have devised an efficient scheme for implementing this idea for Bayesian logistic sequence prediction models. A similar, but more complex, scheme for grouping parameters of general Bayesian logistic classification models is presented in Li (2007).

We have also illustrated here that Cauchy distributions with location parameter zero, 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 zero, but a few may be much larger in absolute value. This prior will often be appropriate for the regression coefficients in high-order models. In light of the fact that Cauchy distributions allow some very large values, which may be inappropriate as parameter values of regression and classification models, symmetric stable distributions with index between 1 (Cauchy) and 2 (Gaussian) may be more appropriate as priors for high-order models. These distributions would fit with our compression scheme if we can find a way of evaluating their probability density functions efficiently and a way of splitting the sum of random variables with such distributions.

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 model the conditional distribution of the continuous response variable given the predictor variables with, for example, a Gaussian or t distribution. Unless one converts the continuous features into discrete values, it is not clear how to apply the method described in this paper 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 the likelihood function when training models with MCMC, probably by transforming the original parameters.

References

Baker, J. K. (1975). "The Dragon system - an overview." *IEEE Transactions on. Acoustic Speech Signal Processing*, ASSP-23(1): 24–29. 810

Barankin, E. W. (1961). "Sufficient Parameters: Solution of the Minimal Dimensionality Problem." *Annals of the Institute of Statistical Mathematics*, 12: 91–118. 800

Bell, T. C., Cleary, J. G., and Witten, I. H. (1990). *Text Compression*. Prentice-Hall. 793

Cheverud, J. M. and Routman, E. J. (1995). "Epistasis and Its Contribution to Genetic Variance Components." *Genetics*, 139: 1455–1461. 793

Dawid, A. P. (1979). "Conditional Independence in Statistical Theory." *Journal of Royal Statistical Society (B)*, 41(1): 1–31. 800

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1999). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press. 793

Feller, W. (1966). *An Introduction to Probability Theory and its Applications*, volume II. New York: John Wiley. 796

Gelfand, A. E. and Smith, A. F. M. (1990). "Sampling-based approaches to calculating marginal densities." *Journal of the American Statistical Association*, 85: 398–409. 808

Li, L. (2007). "Bayesian Classification and Regression with High Dimensional Features." Ph.D. thesis, University of Toronto. 819

Neal, R. M. (1993). "Probablistic Inference using Markov Chain Monte Carlo Methods." Technical report, Dept. of Computer Science, University of Toronto. URL ftp://ftp.cs.toronto.edu/pub/radford/review.ps.Z 808

— (2003). "Slice Sampling." *Annals of Statistics*, 31: 705–767. 808

Poirier, D. J. (1998). "Revising Beliefs in Nonidentified Models." *Econometric Theory*, 14: 483–509. 800

Ritchie, M. D., Hahn, L. W., Roodi, N., Bailey, L. R., Dupont, W. D., Parl, F. F., and Moore, J. (2001). "Multifactor-Dimensionality Reduction Reveals High-Order Interactions among Estrogen-Metabolism Genes in Sporadic Breast Cancer." *The American Journal of Human Genetics*, 69: 138–147. 793

Romberg, J., Choi, H., and Baraniuk, R. (2001). "Bayesian tree-structured image modeling using wavelet-domain hidden Markov models." *IEEE Transactions on image processing*, 69: 138–147. 810

Sun, S. (2006). "Haplotype Inference Using a Hidden Markov Model with Efficient Markov Chain Sampling." Ph.D. thesis, University of Toronto. 810

Thisted, R. A. (1988). *Elements of Statistical Computing*. Chapman and Hall. 802

Wright, S. (1980). "Genic and organismic selection." *Evolution*, 34: 825–843. 793

Acknowledgments

This research was supported by Natural Sciences and Engineering Research Council of Canada. Radford Neal holds a Canada Research Chair in Statistics and Machine Learning. The authors also appreciate the anonymous reviewers for providing many valuable comments on the previous draft of this paper.