Noname manuscript No.
(will be inserted by the editor)

Accelerating a Gibbs Sampler for Variable Selection on Genomics Data with Summarization and Variable Pre-selection combining an Array DBMS and R

David Sergio Matusevich · **Wellington Cabrera** ·
Carlos Ordonez

Received: date / Accepted: date

Abstract Variable selection in high dimensional data is a challenging problem due to the exponential number of variable combinations, and Markov Chain Monte Carlo (MCMC) methods represent the state of the art to solve it. With genomics data this problem becomes even more difficult because there are generally more dimensions (variables) than points (records) leading to slow convergence and numerically unstable solutions. On the other hand, despite many alternative prototypes and languages, R remains a popular system to compute machine learning models. Unfortunately, R can be particularly slow with heavy matrix computations and the high number of iterations required by MCMC methods. Moreover, making R scale to large matrices, possibly beyond RAM, requires careful system integration. Recently, array DBMSs have opened the possibility of manipulating matrices of unlimited size. With such motivation in mind, we present algorithmic optimizations to accelerate the computation of variable selection in linear regression with the Gibbs sampler, a fundamental MCMC method. Such optimizations have the potential to accelerate other models. We study how to leverage the speed and scalability of the array DBMS to exploit our optimizations in R. We present a comprehensive experimental evaluation to assess time efficiency and model quality with a cancer data set containing RNA and miRNA variables to predict survival time. We show our optimized algorithm combining DBMS and R processing is significantly faster than R alone. We show our system allows fast joint analysis of RNA and miRNA variables, instead of analyzing them separately. Finally, we confirm our algorithm finds medically significant variables already identified in the biomedical literature. Our optimized MCMC method for the array DBMS can be easily called from R, leaving the final model within R runtime in RAM for further interpretation.

Keywords MCMC · Linear Regression · Variable Selection · Array DBMS · Genomics

D. S. Matusevich
University of Houston
Houston, TX 77204
USA
E-mail: matusevich@cs.uh.edu

W. Cabrera
E-mail: wcabrera@cs.uh.edu

C. Ordonez
E-mail: ordonez@cs.uh.edu

1 Introduction

Many diseases cause changes at the molecular level that drive their development and progression. Cancer, in particular, involves nuclear structure alterations, such as mutations, amplifications, copy number variations (alterations of the DNA of a genome that results in the cell having an abnormal or, for certain genes, a normal variation in the number of copies of one or more sections of the DNA), and other events that drastically change chromatin, nuclear body and chromosome organization (Schirmer and Jose, 2014). Recent advances in medical technology have enabled researchers to measure the molecular changes in messenger RNA (mRNA) (Debouck and Goodfellow, 1999; Duggan et al, 1999) and microRNA (miRNA), a non-coding RNA molecule involved in the regulation of gene expression (Brennecke et al, 2005). These changes can be used to track the course of the disease and provide clues to it's development. The problem lies in the fact that out of thousands of genetic markers only a few are believed to affect the outcome of the disease and predicting which of them is relevant is a pressing issue, since it could aid in the development of targeted therapies based on the genetic footprint of the patient.

In order to understand the relationship between the genetic markers and the dependent variable, researchers are interested in finding models that could standardize how they relate to one another. However, models containing too many parameters are likely to describe random noise instead of the underlying relationship between the variables. Such a model will generally have a poor predictive performance outside the training set and will exaggerate minor fluctuations in the sample data (Hastie et al, 2001; Sauerbrei et al, 2007). Therefore we are interested in finding smaller subsets of markers that produce accurate models that are, at the same time, easy to interpret (Rockova et al, 2012). In statistics and machine learning, the process of selecting such a subset of relevant features is called feature or variable selection (George and McCulloch, 1993; George, 2000; Guyon and Elisseeff, 2003).

Data mining problems usually consist of a large number of samples with just a relatively small number of variables. This is usually the case in marketing or web data, where the problem could have tens or hundreds of millions of data samples and a few hundred variables. High dimensional problems are very difficult to solve and manipulate, making it desirable to find the variables that are most related to the dependent variable. Genomics data differs from the typical problem in that while the data might have on the order of tens of thousands of variables, the researcher only counts with no more than a few hundred samples, since it is difficult and expensive to collect the data from real patients. This is commonly referred as a *small n, large d* problem. There are several technical and computational issues that arise when dealing with this kind of problem. Since the system is over-specified, the solutions cannot be uniquely determined. Data structures become too large to fit in main memory, requiring them to be kept in secondary storage. Therefore we must take into account not only the storage space required for the data set, but also we need to be able to safely and efficiently store the intermediate data structures required by the algorithm. As an example, in the case of a genomics data set, a typical covariance matrix could have in the order of hundreds of millions of elements that, depending on the algorithm, may be accessed hundreds or thousands of times.

The variable selection problem can be solved in different ways. For data sets with a relatively small number of variables, we could tackle the problem using a brute force or exhaustive approach. However, since the number of different subsets from a set of d variables is 2^d, and for data sets with thousands of variables, it is clearly not feasible to test all of them. Common approaches are iterative in nature. The learner usually start from an arbitrary solution and then progressively improves it. Hill climbing methods (Caruana and

Freitag, 1994), such as Forward Selection, Backward Elimination, Forward Step-wise Selection and Backward Step-wise Elimination are effective at finding good performing subsets of attributes but are prone to getting trapped in local maxima. This problem can be solved using Simulated annealing techniques and Genetic algorithms (Moore et al, 2001), however solutions are complex and typically require large high performance computing (HPC) clusters for optimal performance. Another kind of technique that combines feature selection and visualization is Targeted projection pursuit (Faith et al, 2006). However this method, while powerful, is mostly optimized towards visualization in two or three dimension plots, and not appropriate for finding sets of tens of dimensions.

We propose and evaluate a stochastic variable selection algorithm for linear regression on high dimensional genomics data sets, based on the Gibbs Sampler, a fundamental MCMC method, widely used Bayesian statistics and machine learning. Our main processing infrastructure was SciDB (Stonebraker et al, 2013), an array database system designed for science data management, working in tandem with R (Ihaka and Gentleman, 1996), the most popular open-source statistical software. SciDB implements an innovative array model for efficient storage and manipulation of large multidimensional arrays as well as efficient mathematical and relational operators on those arrays. Our optimized algorithm exploits sufficient statistics on the input data set, which take the form of matrices, extending our previous work, on the Gamma summarization operator (Ordonez et al, 2014b), to speed up the computation of complex matrix equations. Having a bioinformatics application in mind, we analyze a data set of gene expressions of Glioblastoma Multiforme (brain tumor) patients to predict their survival time after diagnosis. In summary, our main contributions are:

- The introduction of the first MCMC method optimized to run on an array DBMS using sufficient statistics,
- a pre-processing phase working entirely inside the DBMS, that reduces the data set to a much lower dimensionality using variable pre-selection, but without discarding important models,
- a library that can be called by R, without requiring changes to the R run-time, while at the same time it removing R's RAM and speed limitations,
- a careful experimental evaluation on a cancer data set.

2 Related Work

In this section we will provide a brief review of the prior art in variable selection, efficient solutions to this problem in relational databases, and some background on the system used for this work.

2.1 Methods for Variable Selection

The Variable Selection (VS) problem has been shown in (Natarajan, 1995) and (Garey and Johnson, 2002) to be NP-complete. As such, considerable attention has been devoted to finding approximate solutions by methods that converge in a reasonable amount of time. We will briefly review some of the most popular algorithms to calculate solutions to this problem.

Stepwise Regression

The basic idea behind Stepwise Regression (Hocking, 1976) is adding or deleting variables one at the time. Let us define the Residual Sum of Squares (RSS) as the sum of the squares of the deviations of predicted from actual empirical values of data.

$$\mathrm{RSS} = \sum_{i=1}^{N}\left(y_i - \alpha - \sum_j \beta_j x_{ij}\right)^2 \tag{1}$$

where $(\boldsymbol{x}^i, y_i)$ is the data set and N is the number of data points in the data set.

Forward selection (FS) starts with no variables and adds the variable that yields the largest single degree of freedom. This process continues until some stopping criterion is satisfied.

The variable i is added to the the p-term equation if

$$F_i = \max_i\left(\frac{\mathrm{RSS}_p - \mathrm{RSS}_{p+i}}{\hat{\sigma}^2_{p+i}}\right) > F_{\mathrm{in}} \tag{2}$$

In this equation RSS_p is the residual sum of squares for the p-term equation and the subscript $p+i$ refers to quantities calculated if the variable with index i is added to the p-term equation.

Backward elimination (BE), conversely, starts with all the variables being included and each step eliminates the variable with the smallest F-ratio. In other words, the variable i is eliminated if

$$F_i = \max_i\left(\frac{\mathrm{RSS}_{p-i} - \mathrm{RSS}_p}{\hat{\sigma_p}^2}\right) < F_{\mathrm{out}} \tag{3}$$

In this case, $p-i$ refers to quantities calculated when the variable i is deleted from the current p-term equation.

Hybrid methods use a mixture of FS and BE. The main criticism against stepwise methods is that they ignore variable combinations ((Mantel, 1970)). Furthermore, stepwise methods are computationally expensive and prone to overfitting (Caruana et al, 2004).

The Lasso Method

In the Lasso method (Tibshirani, 1996; Kolar et al, 2011) some of the coefficients are shrunk and others are set to 0 using an operator called the "Lasso" (Least Absolute Shrinkage and Selection Operator). The original technique used quadratic programming to find:

$$\left(\hat{\alpha}, \hat{\beta}\right) = \arg\min\left\{\sum_{i=1}^{N}\left(y_i - \alpha - \sum_j \beta_j x_{ij}\right)^2\right\}, \tag{4}$$

where $(\boldsymbol{x}^i, y_i)$ is the data set, $\sum_i x_{ij}/N = 0$, $\sum_i x_{ij}^2/N = 1$ (normalized set) and the β coefficients are subject to the condition $\sum_j |\beta_j| \leq t$ (t is a tuning parameter). Wainwright (2009) shows the necessary and sufficient conditions on the dimensionality of the vector β, the number of selected variables (sparsity index) k, and the number of observations N for which it is possible to find the set $\left(\hat{\alpha}, \hat{\beta}\right)$ using the Lasso. The Lasso uses a regularization term that enables finding sparse coefficient vectors, which is equivalent to the non-informative prior in our model specification. Our approach does not require such term.

Least Angle Regresssion

Least Angle Regression (LARS) (Efron et al, 2004) is a selection algorithm that can be modified to implement the Lasso and Forward Stagewise linear regression. Meinshausen and Bühlmann (2006) use a neighborhood estimate, coupled with the Lasso to estimate solutions to the variable selection problem in high dimensional data sets. Wasserman and Roeder (2009) study the general application of these methods to the high dimensional problem. Orthogonal matching pursuit (OMP) (Tropp, 2004; Tropp and Gilbert, 2007; Zhang, 2011) is a greedy algorithm that can be used for variable selection in signal processing. This algorithm can be proven to converge quickly to the signal of interest. However it is unknown if OMP converges on noisy measurements (Needell et al, 2008). OMP algorithms are useful to detect a hidden sparse signal, a somewhat similar problem to variable selection. That is, they can help identifying a signal where there is a significant fraction of dimension values that are zero. In contrast, our problem is more general: in microarray data there exist multiple signals, each with different variable combinations

Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) (Fukushima, 1992; Boyd et al, 2011) is a type of Augmented Lagrangian scheme that uses dual variables with partial updates. ADMM soles the constrained problem $\min_{x,y} f(x) + g(y)$ with the constraint $x = y$. While this can lead to highly parallel and efficient algorithms to select variables, ADMM assumes the cost function to minimize can be decomposed into a sum of two simpler functions (generally convex, but they can be non-convex, (Derbinsky et al, 2013)). This decomposition aspect is orthogonal to the data set being dense or sparse or the data set having more dimensions than points. Comparing ADMM and SSVS is an interesting issue for future work.

Other Methods and Applications

Guyon et al (2010) present a survey of several methods for variable selection and the premises behind the different methodologies and algorithms to achieve parsimonious models. Davies et al (2014) use Bayesian variable selection to predict antigenic variability and to identify sites that are important to the neutralization of the FMD virus (Foot-and-Mouth disease). The varying-coefficient model has been applied to the variable selection model in high dimensional data by Xue and Qu (2012). In this work the authors show that this method is able to select true models at a higher frequency than the LASSO and SCAD approaches, but their method is a local minimizer and combining it with other methods to provide a global minimum makes it more computationally intensive.

2.2 Learning Methods in Relational Database Systems

The efficient computation of analytic algorithms in a DBMS has received moderate attention. Ordonez (2010) shows several machine learning models that can be optimized by sufficient statistics using User Defined Functions (UDF). As we will see in Section 3.2, sufficient statistics are fundamental for the optimization of our Bayesian variable selection algorithm. Other models can also take advantage of sufficient statistics, for instance: Naive Bayes (Pitchaimalai et al, 2010) and Clustering with a mixture of Gaussians (Matusevich and

Ordonez, 2014). Our recent work (Cabrera et al, 2013; Ordonez et al, 2014a) also exploits sufficient statistics, along with additional optimizations applied to a row DBMS: hashing, variable pruning and integration with Intel Math Kernel Library. Interesting results are presented with a genomic data set. However this work has certain limitations: the data set needs to be loaded in memory and sufficient statistics cannot take advantage of multi-core architecture because the computation of the summarization is performed as part of the sampling step, in a Table Value Function.

The MADlib Analytics library (Hellerstein et al, 2012) is a comprehensive collection of supervised and unsupervised learning routines for relational database systems. MADlib also provides linear regression as a subroutine. However MADlib does not include methods for the problem of variable selection. Furthermore MADlib does not solve the regression problem on data sets with very high number of variables.

Our recent work (Ordonez et al, 2014b) presents a parallel, scalable algorithm for the computation of sufficient statistics. Our initial work in high dimensional variable selection is found in Cabrera et al (2013). These papers are among the first to solve linear regression and variable selection in such a high dimensional data set, combining processing in R and a DBMS.

2.3 SciDB-R Integration

SciDB (Stonebraker et al, 2013) also provides a set of analytic functions: Singular Value Decomposition (SVD), quartiles, average, standard deviation but it lacks complex analytics (classification, clustering). RIOT (Zhang et al, 2010) extends R, making I/O more efficient and scalable for larger data sets, but does not interact well with large matrices. Recently, Taft et al (2014) presented a benchmark comparing execution times in SciDB with R, a column DBMS and a row DBMS, running complex analytics: regression, correlation, bi-clustering and SVD. SciDB showed good performance in one node, but poor scalability when running on several nodes. Several machine learning packages for the R language address the variable selection problem. BayesVarSel and BMS are R packages focused in Bayesian methods for variable selection. Regarding Monte Carlo methods in SciDB, in (Ge et al, 2011) the authors use a Monte Carlo query processing algorithm to query uncertain data. However this work is significantly different from ours, since we use Monte Carlo for machine learning.

3 Definitions and Preliminaries

3.1 Linear Regression

Linear regression is a machine learning model that explains the relationship between a scalar *dependent variable* and a set of *explanatory variables* (or *independent variables*). Let $X = \{x_1, \ldots, x_n\}$ be a set of n data points with d explanatory variables and $Y = \{y_1, \ldots, y_n\}$ be a set of numbers such that each x_i is associated with y_i. If we represent the data set by matrices Y ($[1 \times n]$) and X ($[d \times n]$), the linear regression model can be expressed as:

$$Y = \beta^T \boldsymbol{X} + \varepsilon \tag{5}$$

where $\beta = [\beta_0, \ldots, \beta_d]$ is the vector of regression coefficients; ε represents the error and has a Gaussian distribution and $\boldsymbol{X}$ is X augmented with a row of n 1s stored in an extra column, X_0. The vector β is usually estimated using the ordinary least squares method (Puntanen and Styan, 1989). Table 1 summarizes the notation of the linear regression model.

Table 1 The Linear Regression Model.

Matrix	Dimensions	Name
Y	$1 \times n$	Dependent Variable
X	$d \times n$	Independent Variable
$\boldsymbol{X}$	$(d+1) \times n$	Augmented Independent Variable
β	$(d+1) \times 1$	Regression Coefficients
ε	$1 \times n$	Gaussian Error

3.2 Variable Selection Problem Definition

Variable selection is the search for the best subsets of variables (submodels) that are good predictors of Y (Mitchell and Beauchamp, 1988). The assumption of this search is that the data contains variables that are redundant and can be safely excluded from the model, since they provide little relevant information. There are many reasons to undertake this search:

- To express the relationship between the dependent variables and the explanatory variables in the simplest way possible (Mitchell and Beauchamp, 1988)
- To identify which variables are important and which are negligible predictors (Mitchell and Beauchamp, 1988)
- To facilitate data visualization and model interpretation (Guyon and Elisseeff, 2003)

Finding which subsets are the most appropriate is not computationally trivial, since there are 2^d potential combinations of variables.

In this article the set of selected variables is represented by a d-dimensional vector $\gamma \in \{0,1\}^d$, such that $\gamma_j = 1$ if the variable j is selected and $\gamma_j = 0$ otherwise. We denote by M_γ the model that selects k of the d variables, corresponding to the vector γ. Given γ we can easily find how many variables were selected by performing the dot product $k = \gamma^T \cdot \gamma$. Throughout this paper we will use γ as an index on selected variables such as β_γ and $\boldsymbol{X}_\gamma$ to denote that these quantities are projected using a particular set of selected variables.

3.3 Gibbs sampler

As mentioned before, an exhaustive search on the 2^d subsets of variables is impractical for even a moderately large d. While many techniques have been developed (Hocking, 1976) to deal with such a problem, a Bayesian approach provides information about the prior probabilities as an added bonus. Since it is very difficult to produce a solution to such a problem, several authors (George and McCulloch, 1993; Mitchell and Beauchamp, 1988), exploit Markov Chain Monte Carlo (MCMC) techniques, such as the Metropolis-Hastings Method and the Gibbs sampler, to obtain reasonably accurate models by sampling from the posterior distributions.

The Gibbs sampler, the approach taken in this work, uses the posterior probability $\pi(\gamma|X,Y)$ as a criterion for the selection of promising sets of variables. The Gibbs sampler is a MCMC method to obtain a sequence of observations approximated from the posterior probability. This sequence of observations is characterized by a vector $\gamma^{[i]}$, that describes the variables selected at iteration i of the sequence. $\gamma^{[i]}$ is obtained from the previous vector, $\gamma^{[i-1]}$, as follows:

For every variable x_j the normalized probabilities of $p(\gamma_j^{[i]} = 0)$ and $p(\gamma_j^{[i]} = 1)$ are calculated. Based on these probabilities either $\gamma_j^{[i]} = 0$ or $\gamma_j^{[i]} = 1$ is chosen by sampling, and the

j position of vector $\gamma^{[i]}$ is updated. After N iterations we obtain the Markov chain sequence $\gamma^{[0]},\ldots,\gamma^{[N]}$. To avoid introducing a bias into the prior computation due to initial instabilities, the first B iterations of the Markov chain (*burn-in period*) are usually discarded. After the burn-in, it is assumed that we have reached a stable distribution (the process becomes ergodic) and we can safely sample the priors. Finding the number of iterations that need to be discarded is not a trivial problem (Jones and Hobert, 2004). In the present work B was fixed empirically to a large number, determined experimentally.

We base the computation of $\pi(\gamma|X,Y)$ on the Zellner G-prior (Zellner, 1986)), which simplifies the Gibbs priors specification and sampling from the posteriors. Zellner's G-prior relies on a conditional Gaussian prior for β and an improper (Jeffreys) prior for σ^2 (Marin and Robert, 2007), with parameters c and $\widetilde{\beta}$.

$$\beta|\sigma^2,X \sim \mathcal{N}_{k+1}\left(\tilde{\beta},c\sigma^2\left(XX^T\right)^{-1}\right) \tag{6}$$

$$\sigma^2 \sim \pi\left(\sigma^2|X\right) \propto \sigma^{-2} \tag{7}$$

Our approach uses a uniform prior probability for γ: $\pi(\gamma|X) = 2^{-d}$ (George and McCulloch, 1993; Marin and Robert, 2007).

Under Zellner's prior the parameter c also influences the number of variables selected by the model: large c values lead to small models, whereas small values of c promote saturated models (Liang, 2008). Since we aim to solve problems with very large dimensionality ($d \gg n$) we will use a high c value, in order to select smaller sets of variables. We calculate the posterior probability $\pi(\gamma|X,Y)$ using the marginalized posterior distribution (Marin and Robert, 2007):

$$\pi(\gamma|X,Y) \propto \frac{1}{(c+1)^{\frac{k+1}{2}}}\left(YY^T - \frac{cY\boldsymbol{X}_\gamma(\boldsymbol{X}_\gamma\boldsymbol{X}_\gamma^T)^{-1}\boldsymbol{X}_\gamma^T Y^T + \tilde{\beta}_\gamma^T\boldsymbol{X}_\gamma\boldsymbol{X}_\gamma^T\tilde{\beta}_\gamma}{c+1}\right)^{-n/2} \tag{8}$$

3.4 Extended Sufficient Statistics for Linear Regression

Throughout this work we will make use of the concept of sufficient statistics. Sufficient statistics are multidimensional functions that summarize the properties of the data set or of a portion of the data set. These functions are of importance since they can be calculated using simple aggregations and they reduce considerably the number of times the data set must be read. Since we are operating on the assumption that the data set does not fit in memory, reading the data set from secondary storage as few times as possible is a key ingredient to any time optimization. The statistics used in this work are:

n, the number of input d-vectors,
$L = \sum_i y_i$, stores the sum of the data points, and
$Q = \sum_i y_i.y_i^T$, stores the sum of the squares of the elements of the data set.

For computing linear models we consider an extension of sufficient statistics. Let $\boldsymbol{X} = [1,X]$ be the *augmented* X matrix; since X is a $d\times n$ matrix, $\boldsymbol{X}$ is a $(d+1)\times n$ matrix such that:

$$\boldsymbol{X}_{i,j} = \begin{cases} 1 & \text{if } i=0 \\ X_{i,j} & \text{if } i>0 \end{cases} \tag{9}$$

Table 2 Summarization matrices

Name	Size
L	$d \times 1$
Q	$d \times d$
Γ	$(d+2) \times (d+2)$

and let $\boldsymbol{Z} = [\boldsymbol{X}, Y]$ be the $(d+2) \times n$ matrix where we store the augmented X matrix and Y:

$$\boldsymbol{Z}_{i,j} = \begin{cases} \boldsymbol{X}_{i,j} & \text{if } i < d+1 \\ Y_j & \text{if } i = d+1 \end{cases} \tag{10}$$

$\Gamma = \boldsymbol{Z}\boldsymbol{Z}^T$ contains a summary of the whole data set, sufficient to compute the linear model.

$$\begin{aligned} \Gamma &= \begin{bmatrix} n & \sum x_i^T & \sum y_i \\ \sum x_i & \sum x_i x_i^T & \sum x_i y_i \\ \sum y_i & \sum y_i x_i^T & \sum y_i^2 \end{bmatrix} \\ &= \begin{bmatrix} n & L^T & \sum y_i \\ L & Q & XY^T \\ \sum y_i & YX^T & YY^T \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{X}Y^T \\ Y\boldsymbol{X}^T & YY^T \end{bmatrix} \end{aligned} \tag{11}$$

Notice that $\boldsymbol{X}Y^T, Y\boldsymbol{X}^T$ and YY^T can be calculated once, in one pass, at the beginning of the algorithm. Furthermore from equation 11 we can see that Γ is symmetric: $\Gamma = \Gamma^T$. This has important implications both on the issue of storage as well as the simplification of some of our computations. The dimensionality of the summarization matrices is presented in Table 2.

We can derive the posterior of $\pi(\gamma|X,Y)$ as an expression dependent on n, L, $\boldsymbol{Q}$ and $Y\boldsymbol{X}^T$:

$$\pi(\gamma|X,Y) \propto \frac{1}{(c+1)^{\frac{k+1}{2}}} \left(YY^T - \frac{c}{c+1}(Y\boldsymbol{X}_\gamma^T)(\boldsymbol{Q}_\gamma)^{-1}(\boldsymbol{X}_\gamma Y^T) - \frac{1}{c+1}\tilde{\beta}_\gamma^T \boldsymbol{Q}_\gamma \tilde{\beta}_\gamma \right)^{-n/2} \tag{12}$$

3.5 The SciDB Array Database System

SciDB is an array DBMS that manipulates multidimensional arrays of unlimited size. Array operations are performed natively, removing RAM limitations. SciDB differs from a traditional DBMS in that data is partitioned and stored in *chunks*. A chunk is a large multidimensional block on secondary storage. We can visualize a 2-dimensional array stored on disk as a grid of rectangular chunks, where each chunk represents a sub-array of contiguous cells. Like most parallel DBMSs, SciDB is based on a SHARED-NOTHING architecture (Stonebraker et al, 2011) and (Stonebraker et al, 2013). In the SciDB shared-nothing architecture each node (processing computer) hosts one or more SciDB instances (threads), where each instance is responsible for local storage and processing of a partition of the input array.

SciDB offers several models of matrix computation. The most naive approach is to export an array to a linear algebra package such as LAPACK (Anderson et al, 1990), but in doing this we lose the advantage of storing data in a DBMS. Moreover the exporting time of large amounts of data could be prohibitive. A second way of operating in SciDB is to use the query languages provided by the DBMS: The array query language (AQL), a high level declarative language similar to SQL and the array functional language (AFL), a functional

Algorithm 1 The Variable Selection Algorithm

```
Require: $X, Y$
Ensure: $\gamma = \gamma^{[1]}, \gamma^{[2]}, \ldots, \gamma^{[N]}$ the Markov Chain
  /* Parameters (defined in the text) */
  $B, N, c$
  /* Pre-selection */
  Call SciDB function: correlation()
  Call SciDB function: preselect()
  /* Summarization */
  Call SciDB function: Gamma()
  /* Gibbs Sampler */
  for $I = 1$ to $N$ do
    for $j = 1$ to $d$ do
      Choose a variable $x_j$ at random without replacement
      Change the state of $\gamma_j^{[I]}$
      Prob. current model ← Calculate Prob $\left(\gamma^{[I]}\right)$
      $\gamma_j^{[I]}$ ← Sample from $\{0,1\}$, based on Probabilities of last and current models
      if $I > B$ then
        /* After the burn in period */
        Append $\gamma^{[I]}$ to the Markov chain
      end if
    end for
  end for
```

language for working with SciDB arrays where AFL operators are used to compose queries or statements. Lastly, we can define a new operator programmed in C++. This operator can be called directly or using SciDB's R extension, SciDB-R (Stonebraker et al, 2013).

We make use of these properties to write R programs that will be recognizable to any machine learning specialist, but that leverage the power granted by the SciDB database system (Section 5.2).

4 An Efficient Gibbs Sampler Algorithm

In this section we present a general treatment of our algorithm for variable selection and linear regression in high dimensional data.

4.1 Global Algorithm

Our algorithm consists of three parts: Variable Pre-selection (Section 4.2), Sufficient Statistics Calculation (Section 4.3) and Iterative Computation.

4.2 Variable Pre-selection

In order to reduce the computational effort, we perform an initial screening of the variables. We considered two approaches to reduce the dimensionality of the data set: The correlation ranking method (Bondell and Reich, 2012; Guyon and Elisseeff, 2003; Fan and Lv, 2008) works by calculating the marginal correlation ρ_i, of each of the independent variables with

Table 3 Model parameters for the Gibbs Sampler

N	Number of iterations
B	Number of burn in iterations
c	Parameter of Zellner's prior
$\tilde{\beta}$	Prior belief of the regression coefficients

the dependent variable.

$$\rho_k = \frac{\sum_{i=1}^{n}\left(x_k^i - \bar{x}\right)\left(y^i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_k^i - \bar{x}\right)^2 \sum_{i=1}^{n}\left(y^i - \bar{y}\right)^2}} = \frac{\sum_{i=1}^{n} x_k^i y^i - n\bar{x}\bar{y}}{(n-1)\,\sigma_{x_k}\sigma_y} \tag{13}$$

Once all the correlations are calculated, they are ordered (ranked) in descending order and the top d ranked variables are kept. The data set thus produced has its dimensionality reduced to d, simplifying the subsequent calculations.

A second, popular approach is to calculate all the univariate linear models for each of the independent variables with the dependent variable. The coefficients of regression β_k are then used for the ranking of the variables.

$$\beta_k = \frac{\sum_{i=1}^{n} x_k^i y^i}{\sum_{i=1}^{n}\left(x_k^i\right)^2} \tag{14}$$

However, it is evident that if the variables are normalized ($\bar{x_k} = 0$ and $\sigma_{x_k} = 1$), the ranking will be the same for both methods.

Since the data sets we used were not normalized we used correlation ranking as our pre-selection method. Moreover, the correlation is directly interpretable, and the difference between low correlation and high correlation variables is easy to understand.

4.3 Sufficient Statistics Computation

As mentioned in 3.4, we will use sufficient statistics to minimize the number of times the data set is read. By computing the matrix Γ after pre-selection and before the beginning of the iterations, we avoid reading the whole data set, since we only need to work with the subset of variables pre-selected. Furthermore Γ does not need to be recalculated during the iterative process, so the data set is never directly read again. This is an important requirement to improve the efficiency of the algorithm. Reducing the number of times the data is read from secondary storage reduces the time that is spent during I/O operations.

4.4 Iterative Computation

Once the pre-selection has been performed and the sufficient statistics computed, we can start the Gibbs Sampler. This step requires that we fix a number of parameters (see table 3). The numerical values of the parameters N and B are fixed experimentally.

The number of variables selected in each iteration, k, can be calculated by performing the dot product $k = \gamma^T \cdot \gamma$. If n is the number of data points in the data set when we run the Gibbs sampler in a sub-model, at every iteration the algorithm requires that $k < n$, to avoid the possibility of the matrix XX^T being singular. This can be explained by realizing that we are dealing with an under-determined system of linear equations. In such a system, if the number of variables is equal or larger than the number of equations, the linear regression will give unrealistic results.

In each iteration of the Gibbs sampler, the inclusion of every γ_j is calculated based on the probabilities $\gamma_j = 0$ and$\gamma_j = 1$. The computation of such probabilities is the most intensive computation of the process. The probability computation requires several projections of Γ based on the current model. Note that the most expensive operation in the calculation of the posterior probabilities is the matrix inversion $\left(XX^T\right)^{-1}$.

4.5 Time complexity

In order to calculate the time complexity of the function, we will break it down to its component subroutines. The pre-processing of the data set involves the variable pre-selection and the one pass calculation of Γ. Let us define p as the number of dimensions of the data set before pre-selection, as opposed to d, the dimensionality of the data set after pre-selection. The pre-selection requires the calculation of the all the correlations and sorting them. Therefore, the time complexity of pre-selection is $O(np + p\log p)$, where p is the dimensionality of the data set before pre-selection. The computation of the summarization step requires a data set scanning as well as $O(d^2)$ operations for each data point; thus, the total time complexity of the summarization step is $O(nd^2)$. Since the pre-processing steps (pre-selection and summarization) are performed only once, the time spent in these two steps is negligible compared to the time spent in the main steps of the process.

For every variable we need the probabilities $\gamma_j = 0$ and $\gamma_j = 1$. The computation of such probabilities is the most intensive computation of the process. The probability computation requires several projections of Γ based on the current γ: $Y\boldsymbol{X}_\gamma^T$, $\boldsymbol{Q}_\gamma$, $\boldsymbol{X}_\gamma Y^T$. Afterwards, the probabilities are computed performing the inversion of $\boldsymbol{Q}_\gamma$ and several matrix multiplications. The matrix inversion, the bottleneck of the computation, is performed in $O((k+2)^3)$ time. Since this computation must be performed for every variable, the time complexity per iteration is $O(d(k+2)^3)$, and the total time complexity of the Gibbs sampler is $O(Nd(k+2)^3)$.

5 Implementation of the Algorithm in SciDB/R

Recalling section 4, our algorithm consists of three main parts: Variable Pre-selection, Sufficient Statistics calculation, and Iterative computation. The first two steps are performed entirely in the database, using a mix of custom operators that can be called using AFL or R. The summarization matrix calculated is stored in the database. Once the summarization step is completed, an R program performs the variable selection, calling the matrix stored in SciDB to compute the probabilities. We programmed the Gibbs sampler as an R function that receives the summarization matrix as a parameter. After each iteration ends, the model vector γ_i is stored in the database for later analysis (See Algorithm 1).

5.1 Sufficient Statistics in SciDB

In equation 12 we showed that the posterior probabilities can be expressed in terms of n, L and Q, so calculating these quantities once will result in a significant improvement in the efficiency of the algorithm.

While n and L can be efficiently calculated in SciDB using simple AQL queries, the computation of Q with built in operators does not fare as well (Ordonez et al, 2014b). In (Ordonez et al, 2014b) we propose an algorithm to compute Γ, a matrix that contains all the sufficient statistics we require to characterize the data set. We propose a powerful matrix summarization operator which performs the calculation of Γ using

$$\Gamma = \boldsymbol{Z} \cdot \boldsymbol{Z}^T = \sum_{i=1}^{n} z_i \cdot z_i^T \tag{15}$$

taking advantage of SciDB's parallel architecture. The matrix thus calculated is then stored in the database, ready to be queried during the iterative part of the algorithm.

Internally R calls LAPACK for the computation of the inverse, but it is still the bottleneck in the algorithm.

5.2 Integrating Algorithm with R

The first two parts of the algorithm are performed directly in SciDB using operators developed to this end. The pre-selection step reads the whole data set from a SciDB array, and generates an alternate data set by discarding the lowest ranked variables (Section 3.2). This new dataset is stored as a SciDB array. The `correlation()` operator calculates the univariate correlations and ranks the variables. After the correlations are calculated, the `preselect()` operator generates a new SciDB array by discarding all variables that have a low correlation ranking. Finally the operator `Gamma` uses the preselected data set to generate the sufficient statistics matrix described in 3.4 and stores it back in SciDB. The Gamma operator may be called from R or from SciDB. SciDB queries are executed by the iquery R-function:

```
iquery(store(GammaOperator(mydata), SummarizationMatrix))
```

Then, the data is retrieved as a regular R matrix, as explained above

The Gibbs sampler is entirely programmed in R. First we use SciDB-R to retrieve the Γ matrix stored in the database as a regular array. The Γ matrix is small enough that can be copied in its entirety into RAM. Since all the information required by the algorithm is present in the Γ matrix, we don't need to interact again with the data set. The Gamma matrix also accelerates the processing time, because most of the terms in the probability formula (Equation 8) are already calculated in Gamma. The output array, an array with $N - B$ rows and d columns (rows from the burn-in are discarded) is stored in RAM. This output array, $\{\gamma_{B+1}, \gamma_{B+2}, \cdots, \gamma_N\}$, is analyzed in R to determine the output models. We calculate variable posterior probabilities, models accuracy and model size for each row of the output array. After all the iterations have finished, the output array is also stored in SciDB for further analysis, which can be done either on SciDB or in R.

6 Experimental Validation

In previous sections we described the basic Gibbs sampler, as well as the improvements we devised in order to efficiently implement this algorithm. Here we present the results of our experiments as well as the evaluation of their accuracy and time complexity. We used our algorithm to search for parsimonious models of the survival time for patients suffering from Glioblastoma Multiforme (GBM). The data sets are presented in Table 4. The variable of interest Y is the patient's survival time. The data sets $X1$ and $X2$ contain the gene expression and the microRNA expression of the same set of 248 subjects. No right-censored data points are being considered (patients still alive were not considered in the experiment). We apply our algorithm to data set $X1$ (gene expression), $X2$ (microRNA expression) and $X1 \cup X2$ (joint analysis of gene and microRNA expression). Gene/mRNA and microRNA expression provide differently molecular information about the progression of the disease and the main scientific questions are not only to model the effects each platform independently but joint effects as well, on patients' prognosis.

Table 4 Data sets used in this work. For all data sets $n = 248$

Data set	d	Description
$X1$	11972	Gene Expression
$X2$	534	microRNA Expression
$X1 \cup X2$	12506	Combined Gene/microRNA Expression

Table 5 DBMS and server characteristics.

DBMS	SciDB 14.3
Operating System	Linux Ubuntu 14.04
Processor	Intel Xeon X3210, Quad-core CPU, 2.153 GHz
RAM	4 GB
Hard Disk	650 Gb; 7200 RPM

We run several experiments for this high dimensional data set ($p \approx 12000$). In Table 5 we show the system used for these experiments. We pre-select up to 2000 variables for co-variate sets $X1$ and $X1 \cup X2$. We analyze the accuracy of the results by running experiments with different parameters and measuring the average and maximum values of the regression coefficient in each experiment.

6.1 Experimental set-up and parameters used

Since $d \gg n$, the problem is under-determined and shows a proclivity to over-fitting. Therefore, if the parameter c is not large enough, the Gibbs sampler has a tendency to choose as many variables as there are data points. We studied the effect of this parameter on the number of variables chosen as can be seen in Figure 1.

The results are presented in the form of frequencies of the most often chosen variables by the algorithm. It is important to note that while the frequency of the variables is very

Table 6 Most selected variables for $c = 1 \times 10^6$, $d = 2000$, $N = 30000$, $B = 5000$. Data Set $X1 \cup X2$

Variable	Frequency	Name
500	3575	2581-at
503	3022	25833-at
586	3385	2591-at
1036	3445	54952-at
1046	3962	5030-at
1134	4837	55711-at
1197	2708	56978-at
1255	3949	5891-at
1303	7255	6251-at
1327	2759	63894-at
1372	3730	64901-at
1405	7685	6716-at
1571	3961	79647-at
1775	3469	8811-at
1856	4020	9337-at
1869	2715	9425-at
1924	5645	9812-at
1983	4461	hsa-miR-222
1995	3518	hsa-miR-520a

Table 7 Most selected variables for $c = 2 \times 10^6$, $d = 2000$, $N = 30000$, $B = 5000$. Only mRNA was considered in this experiment (Data Set $X1$)

Variable	Frequency	Name
500	2935	2581-at
586	4001	2591-at
1046	3893	5030-at
1134	4335	55711-at
1162	3178	$---$
1197	2571	56978-at
1255	2955	5891-at
1303	6664	6251-at
1372	4136	64901-at
1405	6920	6716-at
1571	3279	79647-at
1856	3492	9337-at
1924	4915	9812-at

stable from experiment to experiment, k, the final size of the output depends on the input parameters (See Figures 2, 3, 4, and 5).

6.2 Interpretation of the results

We ranked the genes based on their marginal posterior probabilities. The most selected genes and miRNA signatures for $X1 \cup X2$ are presented in Table 6. Table 7 shows the most selected genes for $X1$ using a different value of c. Among these markers hsa-mir-222 has been identified before by (Roth et al, 2011) and (Srinivasan et al, 2011) as part of the GBM prediction signature. This would indicate that our method is in agreement with previously reported experimental results. While our experiments find some top markers that have been previ-

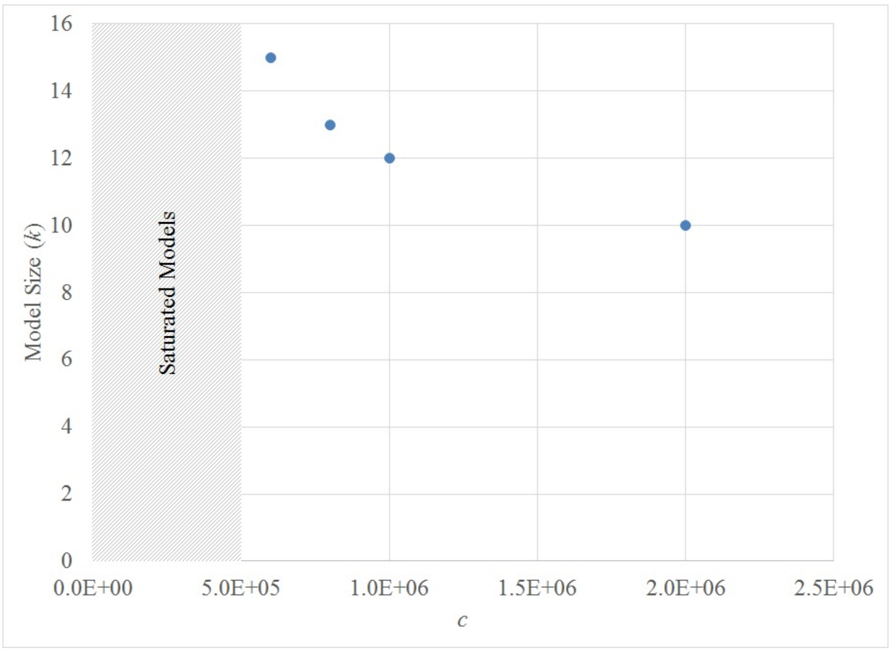

Fig. 1 Most frequent model size as a function of the c parameter. For $c < 5 \times 10^5$ the algorithm saturates (model size is equal to n). $d = 2000$, $N = 30000$, $B = 5000$, data set $X1 \cup X2$

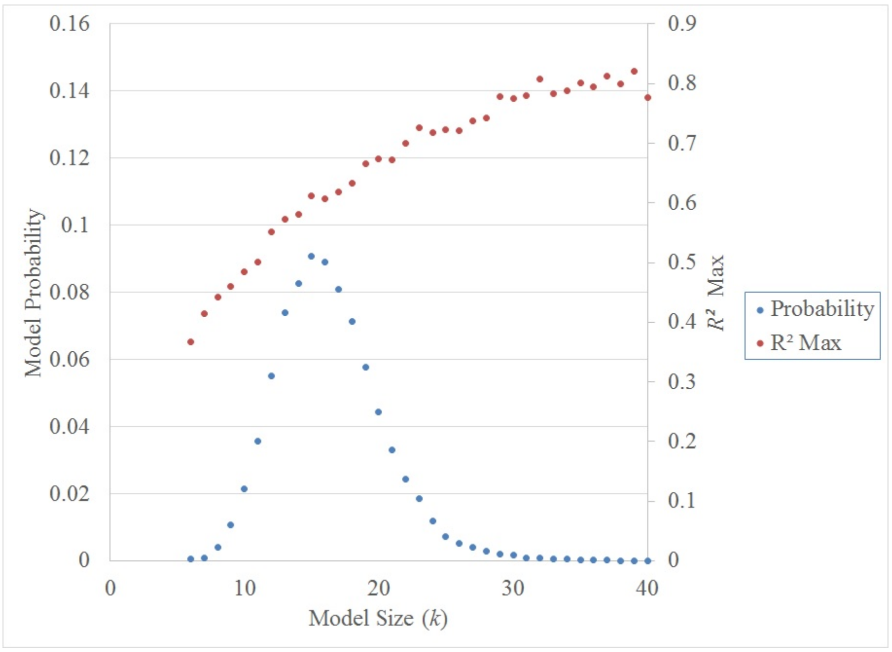

Fig. 2 Model probability and maximum R^2 as a function of the model size, k. $d = 2000$, $c = 0.6 \times 10^6$, $N = 30000$, $B = 5000$, data set $X1 \cup X2$

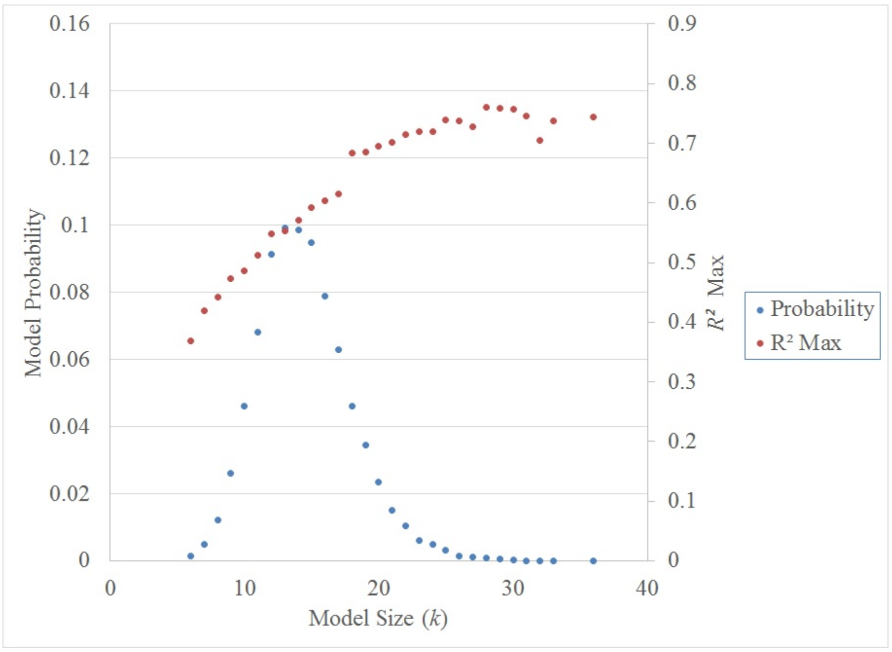

Fig. 3 Model probability and maximum R^2 as a function of the model size, k. $d = 2000$, $c = 0.8 \times 10^6$, $N = 30000$, $B = 5000$, data set $X1 \cup X2$

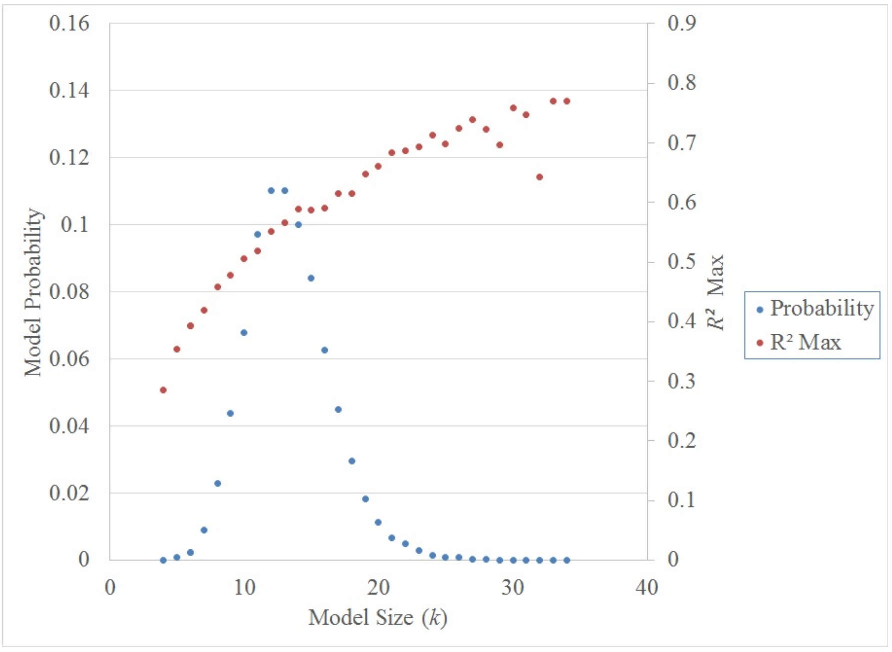

Fig. 4 Model probability and maximum R^2 as a function of the model size, k. $d = 2000$, $c = 1 \times 10^6$, $N = 30000$, $B = 5000$, data set $X1 \cup X2$

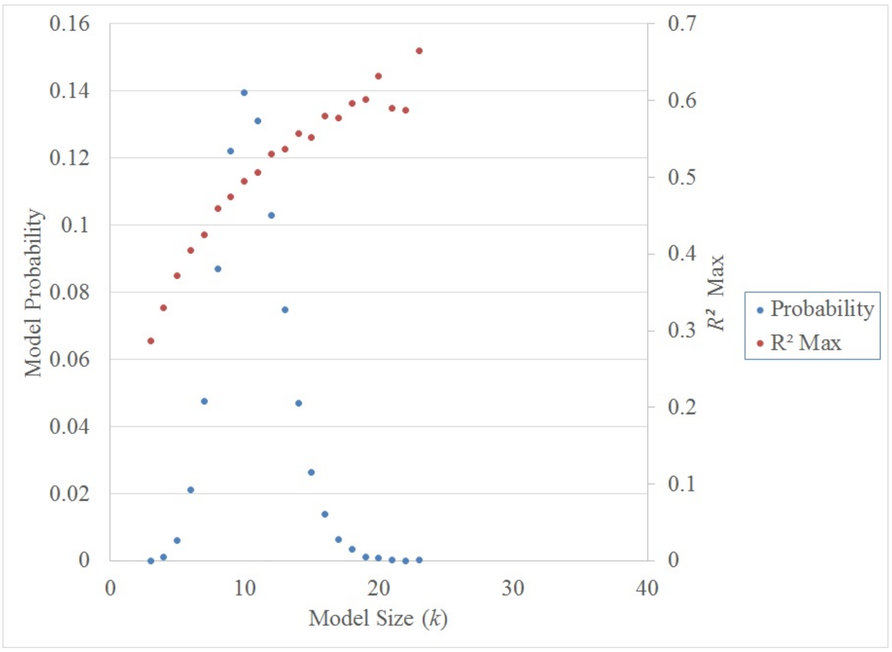

Fig. 5 Model probability and maximum R^2 as a function of the model size, k. $d = 2000$, $c = 2 \times 10^6$, $N = 30000$, $B = 5000$, data set $X1 \cup X2$

ously implicated in the literature we also find a list of new markers that could merit further functional validation for development of future therapeutic strategies.

Figures 2, 3, 4, and 5 show the model size frequency and the corresponding maximum R^2 for different values of the parameter c. We notice that while R^2 increases for larger models (as can be expected) the frequency of selected models is maximized around smaller model sizes. The maximum of the model size frequency grows with c. We must clarify that model size frequency is not the same as model frequency. Ideally, a Markov chain nominates the model that is repeated most times as the best model. However, since the number of variables is so large compared to the final model size, the likelihood of any model repeating is very low. In other words, we do not find that identical models are selected a significant number of times. Nevertheless, models that are almost identical, models that differ in one or two variables, are found frequently. These variable combinations were used to calculate the most selected variables shown in Tables 6 and 7.

6.3 Performance Analysis

In Table 8 we show the time that our system takes to complete 30000 iterations for different number of variables. As expected from Section 4.5, the behavior is linear with the number of dimensions. We changed the parameter c in order to maintain k roughly constant.

Table 9 presents a comparison between a variable selection package in R and our algorithm, showing the execution time for 1000 iterations. Depending on the number of variables in the experiment we can see a 30 to 100 fold improvement using our algorithm. For instance

Table 8 Execution time in seconds on SciDB varying d. $N = 30000$, $B = 5000$, data set $X1 \cup X2$

d	c	Time
500	10K	4310
1000	100K	7126
2000	1000K	13200

Table 9 Time in seconds per 1000 iterations, comparing existing R package with our optimized algorithm combining R and SciDB. Data set $X1 \cup X2$

d	R Package	Optimized Algorithm
400	5468	194
534	8378	196
1000	22898	272
2000	*	250

* Stopped after 2 hours

in the case of $d = 534$ (data set $X2$), R performs 1000 iterations in 8378 seconds, while it takes just 196 seconds using our system. We attribute the substantial speed increase to the use a pre-computed GAMMA matrix that resides in memory

7 Conclusions and future work

We present the first Markov Chain Monte Carlo algorithm integrating matrix computations in R and in the SciDB database. Since we use the database both for a data repository and as the engine for pre-selection and summarization, we are able to handle massively large amounts of data, removing R's RAM and speed limitations, as we only need to deal with a reduced dataset and most important quantities are pre-calculated. Therefore we are able to work with data sets that are vastly larger than the system's main memory. Our method is able to reuse existing R code, with minimal additions to communicate with SciDB. As a consecuence, the presented algorithm is easier to adapt to fit other kinds of data and possibly change priors, as long as they can be expressed in terms of the sufficient statistics summarization matrix. We pre-process the data set to work with a lower number of variables without discarding important models, since the variables discarded have very low correlation with the dependent variable. This new method removes the need for several complicated optimizations that were necessary in our previous work, making the code cleaner, more elegant and easier to understand, modify and parallelize. The most frequent genetic markers found using our algorithm are in good agreement with the medical literature, and we find others that might merit some further attention. Our method has a linear time complexity with the number of dimensions and the algorithm is 30 to 100 times faster than using only R. Furthermore, since the results and output arrays are stored in the database, further analysis can be performed easily using R or SciDB tools.

Future avenues for research includes the analysis of the models selected to find clusters of genes that are most commonly selected together. We will explore different kernels and priors to see if we find results in less iterations. We will work on data sets with more than 50000 variables, maintaining the data set in RAM but computing the summarization matrix on demand. We also will include second order effects, considering a data base with second order interactions between the variables. We want to study the trade offs between using a correlation threshold and discarding possibly valuable variable combinations. This will allow us to determine if there are any variables that, even if they might be poorly correlated with the dependent variable individually, they might merge to produce a strongly correlated combination. Regarding the R part of our algorithm we are working on incrementally computing the inverse, to reduce the need of costly LAPACK calls. Finally we plan to pre-compute the probabilities in parallel using a look-ahead optimization.

Acknowledgements The authors would like to thank Veera Baladandayuthapani, for his help understanding Bayesian statistics. We would also like to thank Michael Stonebraker, for his help on understanding SciDB array storage and array processing.

References

Anderson E, Bai Z, Dongarra J, Greenbaum A, McKenney A, Du Croz J, Hammerling S, Demmel J, Bischof C, Sorensen D (1990) LAPACK: A portable linear algebra library for high-performance computers. In: Proceedings of the 1990 ACM/IEEE conference on Supercomputing, IEEE Computer Society Press, pp 2–11

Bondell HD, Reich BJ (2012) Consistent high-dimensional Bayesian variable selection via penalized credible regions. Journal of the American Statistical Association 107(500):1610–1624

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning 3(1):1–122

Brennecke J, Stark A, Russell RB, Cohen SM (2005) Principles of microRNA–target recognition. PLoS biology 3(3):e85

Cabrera W, Ordonez C, Matusevich DS, Baladandayuthapani V (2013) Bayesian variable selection for linear regression in high dimensional microarray data. In: Proceedings of the 7th International Workshop on Data and Text Mining in Biomedical Informatics, ACM, New York, NY, USA, DTMBIO '13, pp 17–18

Caruana R, Freitag D (1994) Greedy attribute selection. In: ICML, Citeseer, pp 28–36

Caruana R, Niculescu-Mizil A, Crew G, Ksikes A (2004) Ensemble selection from libraries of models. In: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, p 18

Davies V, Reeve R, Harvey W, Maree F, Husmeier D (2014) Sparse Bayesian variable selection for the identification of antigenic variability in the foot-and-mouth disease virus. In: Journal of Machine Learning Research: Workshop and Conference Proceedings, Journal of Machine Learning Research, vol 33, pp 149–158

Debouck C, Goodfellow PN (1999) DNA microarrays in drug discovery and development. Nature genetics 21:48–50

Derbinsky N, Bento J, Elser V, Yedidia JS (2013) An improved three-weight message-passing algorithm. arXiv preprint arXiv:13051961

Duggan DJ, Bittner M, Chen Y, Meltzer P, Trent JM (1999) Expression profiling using cDNA microarrays. Nature genetics 21:10–14

Efron B, Hastie T, Johnstone I, Tibshirani R, et al (2004) Least angle regression. The Annals of statistics 32(2):407–499

Faith J, Mintram R, Angelova M (2006) Targeted projection pursuit for visualizing gene expression data classifications. Bioinformatics 22(21):2667–2673

Fan J, Lv J (2008) Sure independence screening for ultrahigh dimensional feature space. Journal of the Royal Statistical Society 70:849–911

Fukushima M (1992) Application of the alternating direction method of multipliers to separable convex programming problems. Computational Optimization and Applications 1(1):93–111

Garey MR, Johnson DS (2002) Computers and intractability, vol 29. wh freeman

Ge T, Grabiner D, Zdonik S (2011) Monte Carlo query processing of uncertain multidimensional array data. In: Data Engineering (ICDE), 2011 IEEE 27th International Conference on, IEEE, pp 936–947

George E (2000) The variable selection problem. Journal of the American Statistical Association 95(452):1304–1308

George EI, McCulloch RE (1993) Variable selection via Gibbs sampling. Journal of the American Statistical Association 88(423):881–889

Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. The Journal of Machine Learning Research 3:1157–1182

Guyon I, Saffari A, Dror G, Cawley G (2010) Model selection: Beyond the Bayesian/Frequentist divide. The Journal of Machine Learning Research 11:61–87

Hastie T, Tibshirani R, Friedman J (2001) The Elements of Statistical Learning, 1st edn. Springer, New York

Hellerstein J, Re C, Schoppmann F, Wang D, et al (2012) The MADlib analytics library or MAD skills, the SQL. Proc of VLDB 5(12):1700–1711

Hocking RR (1976) A biometrics invited paper. the analysis and selection of variables in linear regression. Biometrics 32(1):1–49

Ihaka R, Gentleman R (1996) R: a language for data analysis and graphics. Journal of computational and graphical statistics 5(3):299–314

Jones GL, Hobert JP (2004) Sufficient burn-in for gibbs samplers for a hierarchical random effects model. Annals of statistics pp 784–817

Kolar M, Lafferty J, Wasserman L (2011) Union support recovery in multi-task learning. The Journal of Machine Learning Research 12:2415–2435

Liang F (2008) Mixtures of g priors for Bayesian variable selection. Journal of the American Statistical Association 103(481)

Mantel N (1970) Why stepdown procedures in variable selection. Technometrics 12(3):621–625

Marin JM, Robert CP (2007) Bayesian Core: A Practical Approach to Computational Bayesian Statistics. Springer Publishing Company, New York

Matusevich DS, Ordonez C (2014) A clustering algorithm merging MCMC and EM methods using SQL queries. Journal of Machine Learning Research (JMLR): Workshop and Conference Proceedings (BigMine 2014) 36:61–76

Meinshausen N, Bühlmann P (2006) High-dimensional graphs and variable selection with the Lasso. The Annals of Statistics pp 1436–1462

Mitchell TJ, Beauchamp JJ (1988) Bayesian variable selection in linear regression. Journal of the American Statistical Association 83(404):1023–1032

Moore JH, Parker JS, Hahn LW (2001) Symbolic discriminant analysis for mining gene expression patterns. In: Machine Learning: ECML 2001, Springer, pp 372–381

Natarajan BK (1995) Sparse approximate solutions to linear systems. SIAM journal on computing 24(2):227–234

Needell D, Tropp J, Vershynin R (2008) Greedy signal recovery review. In: Signals, Systems and Computers, 2008 42nd Asilomar Conference on, IEEE, pp 1048–1050

Ordonez C (2010) Statistical model computation with UDFs. IEEE Transactions on Knowledge and Data Engineering (TKDE) 22(12):1752–1765

Ordonez C, Garcia-Alvarado C, Baladandayuthapani V (2014a) Bayesian variable selection in linear regression in one pass for large datasets. ACM Transactions on Knowledge Discovery from Data (TKDD) 9(1):3

Ordonez C, Zhang Y, Cabrera W (2014b) The Gamma operator for big data summarization on an array DBMS. Journal of Machine Learning Research (JMLR): Workshop and Conference Proceedings (BigMine 2014) 36:61–96

Pitchaimalai S, Ordonez C, Garcia-Alvarado C (2010) Comparing SQL and MapReduce to compute Naive Bayes in a single table scan. In: Proc. ACM CloudDB, pp 9–16

Puntanen S, Styan GP (1989) The equality of the ordinary least squares estimator and the best linear unbiased estimator. The American Statistician 43(3):153–161

Rockova V, Lesaffre E, Luime J, Löwenberg B (2012) Hierarchical Bayesian formulations for selecting variables in regression models. Statistics in Medicine 31(11-12):1221–1237

Roth P, Wischhusen J, Happold C, Chandran PA, Hofer S, Eisele G, Weller M, Keller A (2011) A specific miRNA signature in the peripheral blood of glioblastoma patients. Journal of neurochemistry 118(3):449–457

Sauerbrei W, Royston P, Binder H (2007) Selection of important variables and determination of functional form for continuous predictors in multivariable model building. Statistics in medicine 26(30):5512–5528

Schirmer EC, Jose I (2014) Cancer biology and the nuclear envelope. Advances in Experimental Medicine and Biology

Srinivasan S, Patric IRP, Somasundaram K (2011) A ten-microRNA expression signature predicts survival in glioblastoma. PLoS One 6(3):e17,438

Stonebraker M, Brown P, Poliakov A, Raman S (2011) The architecture of SciDB. In: Scientific and Statistical Database Management, Springer, pp 1–16

Stonebraker M, Brown P, Zhang D, Becla J (2013) SciDB: A database management system for applications with complex analytics. Computing in Science & Engineering 15(3):54–62

Taft R, Vartak M, Satish NR, Sundaram N, Madden S, Stonebraker M (2014) GenBase: a complex analytics genomics benchmark. In: Proc. ACM SIGMOD Conference, pp 177–188

Tibshirani R (1996) Regression shrinkage and selection via the Lasso. Journal of the Royal Statistical Society Series B (Methodological) pp 267–288

Tropp JA (2004) Greed is good: Algorithmic results for sparse approximation. Information Theory, IEEE Transactions on 50(10):2231–2242

Tropp JA, Gilbert AC (2007) Signal recovery from random measurements via orthogonal matching pursuit. Information Theory, IEEE Transactions on 53(12):4655–4666

Wainwright MJ (2009) Sharp thresholds for high-dimensional and noisy sparsity recovery using-constrained quadratic programming (Lasso). IEEE Transactions on Information Theory 55(5):2183–2202

Wasserman L, Roeder K (2009) High dimensional variable selection. Annals of statistics 37(5A):2178

Xue L, Qu A (2012) Variable selection in high-dimensional varying-coefficient models with global optimality. The Journal of Machine Learning Research 13(1):1973–1998

Zellner A (1986) On assessing prior distributions and bayesian regression analysis with g-prior distributions. Bayesian inference and decision techniques: Essays in Honor of Bruno De Finetti 6:233–243

Zhang T (2011) Sparse recovery with orthogonal matching pursuit under RIP. Information Theory, IEEE Transactions on 57(9):6215–6221

Zhang Y, Zhang W, Yang J (2010) I/O-efficient statistical computing with RIOT. In: Proc. ICDE