Stat Comput (2011) 21: 247–259
DOI 10.1007/s11222-009-9165-4

Stochastic matching pursuit for Bayesian variable selection

Ray-Bing Chen · Chi-Hsiang Chu · Te-You Lai · Ying Nian Wu

Received: 3 May 2009 / Accepted: 30 November 2009 / Published online: 10 December 2009

Abstract This article proposes a stochastic version of the matching pursuit algorithm for Bayesian variable selection in linear regression. In the Bayesian formulation, the prior distribution of each regression coefficient is assumed to be a mixture of a point mass at 0 and a normal distribution with zero mean and a large variance. The proposed stochastic matching pursuit algorithm is designed for sampling from the posterior distribution of the coefficients for the purpose of variable selection. The proposed algorithm can be considered a modification of the componentwise Gibbs sampler. In the componentwise Gibbs sampler, the variables are visited by a random or a systematic scan. In the stochastic matching pursuit algorithm, the variables that better align with the current residual vector are given higher probabilities of being visited. The proposed algorithm combines the efficiency of the matching pursuit algorithm and the Bayesian formulation with well defined prior distributions on coefficients. Several simulated examples of small n and large p are used to illustrate the algorithm. These examples show that the algorithm is efficient for screening and selecting variables.

Keywords Gibbs sampler · Metropolis algorithm · Stochastic search variable selection

R.-B. Chen (✉) · C.-H. Chu · T.-Y. Lai
Institute of Statistics, National University of Kaohsiung,
Kaohsiung, Taiwan
e-mail: rbchen@nuk.edu.tw

Y.N. Wu
Department of Statistics, University of California, Los Angeles,
CA, USA

1 Introduction

In the past two decades, Bayesian methods for variable selection in linear regression models have become increasingly popular. In their pioneering paper, George and McCulloch (1993) first proposed a Bayesian variable selection method, called stochastic search variable selection. Some other Bayesian methods related to stochastic search variable selection were studied by Chipman (1996), Chipman et al. (1997), and George and McCulloch (1997). These Bayesian methods have been successfully applied to model selection for supersaturated designs (Beattie et al. 2002), signal processing (Wolfe et al. 2004, and Févotte and Godsill 2006), and gene selection (Lee et al. 2003).

In the Bayesian formulation of George and McCulloch (1993, 1997), the prior distributions of the coefficients are assumed to be independent, and the prior distribution of each coefficient is assumed to be a mixture of two normal distributions. Both normal distributions are centered at 0, but one has a very small variance and the other has a much larger variance. The normal distribution with a very small variance is used to model the coefficients of the variables that are not selected or are not active, and the normal distribution with a very large variance is used to model the coefficients of the variables that are selected or are active. For such a mixture of normal prior distribution, one can augment an indicator for each variable to indicate whether this variable is active or not. The stochastic search variable selection of George and McCulloch (1993) is a Gibbs sampler scheme that samples from the posterior distribution of the indicators and the coefficients. The algorithm iteratively samples the indicators given the coefficients, and then samples the coefficients given the indicators. The subset of variables with the highest posterior probability is considered the “best” model.

In this article, we adopt the commonly used notation of n and p, where n is the number of observations, and p is the number of variables. In the Gibbs sampling scheme of George and McCulloch (1993), the step of sampling the coefficients conditional on the indicators requires the computation of the posterior covariance matrix of these coefficients, which is proportional to the inverse of a $p \times p$ information matrix. The computational complexity is $O(p^3)$, which can be expensive if p is large. In addition to this expensive step, stochastic search variable selection of George and McCulloch (1993) also requires expensive computations for sampling the indicators simultaneously. George and McCulloch (1997) suggested several schemes for reducing the computational costs. One of them is to use the Cholesky decomposition of the information matrix, and others focus on how to efficiently sample the indicators.

This article proposes a stochastic matching pursuit algorithm to avoid the inversion of potentially large matrices and to improve the efficiency of the Markov chain Monte Carlo algorithm for posterior sampling. We adopt the Bayesian formulation of George and McCulloch (1993), except that we assume that the variance of the normal distribution for the inactive coefficients is 0. That is, the distribution of the inactive coefficients is a point mass at 0. The proposed stochastic matching pursuit algorithm is a Markov chain Monte Carlo algorithm for sampling from the posterior distribution of the coefficients.

The original version of the matching pursuit algorithm was proposed by Mallat and Zhang (1993) in the context of wavelet sparse coding. The algorithm is essentially a forward stepwise variable selection method in linear regression. The algorithm is a greedy one. It starts from an empty set of variables. Then at each step, the algorithm identifies a variable that has the maximum correlation with the current residual vector. It selects this variable, computes its coefficient, and updates the residual vector. This algorithm has proven to be very efficient for variable selection, and has been widely used in signal processing.

However, the matching pursuit algorithm is only a procedure, not a principle, in the sense that it does not explicitly minimize any objective function or criterion. The greedy nature of the algorithm can also cause problem in the sense that the procedure may not select the optimal set of variables. Our method takes advantage of the efficiency of the original matching pursuit algorithm, and modifies it into a Markov chain Monte Carlo algorithm that samples from the posterior distribution of the coefficients with well defined mixture prior distributions.

The stochastic matching pursuit algorithm is a Metropolis scheme with a pair of reversible moves. One is the "addition move," which adds a new variable to the existing set of selected variables, where the variables with larger correlations with the residual vector are assigned higher probabilities of being added, in a fashion that is very similar to the original matching pursuit algorithm. The other move is the "deletion move," which deletes a variable from the existing set of selected variables. The deletion move complements the addition move so that the Markov chain can be made reversible.

Several simulated examples are used to illustrate the proposed algorithm. In our simulation studies of small n and large p problems with $(n, p) = (50, 200)$ and $(100, 400)$, the stochastic matching pursuit algorithm performs well in screening and selecting variables. In comparison with existing methods for variable selection, the computational cost of stochastic matching pursuit is lower than that of stochastic search variable selection when the number of predictors, p, is large. The proposed method is competitive with the screening method of Shao and Chow (2007) and a Lasso type method (Tibshirani 1996) in terms of variable selection results.

We also compare our method with a Bayesian method of Smith and Kohn (1996), who used conjugate prior distributions for the coefficients of the selected variables, so that these coefficients can be integrated out in closed form. However, the algorithm of Smith and Kohn (1996) involves matrix inversion, which can be computationally expensive and which may be numerically unstable when p is large. Simulation studies with $n = 50$ and $p = 20$, 50, 100 and 300 show that our method compares favorably with the method of Smith and Kohn (1996) in terms of variable selection results.

This article is organized as follows. Section 2 presents the stochastic matching pursuit algorithm and explains how to choose the tuning parameters. Section 3 studies small n and large p problems, where the stochastic matching pursuit algorithm is applied to the simulated data sets studied by Shao and Chow (2007) as well as to an example of image representation using Gabor wavelet elements. Section 4 compares the computational cost of the stochastic matching pursuit algorithm with that of stochastic search variable selection. It also compares the proposed method with the method of Smith and Kohn (1996) that is based on conjugate prior. Finally Sect. 5 summarizes our findings and discusses future directions.

2 Stochastic matching pursuit

In this section, we first review stochastic search variable selection of George and McCulloch (1993, 1997). Then we present the stochastic matching pursuit algorithm.

2.1 Variable selection in linear regression

To fix notation, consider the following linear regression model

$$Y = \mathbf{X}\beta + \epsilon, \tag{1}$$

where Y is an $n \times 1$ response vector, $\mathbf{X} = [X_1, \ldots, X_p]$ is an $n \times p$ matrix, and X_i is the i-th predictor variable or regressor. $\beta = (\beta_1, \ldots, \beta_p)'$ is a $p \times 1$ vector of the unknown coefficients, and $\epsilon = (\epsilon_1, \ldots, \epsilon_n)'$ is an $n \times 1$ noise vector that follows a multivariate normal distribution with zero mean vector and covariance matrix $\sigma^2 I_n$, where I_n is an $n \times n$ identity matrix.

The variable selection problem is to select the "best" subset of variables from $X_1, \ldots, X_p$ to model the response vector Y. We can introduce a $p \times 1$ vector of latent variables, $\gamma = (\gamma_1, \ldots, \gamma_p)'$, to indicate which variables are selected. Each γ_i is an indicator that takes value 0 or 1. $\gamma_i = 1$ means that the variable X_i is selected or active, and $\gamma_i = 0$ means that the variable X_i is not selected or inactive.

2.2 Stochastic search variable selection

In stochastic search variable selection of George and McCulloch (1993, 1997), the prior distribution of the coefficient β_i given the indicator γ_i is

$$[\beta_i | \gamma_i = 0] \sim N(0, \nu_{0i}), \quad \text{and} \quad [\beta_i | \gamma_i = 1] \sim N(0, \nu_{1i}). \tag{2}$$

The value of ν_{0i} is set to be small, and $N(0, \nu_{0i})$ is the prior distribution of the coefficient β_i if the variable X_i is not selected or if X_i is inactive. The value of ν_{1i} is set to be large, and $N(0, \nu_{1i})$ is the prior distribution of β_i if the variable X_i is selected or active. Usually one can assume that $\nu_{1i} = c_i \nu_{0i}$.

The prior distribution of γ_i is $P(\gamma_i = 0) = p_i$, and $P(\gamma_i = 1) = 1 - p_i$. The prior distributions of (γ_i, β_i) are assumed to be independent for $i = 1, \ldots, p$, and they are independent of the prior distribution of the residual variance σ^2, which is an inverse Gamma distribution, $\sigma^2 \sim IG(\nu/2, \nu\lambda/2)$.

The stochastic search variable selection procedure is a Gibbs sampling scheme where each iteration samples from the conditional distributions $[\beta | \gamma, Y, \sigma]$, $[\gamma | \beta, Y, \sigma]$, and $[\sigma | Y, \beta, \gamma]$. The best subset of variables are selected according to the information contained in the Monte Carlo samples of γ.

Within each iteration of the above Gibbs sampling scheme, the most costly step is to sample β from a multivariate normal distribution, $[\beta | \gamma, Y, \sigma] \sim N(\sigma^{-2} A_\gamma X'Y, A_\gamma)$, where $A_\gamma = (\sigma^{-2} X'X + D_\gamma^{-1} R^{-1} D_\gamma^{-1})^{-1}$, R is the prior correlation matrix, and $D_\gamma^{-2} = \mathrm{diag}[(a_1 \nu_{01})^{-1}, \ldots, (a_p \nu_{0p})^{-1}]$ with $a_i = 1$ if $\gamma_i = 0$, or $a_i = c_i$ if $\gamma_i = 1$. The computational complexity of this step is $O(p^3)$.

2.3 Componentwise Gibbs sampler

To avoid computing the inverse matrices, we may adopt a componentwise Gibbs sampler. The stochastic matching pursuit algorithm is a further improvement of the componentwise Gibbs sampler.

Before describing the componentwise Gibbs sampler, we first make a minor modification to the prior specification. Following Geweke (1996) and Smith and Kohn (1996), we set $\nu_{0i} = 0$, thus (2) becomes

$$\beta_i | \gamma_i \sim (1 - \gamma_i)\delta_0 + \gamma_i N(0, \tau_i^2), \tag{3}$$

where δ_0 is a point mass at 0 and $\tau_i^2 = \nu_{1i}$. That is, we simply assume that the coefficients of the inactive variables equal 0. Without much loss of generality, we assume that all the variables have the same probability to be included into the model, i.e., $\rho = p_i = P(\gamma_i = 0)$, $i = 1, \ldots, p$. We also assume that $\tau = \tau_i$ for $i = 1, \ldots, p$.

For such a prior distribution, Smith and Kohn (1996) and George and McCulloch (1997) suggested modifications of stochastic search variable selection. However, it is still necessary to compute inverse matrices in their modified algorithms.

The componentwise Gibbs sampler samples (γ_i, β_i) one at a time conditioning on $(\gamma_{-i}, \beta_{-i})$, where the commonly used notation $-i$ means all the components except the i-th one. In fact, George and McCulloch (1997) suggested generating γ_i componentwise from the full conditionals, $[\gamma_i | \gamma_{-i}, Y]$, in order to save the computational cost. However, in the componentwise Gibbs sampler, one does not only sample γ componentwise, but one can also sample β componentwise. The key step in this Gibbs sampler is to compute the likelihood ratio

$$z_i = \frac{P(Y | \gamma_i = 1, \{\beta_k, \forall k \neq i\})}{P(Y | \gamma_i = 0, \{\beta_k, \forall k \neq i\})}, \quad i = 1, \ldots, p.$$

It is easy to show that

$$z_i = \sqrt{\frac{\sigma_{i\star}^2}{\tau^2}} \exp\left\{\frac{r_i^2}{2\sigma_{i\star}^2}\right\}, \tag{4}$$

where

$$\sigma_{i\star}^2 = \frac{\sigma^2 \tau^2}{X_i' X_i \tau^2 + \sigma^2},$$

$$r_i = \frac{R_i' X_i \tau^2}{\sigma^2 + X_i' X_i \tau^2},$$

and $R_i = Y - \Sigma_{k \neq i} \beta_k X_k$. For more details about (4), please see Lemma 3.1 in Lai (2007).

Algorithm 1 is a description of the componentwise Gibbs sampler. It is essentially the same as the Bayesian search algorithm of Geweke (1996), who assumed that the prior distribution of β_i conditional on $\beta_i \neq 0$ is a truncated normal distribution.

Algorithm 1 (The componentwise Gibbs sampler for variable selection)

(I) *Randomly select a variable* X_i. *Compute* $R_i = Y - \Sigma_{k \neq i} \beta_k X_k$, *and let*

$$\sigma_{i\star}^2 = \frac{\sigma^2 \tau^2}{X_i' X_i \tau^2 + \sigma^2}, \qquad r_i = \frac{R_i' X_i \tau^2}{\sigma^2 + X_i' X_i \tau^2}.$$

(II) *Compute*

$$z_i = \frac{p(Y|\gamma_i = 1, \{\beta_k, \forall\, k \neq i\})}{p(Y|\gamma_i = 0, \{\beta_k, \forall\, k \neq i\})} = \sqrt{\sigma_{i\star}^2/\tau^2} \exp\left\{\frac{r_i^2}{2\sigma_{i\star}^2}\right\}.$$

Then evaluate the posterior probability

$$P(\gamma_i = 1|\{\beta_k, \forall\, k \neq i\}, Y) = \frac{(1-\rho) z_i}{\rho + (1-\rho) z_i}.$$

(III) *Sample* γ_i *from the above posterior probability. If* $\gamma_i = 0$, *then set* $\beta_i = 0$, *otherwise, sample* $\beta_i \sim N(r_i, \sigma_{i\star}^2)$. *Go back to* (I).

(IV) *After a number of iterations of the above steps, compute the current residual vector,* $\text{Res} = Y - \sum_i \beta_i X_i$. *Then sample* $\sigma^2 \sim IG(\frac{n+\nu}{2}, \frac{\text{Res}'\text{Res}+\nu\lambda}{2})$. *Go back to* (I).

In the above algorithm, σ^2 is updated less frequently than the coefficients and indicators.

2.4 Metropolize the matching pursuit algorithm

In the componentwise Gibbs sampler, the variables are visited in random order. They can also be visited by a systematic scan. Both the random and systematic scans can cause problems. If the variables are highly correlated and if the residual variance is small, an inferior variable can be visited first and then selected, thus preventing a variable of more importance from being selected. So a better selection scheme should be devised, where the variables compete to be selected. This motivates us to design the stochastic matching pursuit algorithm.

The matching pursuit algorithm of Mallat and Zhang (1993) is widely used in wavelet sparse coding, where the goal is to represent a signal by a small number of wavelet elements selected from a large dictionary. This algorithm is essentially a forward stepwise variable selection procedure in linear regression. Initially, all the coefficients β_i's are set to be 0, and the initial residual vector $R = Y$. Without loss of generality, let us assume that all the predictor vectors X_i are normalized to have $\|X_i\|^2 = 1$. Then each step of the matching pursuit algorithm selects the X_i that achieves the maximum magnitude of the inner product $|\langle R, X_i\rangle|$. After X_i is selected, its coefficient is updated to $\beta_i \leftarrow \beta_i + \langle R, X_i\rangle$, and the residual is updated to $R \leftarrow R - \langle R, X_i\rangle X_i$. The procedure stops when the maximum of $|\langle R, X_i\rangle|$ is below a threshold.

Unlike the random or systematic updating of the coefficient β_i, each step of the matching pursuit algorithm updates the coefficient of the variable that gives the best fit to the current residual vector. This can avoid the problem of the componentwise Gibbs sampler discussed above. Our stochastic matching pursuit algorithm incorporates this feature by using a Metropolis scheme. The algorithm is an improvement of the componentwise Gibbs sampler.

The Metropolis scheme consists of a pair of reversible moves: "addition" and "deletion". The addition move adds a variable to the current set of active variables. The addition move is very similar to the matching pursuit algorithm, where those variables that have larger correlations with the current residual vector are given higher probabilities of being selected. The deletion move deletes a variable from the current set of active variables, and it is a reversal of the addition move to make the Markov chain reversible.

Specifically, suppose currently there are A active variables. With probability p_{add}, we propose to add an inactive variable to the set of active variables. With probability $p_{\text{delete}} = 1 - p_{\text{add}}$, we propose to delete an active variable, or make it inactive.

The following is the proposal on how to add a variable, which is similar to matching pursuit. Among all the inactive variables, we compute $z_i = p(Y|\gamma_i = 1, \{\beta_k, \forall k \neq i\})/p(Y|\gamma_i = 0, \{\beta_k, \forall k \neq i\})$ as in (I) and (II) of the componentwise Gibbs sampler of Algorithm 1. This z_i is the likelihood ratio for testing whether $\gamma_i = 1$. The larger z_i is, the more promising the variable X_i is. In the Metropolis scheme, we propose to add a variable by sampling a variable i from the group of inactive variables, where the probability for a variable i to be sampled is proportional to z_i, that is, the proposal probability of adding the variable i into the set of active variables is $z_i / \sum_{i:\gamma_i=0} z_i$. Meanwhile, we propose to sample β_i according to (III) of the componentwise Gibbs sampler.

The proposal for deleting an active variable is simple. Among all the active variables, we randomly select one, and set the corresponding indicator and coefficient to 0.

The pair of addition and deletion moves makes it possible to design a reversible Markov chain using the Metropolis scheme. We only need to calculate the acceptance probability of the addition proposal and the acceptance probability of the deletion proposal.

The acceptance probability for the proposal of adding variable i whose current $\gamma_i = 0$ is

$$\begin{aligned} &p_{\text{accept-add}} \\ &\quad = \min\left[1, \frac{P(\gamma_i = 1|\{\beta_k, \forall\, k \neq i\}, Y)}{P(\gamma_i = 0|\{\beta_k, \forall\, k \neq i\}, Y)} \frac{p_{\text{delete}}}{p_{\text{add}}} \frac{1/(A+1)}{z_i / \sum_{j:\gamma_j=0} z_j}\right] \\ &\quad = \min\left[1, \frac{(1-\rho) z_i}{\rho} \frac{p_{\text{delete}}}{p_{\text{add}}} \frac{1/(A+1)}{z_i / \sum_{j:\gamma_j=0} z_j}\right] \end{aligned}$$

$$= \min\left[1, \frac{(1-\rho)}{\rho}\frac{p_{\text{delete}}}{p_{\text{add}}}\frac{\sum_{j:\gamma_j=0} z_j}{(A+1)}\right]. \tag{5}$$

Recall that ρ is the prior probability that $\gamma_j = 0$.

The acceptance probability for the proposal of deleting variable i whose current $\gamma_i = 1$ is

$$\begin{aligned} &p_{\text{accept-delete}} \\ &= \min\left[1, \frac{P(\gamma_i = 0|\{\beta_k, \forall\, k \neq i\}, Y)}{P(\gamma_i = 1|\{\beta_k, \forall\, k \neq i\}, Y)}\frac{p_{\text{add}}}{p_{\text{delete}}}\frac{z_i/(\sum_{j:\gamma_j=0} z_j + z_i)}{1/A}\right] \\ &= \min\left[1, \frac{\rho}{(1-\rho)}\frac{p_{\text{add}}}{p_{\text{delete}}}\frac{A}{\sum_{j:\gamma_j=0} z_j + z_i}\right]. \end{aligned} \tag{6}$$

Note that in the above calculations, the likelihood ratio $z_i = p(Y|\gamma_i = 1, \{\beta_k, \forall k \neq i\})/p(Y|\gamma_i = 0, \{\beta_k, \forall k \neq i\})$ naturally balances out the posterior ratio $P(\gamma_i = 1|\{\beta_k, \forall k \neq i\}, Y)/P(\gamma_i = 0|\{\beta_k, \forall k \neq i\}, Y)$, so that the resulting acceptance probabilities have quite simple forms.

In particular, it is interesting to see that the $p_{\text{accept-add}}$ calculated in (5) does not depend on the variable i to be sampled, because the probability that i is sampled is proportional to z_i. $p_{\text{accept-add}}$ is decided by the overall fitness of all the inactive variables, i.e., $\sum_{j:\gamma_j=0} z_j$.

The $p_{\text{accept-delete}}$ calculated in (6), however, depends on which variable i is to be deleted. A subtle point is that all the z_j and z_i in (6) are re-calculated with the variable i turned inactive. If the variable i is an important one, $\sum_{j:\gamma_j=0} z_j + z_i$ can be very large, so that $p_{\text{accept-delete}}$ can be very small.

Algorithm 2 gives a detailed description of the stochastic matching pursuit algorithm.

Algorithm 2 (Stochastic matching pursuit for variable selection)

(I) *Let A be the number of active variables. With probability* p_{add}, *go to* (II). *With probability* $p_{\text{delete}} = 1 - p_{\text{add}}$ *go to* (IV).

(II) *With probability* $p_{\text{accept-add}}$ *calculated according to* (5), *go to* (III), *and with probability* $1 - p_{\text{accept-add}}$ *go back to* (I).

(III) *Among all the inactive variables i with* $\gamma_i = 0$, *sample a variable i with probability proportional to* z_i, *then let* $\gamma_i = 1$ *and sample* β_i *as described in* (III) *of Algorithm 1. Go back to* (I).

(IV) *If* $A > 0$, *then randomly select an active variable i with* $\gamma_i = 1$.

(V) *With probability* $p_{\text{accept-delete}}$ *calculated according to* (6), *accept the proposal of deleting the variable i, i.e., set* $\gamma_i = 0$, *and* $\beta_i = 0$. *With probability* $1 - p_{\text{accept-delete}}$, *reject the proposal of deleting variable i, and sample* β_i *as described in* (III) *of Algorithm 1. Go back to* (I).

(VI) *After a number of iterations of the above steps, compute the current residual vector,* $\text{Res} = Y - \sum_i \beta_i X_i$, *and then update* $\sigma^2 \sim IG(\frac{n+\nu}{2}, \frac{\text{Res}'\text{Res}+\nu\lambda}{2})$. *Go back to* (I).

In this algorithm, the step of sampling β_i of an active variable i is the same as in the componentwise Gibbs sampler. The difference is that the inactive variables compete with each other to be included in the model or to be turned active, in a fashion that is similar to matching pursuit. It is possible to introduce other reversible moves into the Metropolis algorithm, such as switching an active variable and an inactive variable. We shall explore such possibilities in future work.

The above algorithm combines the strengths of both matching pursuit and the componentwise Gibbs sampler. As in matching pursuit, the algorithm aggressively pursues promising variables. As in the componentwise Gibbs sampler, the algorithm does not require the inversion of large matrices, and it samples the posterior distribution of the coefficients with well defined mixture priors.

2.5 Implementation details

The stochastic matching pursuit algorithm produces a sequence of dependent random draws from the posterior distribution $[\gamma|Y]$ after a burn-in period. This Monte Carlo sample of γ's can be used for model selection. In this paper, we adopt the median probability criterion proposed by Barbieri and Berger (2004) for variable selection. Specifically, we estimate the posterior inclusion probability $P(\gamma_i = 1|Y)$ for each variable X_i from the Monte Carlo sample. If the estimated $P(\gamma_i = 1|Y)$ is greater than or equal to 1/2, then X_i is included into the model.

As for the tuning parameters, we set p_{add}, i.e., the probability of the addition move, to be 1/2, which is the same as p_{delete}, the probability of the deletion move.

For the parameters ν and λ in the prior distribution of σ^2, we follow George and McCulloch (1993) by setting $\lambda = 1$ and setting ν to be the non-zero components of γ_i.

We set ρ, the prior probability that a variable is excluded from the model, to be 1/2, following George and McCulloch (1993, 1997). In the situation of small n large p, this probability should be set to a larger value to reflect the prior belief of sparsity.

Now we consider the parameter τ, which is the prior standard deviation of the active variables. It plays an important role in determining the values of z_i and the conditional posterior probability of γ_i, $P(\gamma_i = 1|\{\beta_k, k \neq i\}, Y)$. It can be shown that the value of z_i is a decreasing function of τ if $\tau^2 > (R_i' R_i - \sigma^2)/(X_i' X_i)$. Thus the larger τ is, the smaller the value of z_i is. Since the conditional posterior probability of $\gamma_i = 1$ is decreasing as z_i decreases, the larger τ is, the smaller the conditional posterior probability of $\gamma_i = 1$ is, i.e., it is more difficult to include the variable X_i in the model. Hence large τ favors more parsimonious models, and small τ would yield more complex models.

If we have no prior knowledge about the parameter τ, we may use K-fold cross validation method to select τ. Specifically, we partition the whole data set into K subsets. We use $K-1$ subsets to train the model, and use the remaining subset to test the model in terms of the prediction error. We choose τ to be

$$\hat{\tau} = \arg\min_{\tau} \sum_{k=1}^{K} \sum_{j} (y_{kj} - \hat{y}_{-kj}(\tau))^2, \tag{7}$$

where y_{kj} is the j-th observation in the k-th subset, and $\hat{y}_{-kj}(\tau)$ is the value predicted by the model selected by the other subsets.

We can also adopt a Monte Carlo version of K-fold cross validation. We randomly divide the n observations into two subsets with n/K observations and $(K-1)n/K$ observations respectively. The subset with n/K observations is treated as the testing set and the subset with $(K-1)n/K$ observations is used for learning the model. After N replications, a loss function, essentially the same as (7), can be defined, and the value of τ is chosen by minimizing the loss function.

For small n large p problems, we may also choose ρ, the prior probability of excluding a variable, by the above cross-validation scheme.

3 Small *n* and large *p* problems

During the last decade, the small n and large p problems have received increasingly more attention. One example is the gene selection problem in microarray experiments, where the number of candidate genes, p, is much larger than the number of available samples, n. Based on the sparsity assumption, there are only a few active genes in the candidate pool, and the variable selection approach is widely used for searching possible candidate genes. Yi et al. (2003) modified the stochastic search variable selection procedure for identifying multiple quantitative trait loci. Lee et al. (2003) proposed a Bayesian gene selection method, which is similar to the idea of stochastic search variable selection. In addition to the gene selection problem, signal representation can also be considered as a small n large p model selection problem, where a signal such as an image is represented by a linear superposition of basis functions or basis elements selected from an overcomplete dictionary. Here "overcomplete" means that the number of basis functions is larger than the size of the signal. Wolfe et al. (2004) studied such a problem using Bayesian variable selection methods. In this section, we shall apply stochastic matching pursuit algorithm to small n large p problems.

3.1 Simulation studies

Shao and Chow (2007) studied the small n and large p problem in microarray experiments, and proposed a variable screening procedure to eliminate inactive variables. Based on the sparsity assumption, their procedure employs a positive decreasing sequence $\{a_n\}$ such that $a_n \to 0$ as $n \to \infty$. For a fixed n, the i-th variable is screened out if $|\hat{\beta}_i| \le a_n$. Shao and Chow (2007) used the ridge regression estimator:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X} + h_n I_p)^{-1}\mathbf{X}'Y = R_D\mathbf{X}'Y,$$

where I_p is the $p \times p$ identity matrix, h_n is the ridge parameter and $R_D = (\mathbf{X}'\mathbf{X} + h_n I_p)^{-1}$. R_D is similar to A_γ in stochastic search variable selection, because

$$R_D = (\mathbf{X}'\mathbf{X} + D^{-1} I_p D^{-1})^{-1},$$

where D is a diagonal matrix whose elements are equal to $1/\sqrt{h_n}$. Shao and Chow (2007) proved that their procedure is asymptotically consistent. They also pointed out that their idea is similar to that of the Lasso method (Tibshirani 1996).

Shao and Chow (2007) considered two cases of n and p: $(n, p) = (50, 200)$ and $(100, 400)$, and the true parameter vector of the 5 true active variables is set to be

$$\beta = (3, -3.5, 4, -2.8, 3.2, 0, \ldots, 0)'.$$

The variables X_i's are independently generated from the multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix I_n. In addition to the independent structure as in Shao and Chow (2007), we also consider the dependent structure by adding a common factor, G, so that

$$X_i = G_i + kG,$$

where k is a pre-specified constant, G_i's and G are independently generated from the multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix I_n. Two different values of k are considered: $k = 0$ and 1. When $k = 1$, the correlation between any two variables is 0.5. Then the response vector is generated according to (1) with β, and the error term ϵ is also independently generated from the multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix I_n.

In our simulation studies, there are two designs for (n, p), and two designs on the distributions for the variables. So there are 4 different combinations. For each combination, we perform a Monte Carlo study with 100 replications. For each replication, X_i's and ϵ are re-generated independently, and the response vector Y is also re-computed. The parameter τ in stochastic matching pursuit is selected from a pre-specified candidate set, $\mathcal{A}$, according to (7) by 5-fold cross validation. As mentioned in Sect. 2.5, the larger τ is, the more parsimonious the selected model is. Thus we

Table 1 Results based on 100 replications with $(n, p) = (50, 200)$

k	Method		Number of selected variables										
			≤ 2	3	4	5	6	7	8	9	10	>10	# of corr. sel.
0	SC	f_1	1	4	6	14	14	20	18	9	10	4	
		f_2	0	0	0	1	1	10	13	9	9	3	46
	Lasso_CV	f_1	0	0	0	17	5	10	14	6	4	44	
		f_2	0	0	0	17	5	10	14	6	4	44	100
	SMP($\hat{\tau}$)	f_1	0	0	0	94	4	2	0	0	0	0	
		f_2	0	0	0	94	4	2	0	0	0	0	100
1	SC	f_1	1	6	10	21	10	19	13	12	4	4	
		f_2	0	0	0	2	2	9	8	6	4	4	35
	Lasso_CV	f_1	0	0	0	0	1	0	1	4	8	86	
		f_2	0	0	0	0	1	0	1	4	8	86	100
	SMP($\hat{\tau}$)	f_1	0	0	0	97	3	0	0	0	0	0	
		f_2	0	0	0	97	3	0	0	0	0	0	100

Note: f_1 is the frequency (in 100 replications) of the number of selected variables, and f_2 is the frequency (in 100 replications) of including all 5 true active variables. "SC" is used to denote the results obtained by the screening method of Shao and Chow (2007). "Lasso_CV" means the results selected by the Matlab code, `lasso_cv`. "SMP($\hat{\tau}$)" means the stochastic matching pursuit procedure whose τ is chosen by the 5-fold cross validation

would suggest choosing the larger value of τ for small n large p problems due to the sparsity assumption. For $(n, p) = (50, 200)$, we set $\mathcal{A} = \{80, 120, 160, 220\}$, and, when we study the case of $(n, p) = (100, 400)$, $\mathcal{A}$ is chosen as $\{100, 150, 200, 250\}$. For each replication, we run stochastic matching pursuit for $5{,}000 \times p$ iterations. After discarding the first $3{,}000 \times p$ iterations, we take the 2,000 samples by using every pth sample from the remaining $2{,}000 \times p$ posterior samples for variable selection. For comparison, we also code the screening method of Shao and Chow (2007), where we set $h_n = n^{2/3}$ and $a_n = n^{-1/6}$, the same as those in Shao and Chow (2007). In addition to the above two methods, a Matlab implementation of the homotopy/Lars-Lasso algorithm for tracing the regularization path of the L1-penalized squared error loss is available at http://www.stat.berkeley.edu/twiki/Research/YuGroup/Software. In this tool-box, the stopping criterion for this Lasso Matlab code is defined as

$$\|\mathbf{X}'(Y - \mathbf{X}'\hat{\beta})\|_\infty < b,$$

where b is a pre-specified threshold and its default value is 10^{-8}. In this tool-box, the function `lasso_cv` is used to fit the parameters of a linear model by using the Lasso and K-fold cross validation. Thus we apply this Matlab implementation as a selection procedure, and active variables are selected if the corresponding coefficients are non-zero. Here the value of b is chosen from $\{2 \times 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-8}\}$ and K is set to be 10. Then the "best" results, i.e. the minimal number of selected variables, would be recorded among selection results for different values of b.

The variable selection results are shown in Tables 1 and 2. In the tables, SC means the method of Shao and Chow (2007), and SMP($\hat{\tau}$) means the stochastic matching pursuit procedure whose τ is chosen by the 5-fold cross validation method. Lasso_CV means the Matlab code, `lasso_cv`, with $b \in \{2 \times 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-8}\}$. In these two tables, we use two sets of numbers to report the performance of a variable selection method. The first one, denoted by f_1, is the frequency of the number of the selected variables, that is, how many times we select 1 variable, how many times we select two variables, and so on. For instance, Table 1 tells us that when $k = 0$, stochastic matching pursuit selects five variables 94 times per 100 repetitions; selects six variables four times per 100 repetitions, and selects seven variables in the other two replications. The second set of numbers, denoted by f_2, is the frequency of including all the true variables, that is, how often the selected model includes all of the true active variables. For instance, Table 1 tells us that all the 94 times that stochastic matching pursuit selects five variables, all of the 94 times, it selects the five true variables. Each of the four times it selects six variables, the selected six variables also include the five true variables. We also report the frequency (in 100 simulation runs) of including all five true variables in the last column of the table.

From both tables, we can make the following observations.

- For the screening method of Shao and Chow (2007), it is possible that some of the five true variables are not selected in the model. For example, for the case of $(n, p) = (50, 200)$ and $k = 0$, SC includes all five true variables

Table 2 Results based on 100 replications with $(n, p) = (100, 400)$

k	Method		Number of selected variables										
			≤2	3	4	5	6	7	8	9	10	>10	# of corr. sel.
0	SC	f_1	0	3	9	49	22	13	3	1	0	0	
		f_2	0	0	0	41	19	13	3	1	0	0	77
	Lasso_CV	f_1	0	0	0	75	6	5	1	0	2	11	
		f_2	0	0	0	75	6	5	1	0	2	11	100
	SMP($\hat{\tau}$)	f_1	0	0	0	95	5	0	0	0	0	0	
		f_2	0	0	0	95	5	0	0	0	0	0	100
1	SC	f_1	0	1	11	35	28	13	9	2	1	0	
		f_2	0	0	0	33	28	13	8	2	1	0	85
	Lasso_CV	f_1	0	0	0	4	5	3	7	7	7	67	
		f_2	0	0	0	4	5	3	7	7	7	67	100
	SMP($\hat{\tau}$)	f_1	0	0	0	98	2	0	0	0	0	0	
		f_2	0	0	0	98	2	0	0	0	0	0	100

Note: f_1 is the frequency (in 100 replications) of the number of selected variables, and f_2 is the frequency (in 100 replications) of including all 5 true active variables. "SC" is used to denote the results obtained by the screening method of Shao and Chow (2007). "Lasso_CV" means the results selected by Matlab code, `lasso_cv`. "SMP($\hat{\tau}$)" means the stochastic matching pursuit procedure whose τ is chosen by the 5-fold cross validation

in the final model 46 times. When $(n, p) = (100, 400)$ and $k = 0$, the frequency for including all true variable is $77/100$.

- For the selection results by Lasso_CV, all five true variables are included in the final model. However, there can be problem with over-selection. For example, for the case of $(n, p) = (50, 200)$ and $k = 1$, Lasso_CV selects more than 10 variables in the model in 86 replications.
- For both SC and Lasso_CV, their performance is improved when n is larger. For the screening method of Shao and Chow (2007), the frequency that the five true variables are included in the model for $n = 100$ is higher than the frequency for the case with $n = 50$, regardless of the value of k. For the results of the Lasso type method, when $n = 50$, the average numbers of over-selected variables are 6.61 for $k = 0$ and 13.61 for $k = 1$, but when $n = 100$, the average numbers are 1.9 for $k = 0$ and 10.81 for $k = 1$. Thus we believe that the consistent property for both methods still holds.
- For stochastic matching pursuit, the frequency of identifying the true model, i.e., containing only the five true variables, is always larger than or equal to $94/100$. There exist a few cases where more than five variables are selected.

The above simulation results suggest that stochastic matching pursuit performs fairly well. Although it may select more variables into the model, it always includes the five true variables into the model.

3.2 Image representation

Wolfe et al. (2004) applied the Bayesian variable selection method to represent the time-frequency surface, and this surface is modeled as the Gabor regression model, i.e.

$$f = \sum_i c_i g_i + \varepsilon,$$

where g_i's are the Gabor basis functions, c_i's are the unknown coefficients, and ε is the white noise. A stochastic search variable selection type procedure is used to infer the unknown coefficients, c_i.

Given a grid $\mathcal{X} = \{(x_1, x_2) | x_1 \in \{22, 24, \ldots, 40\}$ and $x_2 \in \{7, 9, \ldots, 25\}\}$, the Gabor basis function is defined as

$$g(u, v) = \exp\left[-\frac{1}{2}(\sigma_u u^2 + \sigma_v v^2)\right] \cos\left[\frac{2\pi u}{\lambda} + \varphi\right],$$

$$u = u_0 + x_1 \cos\theta - x_2 \sin\theta,$$

$$v = v_0 + x_1 \sin\theta - x_2 \cos\theta,$$

where (x_1, x_2) is the coordinate of $\mathcal{X}$; u_0, v_0, σ_u and σ_v are user chosen parameters of a two-dimensional Gaussian window satisfying the relationships $\sigma_v = \sqrt{2}\sigma_u$; $\lambda = \sqrt{2\pi}\sigma_u$ and φ are parameters of a sinusoidal grating, and θ is the angle between the x_1-axis of the image and the u-axis of the Gabor functions. Here $(u_0, v_0) \in \mathcal{X}$, and we set $\varphi = 0$, $\sigma_u = 1$ and $\theta \in \{0, 3/8\pi\}$. Thus, we have 200 Gabor basis functions in total whose norms are all equal to 1 on $\mathcal{X}$. For simplicity, we use $X_1, \ldots, X_{200}$ to index all the basis functions, and X_i is a 100×1 vector, $i = 1, 2, \ldots, 200$. Note here the sample correlation structure of these basis functions are not the same as the constant correlation structure in Sect. 3.1. A part of this correlation structure is shown in Appendix. The response is generated by

$$Y = 7X_{17} - 7X_{71} + 7X_{161} - 7X_{177} + \varepsilon,$$

Table 3 Posterior inclusion probabilities of selected basis functions

Selected bases						
X_{17}	X_{71}	X_{73}	X_{161}	X_{177}	SNR1	SNR2
0.9957	0.5243	0.6317	0.6933	1.0000	0.373	0.080

where $\varepsilon \sim N_{100}(\mathbf{0}, I_{100})$. Thus there are four active variables, and $(n, p) = (100, 200)$. So this is a small n and large p selection problem.

For stochastic matching pursuit, the parameter τ is chosen from $\mathcal{A} = \{50, 100, \ldots, 300\}$ by Monte Carlo cross validation with 100 replications, and at each replication, we randomly choose $n/5$ observations as the testing set and $4/5n$ as the training set. The resulting $\hat{\tau}_{SMP} = 50$. We iterate 10,000 × p times. Then we discard the first 7,000 × p draws, and sample the remaining 3,000 × p draws by taking every pth sample to compute the posterior inclusion probabilities. Table 3 shows the variables selected by our method via the median probability criterion. Here our procedure selects all correct variables and X_{73}. The sample correlation of X_{71} and X_{73} is 0.5893. Two signal to noise ratio indices are computed, and SNR1 and SNR2 are defined as

$$\mathrm{SNR1} = \|\hat{Y} - Y\|^2 / \|Y\|^2, \quad \text{and}$$
$$\mathrm{SNR2} = \|\hat{Y} - Y_{\mathrm{true}}\|^2 / \|Y_{\mathrm{true}}\|^2,$$

where $\hat{Y} = X_{\mathrm{in}}\hat{\beta}_{\mathrm{in}}$, X_{in} is the set of X_i's that are included in the model, $\hat{\beta}_{\mathrm{in}}$ is the corresponding Bayesian parameter estimate, and Y_{true} is the original response without noise. The smaller SNR2 is, the better the model we find. Since SNR2 is small (0.080), we believe that the stochastic matching pursuit procedure performs well in this example.

3.3 Comparison with componentwise Gibbs sampler

The componentwise Gibbs sampler is less computationally expensive than the stochastic matching pursuit algorithm. However, as we mention in Sect. 2, the componentwise Gibbs sampler might not perform well when the variables are highly correlated and the residual variance is small.

Using the Gabor regression model, we compare stochastic matching pursuit with the componentwise Gibbs sampler. Here the response is generated by

$$Y = 7X_{35} + 7X_{36} + 7X_{37} - 7X_{71} + 7X_{165} - 7X_{175} + \varepsilon,$$

where $\varepsilon \sim N_{100}(\mathbf{0}, \sigma_T^2 I_{100})$, and $\sigma_T = 10^{-2}$. Thus there are only six active variables and these active variables are highly correlated with the other variables, for example, the sample correlation between X_{35} and X_{36} is 0.86. Here we fix $\tau = 100$. For each procedure, we run 10,000 iterations and use the last 3,000 iterations for computing posterior inclusion probabilities. The stochastic matching pursuit can select the true model, i.e. $X_{35}, X_{36}, X_{37}, X_{71}, X_{165}, X_{175}$, but the componentwise Gibbs sampler also selects 18 other variables in addition to the five true variables, $X_{35}, X_{37}, X_{71}, X_{165}, X_{175}$. This difference may be caused by the small variance. For the cases with larger variance, the componentwise Gibbs sampler may work well.

In addition to comparing the selection results, the computational cost is also measured. Here we run the Matlab code for both algorithms on a PC with 3.20 GHz Pentium 4 CPU. The CPU times are measured for 10,000 × p iterations of the stochastic matching pursuit and 10,000 × 100p iterations of the Gibbs sampler. The times are 10676.51 and 15551.66 seconds for stochastic matching pursuit and componentwise Gibbs sampler respectively.

4 Comparison with related methods

4.1 Computational cost

George and McCulloch (1993) proposed to implement the stochastic search variable selection procedure via the Gibbs sampling scheme. However, there is a high computational cost for sampling the whole coefficient vector β from a multivariate normal distribution because an inverse matrix is involved in this step. Here we code the stochastic search variable selection procedure in Matlab to compare the computational cost of stochastic search variable selection with that of stochastic matching pursuit. We run both methods on a PC with 3.20 GHz Pentium 4 CPU for 10,000 iterations of the stochastic search variable selection procedure and 10,000 × p iterations of the stochastic matching pursuit. Two large n and small p problems with $(n, p) = (60, 5)$ and (60, 10) are used here, and the active variables are two and six respectively. The selection results of the stochastic search variable selection and the stochastic matching pursuit are similar to each other. To save the space here, we do not show the selection results. Table 4 shows the CPU times of the two algorithms. For the situations with $(n, p) = (60, 5)$ and (60, 10), i.e., where there are a small number of candidate variables, stochastic matching pursuit takes slightly more time than stochastic search variable selection. However, if we increase p as shown in the example with $(n, p) = (200, 100)$, stochastic matching pursuit takes much less time than stochastic search variable selection.

Table 4 CPU times (in seconds) for 10,000 iterations of stochastic search variable selection and stochastic matching pursuit

	SMP	SSVS
$n = 60$ and $p = 5$	14.6 s	9.3 s
$n = 60$ and $p = 10$	39.5 s	20.8 s
$n = 200$ and $p = 100$	2016.6 s	7266.4 s

Note: "SMP" means stochastic matching pursuit and "SSVS" means stochastic search variable selection

4.2 Prior assumptions

Different prior assumptions for β will lead to different Bayesian variable selection approaches. Following the prior assumptions in George and McCulloch (1993), the prior of $\beta_i|\gamma_i = 1$ is set to be $N(0, \tau_i^2)$, which is a nonconjugate prior. In contrast, Smith and Kohn (1996) used the conjugate prior for the coefficient vector by setting $\beta_\gamma \sim N(0, c\sigma^2(\mathbf{X}'_\gamma \mathbf{X}_\gamma)^{-1})$, where c is a pre-specified constant, and β_γ and $\mathbf{X}_\gamma$ are the components of β and the columns of $\mathbf{X}$ such that the corresponding γ_i's are equal to 1. George and McCulloch (1997) have discussed the relationship between conjugate and nonconjugate priors in the stochastic search variable selection approach.

With the conjugate prior of β, the posterior distribution $[\gamma|Y]$ can be obtained by integrating out β and σ. For example, in (2.4) of Smith and Kohn (1996), the posterior distribution of γ is

$$P(\gamma|Y) \propto (1+c)^{-q_\gamma/2} S(\gamma)^{-n/2} \prod_{i=1}^{p} \pi_i^{\gamma_i}(1-\pi_i)^{1-\gamma_i},$$

where q_γ is the number of selected variables, $S(\gamma) = Y'Y - \frac{c}{1+c} Y'\mathbf{X}_\gamma(\mathbf{X}'_\gamma\mathbf{X}_\gamma)^{-1}\mathbf{X}'_\gamma Y$ and $\pi_i = P(\gamma_i = 1) = 1 - p_i$. Then a Gibbs sampler can be implemented to generate the posterior samples of $[\gamma|Y]$. However, an inverse matrix computation, $(\mathbf{X}'_\gamma\mathbf{X}_\gamma)^{-1}$, is involved when we sample $\gamma_i|Y, \{\gamma_k, k \neq i\}$. Thus, the computational complexity would be increased when q_γ becomes larger. Numerical instability might also be a problem for highly correlated variables.

We code the algorithm of Smith and Kohn (1996) using Matlab by fixing $\pi_i = \rho = 1/2$, and we conduct two simulation studies for comparison with stochastic matching pursuit.

We apply this Matlab code to the image representation problem in Sect. 3.2. In their algorithm, c is a pre-specified parameter, and Smith and Kohn (1996) suggested that the value of c is in the range $10 \leq c \leq 1{,}000$ when the norm of X_i's are all equal to 1. Following the suggestion in Smith and Kohn (1996), we set c to be 100. Here we iterate the code 10,000 times and the selection result is obtained after discarding the first 7,000 iterations. The final model is selected based on the median probability criterion. However, the selection result is not very good, while three of the true active variables are selected, the other 96 variables are also included. Thus it also causes the warning message about the singularity of $\mathbf{X}'_\gamma\mathbf{X}_\gamma$. When we set $c = 500$ and re-run the code, the selected result is similar to what we obtain for $c = 100$. The result for SMP is shown in Sect. 3.2, where SMP selects five variables included the four true active variables.

Another simulation with $(n, p) = (50, 300)$ is also studied. In this simulation, the variables X_i, $i = 1, \ldots, p$, are independently generated from the multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix I_n, and the true response is set to be

$$Y = 3X_1 + 3X_2 + \cdots + 3X_{10} + \varepsilon, \tag{8}$$

where ε also comes from a multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix I_n. For the algorithm of Smith and Kohn (1996), two different values of c, $c = 500$ and $c = 1{,}000$, are used. The algorithm is iterated 5,000 times and the last 2,000 samples are kept. The median probability criterion is used for variable selection. Selection result with both $c = 500$ and $c = 1{,}000$ are similar and we only report the results with $c = 1{,}000$. For $c = 1{,}000$, 51 variables are selected but only six true active variables are included. Because too many variables are included in the model, the singularity of $\mathbf{X}'_\gamma\mathbf{X}_\gamma$ becomes a problem. In comparison, the stochastic matching pursuit with $\tau = 250$ selects 10 variables and all of them are the true active variables. We repeat this simulation five times. The stochastic matching pursuit with $\tau = 250$ identifies 10 true variables in each replication. The algorithm of Smith and Kohn (1996) might include the 10 true variables in the final model, but over-selection of variables and the singularity problem of $\mathbf{X}'_\gamma\mathbf{X}_\gamma$ still exist.

These two simulation studies show that stochastic matching pursuit is more stable. In stochastic matching pursuit, we use more information to decide whether a variable is to be included (or deleted), because, based on the current model, the added variable is chosen from the inactive set of variables according to their likelihood ratios. Another reason that stochastic matching pursuit is more stable may be related to the inverse matrix of $\mathbf{X}'_\gamma\mathbf{X}_\gamma$ and the choice of c. In our experience, once $\mathbf{X}'_\gamma\mathbf{X}_\gamma$ is singular, i.e. too many variables are included in the model, then γ might not be able to progress beyond the current status or might be unstable in the model space. In fact, the key problem is about the choice of c. Just like the prior parameter, τ, in stochastic matching pursuit, c should be larger for larger p, especially when $p > n$, because the larger c is, the smaller the probability that $\gamma_i = 1$ is. Thus, when c is too small, $\mathbf{X}'_\gamma\mathbf{X}_\gamma$ tends to be singular for small n and large p problems. To illustrate this point, we first run the algorithm of Smith and Kohn (1996) for the cases of $(n, p) = (50, 20)$ and $(50, 50)$ with the true

Table 5 Results based on 100 replications with $(n, p) = (50, 200)$ for full Bayesian SMP

k	Method		Number of selected variables										
			≤ 2	3	4	5	6	7	8	9	10	>10	# of corr. sel.
0	B-SMP	f_1	0	0	0	68	27	5	0	0	0	0	
		f_2	0	0	0	68	27	5	0	0	0	0	100
1	B-SMP	f_1	0	0	0	73	18	9	0	0	0	0	
		f_2	0	0	0	73	18	9	0	0	0	0	100

Note: f_1 is the frequency (in 100 replications) of the number of selected variables, and f_2 is the frequency (in 100 replications) of including all 5 true active variables. "B-SMP" is used to denote the results obtained by "full" Bayesian stochastic matching pursuit procedure

model as in (8). The algorithm successfully identifies the true model with 10 variables regardless of whether $c = 10$ and 100, because $\mathbf{X}'_\gamma \mathbf{X}_\gamma$ is always invertible. Next, using the same model in (8), (n, p) is set to be $(50, 100)$. The result with $c = 10$ fails to identify the model due to the singularity of $\mathbf{X}'_\gamma \mathbf{X}_\gamma$, but when we choose $c = 100$ or 1,000, we can obtain the true model successfully. However, when $p = 300$, the algorithm of Smith and Kohn (1996) fails to identify the true model even when we set $c = 1{,}000$. The stochastic matching pursuit with $\tau = 250$ successfully selects the true model for (8), when $n = 50$ and $p = 20, 50, 100, 300$. Therefore, it seems that stochastic matching pursuit is more stable than the algorithm of Smith and Kohn (1996) for small n and large p problems. These selection results also show that the algorithm of Smith and Kohn (1996) is sensitive to the value of c for small n and large p problems, and c probably can be determined by a similar approach to what we use for τ.

As pointed out by one reviewer, the scheme of stochastic matching pursuit can also be applied to the model of Smith and Kohn (1996). In (2.3) of Smith and Kohn (1996), $P(Y|\gamma)$ is derived in closed form. We can define $\tilde{z}_i = P(Y|\gamma_i = 1, \{\gamma_k, k \neq i\})/P(Y|\gamma_i = 0, \{\gamma_k, k \neq i\})$ for $i = 1, \ldots, p$. Then we can replace $P(\gamma_i|\{\beta_k, k \neq i\}, Y)$ by $P(\gamma_i|\{\gamma_k, k \neq i\}, Y)$ and replace z_i by $\tilde{z}_i$. Thus, we can use the stochastic matching pursuit algorithm to sample from the posterior distribution $P(\gamma|Y)$ for the purpose of variable selection.

4.3 Full Bayesian approach

In this article, cross validation method is used to choose the proper value for the parameter τ in the prior distribution of β_i. From Tables 1 and 2, we can see that the stochastic matching pursuit procedure works well for small n and large p problems by coupling with cross validation method for the selection of τ. We can also implement the stochastic matching pursuit procedure in a full Bayesian treatment. Following the prior assumption of τ_i in Wolfe et al. (2004), we use an inverse gamma prior with parameters, κ and ξ, for τ, i.e. $IG(\kappa, \xi)$. Then we sample τ from its posterior inverse Gamma distribution in Step (VI). We apply this full Bayesian procedure to the small n and large p problem with $(n, p) = (50, 200)$ by setting $\kappa = 1$ and $\xi = 10$. The selection results are shown in Table 5. It seems that the selection results are not as good as those of the stochastic matching pursuit where τ is selected by cross validation. When we trace the values of τ in this full Bayesian approach, we find that most of τ are around 10 which might be too small for this case. We also try the other sets of (κ, ξ), for example, $(\kappa, \xi) = (1, 100)$. The selection results are slightly better than what we show in Table 5. Thus, how to select these two parameters, κ and ξ, is another problem for this procedure.

5 Conclusion

This article proposes a stochastic matching pursuit algorithm for Bayesian variable selection. Our experiments suggest that it performs well for both large n small p and small n large p problems. The algorithm combines the advantages of the original matching pursuit algorithm and the componentwise Gibbs sampler.

Compared with other Bayesian variable selection methods, the stochastic matching pursuit algorithm avoids the computation of inverse matrices, thus reduces the computational burden and produces stable selection results. Our simulation studies show that the proposed method compares favorably with existing variable selection methods in terms of the selection results.

It is possible to further improve the efficiency of the current version of the stochastic matching pursuit algorithm. The first scheme is blocking. Before running the algorithm, we find blocks of variables, where the variables within the same block are highly correlated. Different blocks can overlap with each other. Then within each iteration of the algorithm, instead of looking at all the inactive variables, we randomly choose a block, and apply the addition and deletion moves only for the variables within the block. This will greatly reduce computational cost. The last author once applied such a scheme for image modeling (Wu et al. 2002). The second scheme is switching, where we choose an active variable and an inactive variable that is highly correlated with this active variable, and then we switch their status by making the active variable inactive while making the inactive variable active.

For the small n large p problems, we also need to devise a method for tuning the parameter ρ, the prior probability of excluding a variable from the model. We leave this to future investigations.

Reproducibility Data and Matlab code for reproducing the experimental results reported in this paper can be downloaded at http://www.stat.nuk.edu.tw/Ray-Bing/selection_web/homepage.htm.

Acknowledgements The authors are grateful to the associate editor and the two reviewers for their insightful comments and useful suggestions. In particular, we have incorporated one reviewer's summary of our work into the abstract, and we have also incorporated the other reviewer's comment on applying our method to the model with conjugate prior into Sect. 4.2. We would also like to thank Dr. Ching-Kang Ing and Dr. Eric Jaehnig for helpful discussions. This work is partially supported by the National Science Council in Taiwan; National Center for Theoretical Sciences (South), Tainan, Taiwan, and NSF-DMS 0707055.

Appendix

References

Barbieri, M., Berger, J.O.: Optimal predictive model selection. Ann. Stat. **32**, 870–897 (2004)

Beattie, S.D., Fong, D.K.H., Lin, D.K.J.: A two-stage Bayesian model selection strategy for supersaturated designs. Technometrics **44**, 55–63 (2002)

Chipman, H.: Bayesian variable selection with related predictors. Can. J. Stat. **24**, 17–36 (1996)

Chipman, H., Hamada, M., Wu, C.F.J.: A Bayesian variable selection approach for analyzing designed experiments with complex aliasing. Technometrics **39**, 372–381 (1997)

Févotte, C., Godsill, S.J.: Sparse linear regression in unions of bases via Bayesian variable selection. IEEE Signal Process. Lett. **13**, 441–444 (2006)

George, E.I., McCulloch, R.E.: Variable selection via Gibbs sampling. J. Am. Stat. Assoc. **88**, 881–889 (1993)

George, E.I., McCulloch, R.E.: Approaches for Bayesian variable selection. Stat. Sin. **7**, 339–374 (1997)

Geweke, J.: Variable selection and model comparison in regression. In: Bernardo, J.M., Berger, J.O., Dawid, A.P., Smith, A.F.M. (eds.) Bayesian Statistics, vol. 5, pp. 609–620. Oxford Press, Oxford (1996)

Lai, T.-W.: Variable selection via MCMC matching pursuit. M.S. Thesis, Institute of Statistics, National University of Kaohsiung, Kaohsiung, Taiwan (2007)

Lee, K.E., Sha, N., Dougherty, E.R., Vannucci, M., Mallick, B.: Gene selection: a Bayesian variable selection approach. Bioinformatics **19**, 90–97 (2003)

Mallat, S.G., Zhang, Z.: Matching pursuit with time-frequency dictionaries. IEEE Trans. Signal Process. **41**, 3397–3415 (1993)

Shao, J., Chow, S.-C.: Variable screening in predicting clinical outcome with high-dimensional microarrays. J. Multivar. Anal. **98**, 1529–1538 (2007)

Smith, M., Kohn, R.: Nonparametric regression using Bayesian variable selection. J. Econom. **75**, 317–343 (1996)

Tibshirani, R.: Regression shrinkage and selection via the Lasso. J. R. Stat. Soc. B **58**, 267–288 (1996)

Wolfe, P.J., Godsill, S.J., Ng, W.J.: Bayesian variable selection and regularization for time-frequency surface estimation. J. R. Stat. Soc. B **66**, 575–589 (2004)

Table 6 A Part of correlation matrix of Gabor basis functions

	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}
X_{11}	1.0000	0.8641	0.5893	0.3105	0.1353	0.0117	0.0108	⋆	⋆	⋆
X_{12}		1.0000	0.3065	−0.0176	0.0102	0.0032	0.0003	⋆	⋆	⋆
X_{13}			1.0000	0.8641	0.5893	0.3105	0.1353	0.0117	0.0114	⋆
X_{14}				1.0000	0.3065	−0.0176	0.0102	−0.0032	−0.0004	⋆
X_{15}					1.0000	0.8641	0.5894	0.3105	0.1432	0.0118
X_{16}						1.0000	0.3066	−0.0176	0.0108	−0.0032
X_{17}							1.0000	0.8642	0.6180	0.3121
X_{18}								1.0000	0.3245	−0.0176
X_{19}									1.0000	0.9238
X_{20}										1.0000

⋆ is a symbol of the value less than 10^{-4}

Wu, Y.N., Zhu, S.C., Guo, C.: Statistical modeling of texture sketch. In: Proceedings of European Conference of Computer Vision, pp. 240–254 (2002)

Yi, N., George, V., Allison, D.B.: Stochastic search variable selection for identifying multiple quantitative trait loci. Genetics **164**, 1129–1138 (2003)