On the Computation of Stochastic Search Variable Selection in Linear Regression with UDFs

Mario Navas
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

Carlos Ordonez
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

Veerabhadran Baladandayuthapani
University of Texas
MD Anderson Cancer Center
Houston, TX 77030, USA

***Abstract*—Computing Bayesian statistics with traditional techniques is extremely slow, specially when large data has to be exported from a relational DBMS. We propose algorithms for large scale processing of stochastic search variable selection (SSVS) for linear regression that can work entirely inside a DBMS. The traditional SSVS algorithm requires multiple scans of the input data in order to compute a regression model. Due to our optimizations, SSVS can be done in either one scan over the input table for large number of records with sufficient statistics, or one scan per iteration for high-dimensional data. We consider storage layouts which efficiently exploit DBMS parallel processing of aggregate functions. Experimental results demonstrate correctness, convergence and performance of our algorithms. Finally, the algorithms show good scalability for data with a very large number of records, or a very high number of dimensions.**

***Keywords*-Bayesian statistics;variable selection;UDF**

I. Introduction

The objective of data mining algorithms is to efficiently compute models for large amounts of data. In variable selection, the major concern is to find the attributes which better explain the output of interest. Since, finding the best set of explanatory variables implies an exhaustive search through all possible combination of variables, selecting a linear model is demanding and often untenable in large datasets with many variables. Stepwise regression is the conventional approach to select attributes; evaluating whether to include or exclude variables in the model one at a time. However, only a limited fraction of the model space is visited, and solutions could be suboptimal. On the other hand, stochastic search variable selection (SSVS) [4] evaluates models based on a stochastic probabilistic search over the possible model configurations best explaining the output variable. Based on the configuration yielding the highest posterior probabilities, the best model can be evaluated without conducting an exhaustive search over the model space.
Due the limitation of external tools to manage large amounts of data, we have seen the feasibility to implement SSVS within the DBMS context. Thus, the importance of combining SQL and UDFs (user-defined functions) to efficiently perform data mining of data stored in databases [6], [1]. Our contribution is to yield the DBMS with the capability to perform Bayesian variable selection of large datasets. SSVS is computed from summarization matrices after one scan over the input table. However, high-dimensional data demand further optimizations like caching submatrices with efficient memory management algorithms. Finally, we include a set of measurements to assess convergence of SSVS, and to select the elements in the resulting sequence that better estimate the posterior distribution.
This paper is organized as follows. Section 2 presents an overview of definitions. Section 3 discusses our algorithms for efficiently computing SSVS for large datasets. Experimental results, in Section 4, evaluate accuracy and performance of different methodologies when integrated to a DBMS. Section 5 discloses related work. Finally, Section 6 concludes the article.

This work was partially supported by NSF grants CCF 0937562 and IIS 0914861.

II. Preliminaries

A. Definitions

Assume the input data $X = \{x_1, \ldots, x_n\}$, with d dimensions and n data points x_i as column vectors. Consequently, the matrix X has size $d \times n$. The response Y, or variable of interest, is a matrix $1 \times n$. To indicate matrix transposition we use the superscript T. The a^{th} dimension is referred with the subscript X_a, while a superscript γ^I is used for the I^{th} iteration in stochastic search variable selection (γ is defined bellow). We will use the terminology 1_n to represent a column vector of n ones (as in [7]). Finally, $\mathcal{X}$ is used for the augmented version of X in linear regression.

B. Summarization Matrices

Data set summarization is fundamental is fundamental in data mining and statistics [5]. The set of summarization matrices which could substitute X in computations are [11]: n, L, and Q. The value n is a count of points in the input data set X. L is the linear sum of the d dimension in X, or $L = X \cdot 1_n = \sum_{i=1}^{n} x_i$, and forms a vector of d values. The matrix Q is the quadratic sum of the cross-products of points, or $Q = XX^T = \sum_{i=1}^{n} x_i \cdot x_i^T$, with size $d \times d$.

Considering the data set X stored inside the DBMS, summarization is computed with aggregate operations and only one scan over the input table. Moreover, aggregate operations are distributive; thus, they can be computed in parallel over different sections of the data, where the global aggregation is given by the addition of all partial results. The storage layouts of points in the input table could be by row or by column. In order to compute n, L, and Q with aggregate operations, points are stored by row. Also, the matrix operations XY^T, YY^T, and $Y1_n$ are computed as part of the summarization by adding the variable Y as an attribute of the input table $(X_1, \ldots, X_d, Y)$. Our focus is to exploit the summarization matrices for Bayesian computations.

C. Bayesian Variable Selection

Linear regression models the relationship between an output or response variable Y, and a set a set of explanatory variables or estimators X. The regression model $Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_d X_d + \epsilon$, is defined by $\beta = (\beta_0, \beta_1, \ldots, \beta_d)$, the regression coefficients and a Gaussian white noise error ϵ. Since increasing d (the number of estimators) not necessarily increases the quality of the model, the problem of variable selection is to find a set of explanatory variables that best predicts the output variable Y, among the 2^d possible subsets of X. In variable selection via Gibbs sampler [4], a model $\mathcal{M}_\gamma$ is represented by a $\gamma \in \Gamma = \{0,1\}^d$, where the variable X_a is part of the model if $\gamma_a = 1$. The estimation for the probability distribution $\pi(\gamma|Y,X)$ obtained by convolving the likelihood and the prior distribution is computed with the number of appearances of each model $\mathcal{M}_\gamma$ taken S iterations. Therefore, stochastic search variable selection (SSVS) generates a sequence γ^I, where $I = 1, \ldots, S$. Given that $\gamma^I_{(a)} = (\gamma^I_1, \ldots, \gamma^I_{a-1}, \gamma^{I-1}_{a+1}, \ldots, \gamma^{I-1}_d)$, the sample γ^I is obtained by evaluating the probabilities of $\gamma^I_a = 0$ and $\gamma^I_a = 1$ for each explanatory variable a, using the probability function $\pi(\gamma^I_a|Y, X, \gamma^I_{(a)})$. In contrast to stepwise variable selection which does an exhaustive search to find the model that better explains the output variable, SSVS finds this subset of variables based on posterior probabilities.

D. Posterior Probability Estimation

The general linear regression modeling of the output variable is $Y = \beta^T \mathcal{X} + \epsilon$, where $\beta \in \mathbb{R}^{d+1}$, and the augmented $\mathcal{X}$ is defined as in Equation 1. The Zellner's G-prior [7] is used to introduce prior information into the Bayesian regression modeling. Model selection depends on the G-prior function, which is conditional on the error variance, σ^2, a constant c and a hyperparameter $\tilde{\beta}$ defined as $\beta|\sigma^2, X \backsim \mathcal{N}(\tilde{\beta}, c\sigma^2(\mathcal{X}\mathcal{X}^T)^{-1})$. To complete the hierarchical formulations we define another prior on the error variance as $\sigma^2 \backsim \pi(\sigma^2|X) = \sigma^{-2}$, i.e. a non-informative prior distribution.

$$\mathcal{X} = \begin{bmatrix} 1_n^T \\ X \end{bmatrix} \tag{1}$$

For the model $\mathcal{M}_\gamma$, we have that $\beta_\gamma = (\beta_0, \{\beta_a : \gamma_a = 1\})$, the number of dimension in the model $d_\gamma = 1_n^T \cdot \gamma$, and the explanatory variables $X_\gamma = \{X_a : \gamma_a = 1\}$. Therefore, the prior distribution of β_γ for the model $\mathcal{M}_\gamma$ is shown in Equation 2, where $\tilde{\beta}_\gamma = (\mathcal{X}_\gamma \mathcal{X}_\gamma^T)^{-1} \mathcal{X}_\gamma \mathcal{X}^T \tilde{\beta}$. Consequently, the prior density function (see Equation 3) is used by the Gibbs sampler to select models in the true posterior distribution of $\pi(\gamma|Y,X)$.

$$\beta_\gamma|\gamma, \sigma^2 \backsim \mathcal{N}(\tilde{\beta}_\gamma, c\sigma^2(\mathcal{X}_\gamma \mathcal{X}_\gamma^T)^{-1}) \tag{2}$$

$$\begin{aligned} \pi(\gamma|Y,X) \quad \propto \quad & (c+1)^{-(d_\gamma+1)/2}[YY^T \\ & -\frac{c}{c+1} Y\mathcal{X}_\gamma^T(\mathcal{X}_\gamma \mathcal{X}_\gamma^T)^{-1}\mathcal{X}_\gamma Y^T \\ & -\frac{1}{c+1}\tilde{\beta}_\gamma^T \mathcal{X}_\gamma \mathcal{X}_\gamma^T \tilde{\beta}_\gamma]^{-n/2} \end{aligned} \tag{3}$$

Our choice is the prior function is due the feasibility to express $\pi(\gamma|Y,X)$ of the Zellner's informative G-prior in a closed form, and in terms of summarization matrices. We define algorithms for efficiently computing SSVS inside the DBMS, with one or few scans over the input table; thus, resulting in high gains in computational efficiency.

III. Algorithms

In this section we propose two algorithms to compute Bayesian variable selection. The first algorithm performs only one scan over the input table. The posterior distribution is computed from the summarization matrices n, L, and Q. Even though performing operations over matrices of size $d \times d$ is very convenient when $n >> d$, high-dimensional data demands further optimizations. A second algorithm uses memory resources to efficiently update X_γ, without a complete scan to the input table. Finally, the posterior distribution $\pi(\gamma|Y,X)$ is estimated based on the number of appearances of a model γ in the output sequence.

A. SSVS with Summarization Matrices

Data summarization can be used to compute the posterior probability distributions in variable selection with only one table scan over the input data. The algorithm is divided in two main phases: (1) summarize the input data with n, L, Q, and (2) compute the Bayesian regression model. In order to include XY^T, $Y1_n$, and YY^T in the summarization step, the explanatory variable Y is added as another attribute of the input table. Therefore, we compute n, L and Q for $(X_1, \ldots, X_d, Y)$. By expressing probability density functions in SSVS with summarization matrices, we overcome the main disadvantage, especially for large datasets, of having to scan the input dataset multiple times to compute a posterior probability of a model.
Since the multiplication $X_\gamma 1_n = \{L_a : \gamma_a = 1\}^T$, and the cross-product $X_\gamma X_\gamma^T$ has as elements $\{Q_{ab} : \gamma_a = 1 \wedge \gamma_b = 1\}$, the cross-product $\mathcal{X}_\gamma {\mathcal{X}_\gamma}^T$ of the augmented set of variables of the model $\mathcal{M}_\gamma$ can be derived from

Algorithm 1: Stochastic Search Variable Selection

```
Input: Table $data(X_1, \ldots, X_d, Y)$,
        iterations $S$, priors $c$, and $\tilde{\beta}$
1 : $(n, L, Q) \leftarrow$ Summarization($data$)
2 : Pick a random $\gamma^0$ .
3 : For $I \leftarrow 1$ to $S$
4 :     $\gamma^I \leftarrow \gamma^{I-1}$
5 :     For $a \leftarrow 1$ to $d$
6 :         $\gamma_a^I \leftarrow 0$
7 :         $p_0 \leftarrow PostProb(\gamma^I, c, \tilde{\beta}, n, L, Q, )$*
8 :         $\gamma_a^I \leftarrow 1$
9 :         $p_1 \leftarrow PostProb(\gamma^I, c, \tilde{\beta}, n, L, Q, )$*
10:         If Rand([0,1]) $< p_0/(p_0 + p_1)$
11:             $\gamma_a^I \leftarrow 0$
12:         else
13:             $\gamma_a^I \leftarrow 1$
Return: $\pi(\gamma|Y, X)$ from $\gamma^1, \ldots, \gamma^S$
```

*$\pi(\gamma_a^I|Y, X, \gamma_{(a)}^I)$ is evaluated, as shown in Equation 3, where X is substituded by n, L, and Q.

summarization matrices (see Equation 4). Likewise, the matrix multiplication $\mathcal{X}_\gamma \mathcal{X}^T$ (see Equation 5), in $\tilde{\beta}_\gamma$, is constructed knowing that $X_\gamma X^T = \{Q_{a,[1:d]} : \gamma_a = 1\}$. The remaining matrix multiplication in the probability distribution, $\mathcal{X}_\gamma Y^T$ or $(Y \mathcal{X}_\gamma^T)^T$, is derived from the fact that $X_\gamma Y^T = \{Q_{a,d+1} : \gamma_a = 1\}$ (see Equation 6). Finally, the dot product of the explanatory variable $YY^T = Q_{d+1,d+1}$.

$$\mathcal{X}_\gamma {\mathcal{X}_\gamma}^T = \begin{bmatrix} 1_n^T 1_n & 1_n^T X_\gamma^T \\ X_\gamma 1_n & X_\gamma X_\gamma^T \end{bmatrix} = \begin{bmatrix} n & {L_b}^T \\ L_a & Q_{ab} \end{bmatrix} \tag{4}$$

$$\mathcal{X}_\gamma \mathcal{X}^T = \begin{bmatrix} 1_n^T 1_n & 1_n^T X^T \\ X_\gamma 1_n & X_\gamma X^T \end{bmatrix} = \begin{bmatrix} n & L_{[1:d]}^T \\ L_a & Q_{a,[1:d]} \end{bmatrix} \tag{5}$$

$$\mathcal{X}_\gamma Y^T = \begin{bmatrix} 1_n^T Y^T \\ X_\gamma Y^T \end{bmatrix} = \begin{bmatrix} L_{d+1} \\ Q_{a,d+1} \end{bmatrix} \tag{6}$$

In our algorithm to compute SSVS, results are exact (see Algorithm 1). Only one scan to the input table is required, after that, operations are done using the summarization matrices n, L, and Q; without accessing the input table again. SSVS is computed efficiently for very large datasets, because the number of records only affects performance of the summarization step.

B. SSVS for High-dimensional Data

The posterior distribution $\pi(\gamma|Y, X)$ is estimated, for high-dimensional data, taking advantage of memory resources. The layout of the input table is to store points by column. One dimension is accessed at a time, X_γ is changed to include or exclude such dimension, and to compute the selection probability for the new model. Therefore, only one table scan is required to evaluate the d models in an iteration of SSVS. As an initialization of SSVS, memory space is allocated for caching a limited number of explanatory variables. Variables are evaluated one at a time. If the variable is picked by SSVS then its value is cached, otherwise it is removed from memory. Converge of SSVS depends on the initial pick for γ^0; if the memory space for caching is not enough to store X_γ, we assume the sequence is not mixing properly, and the algorithms is restarted with a different γ^0. The efficient execution of SSVS depends on appropriately estimate the memory space for caching the variables X_γ of the models $\mathcal{M}_\gamma$ in the true posterior distribution.

C. SQL and UDF Optimizations

We consider two storage layouts for the input matrix X inside the DBMS. The first layout is to store points by row, where columns are used to identify dimensions. The second layout is to store points by columns, where each row contains all the values for a dimension. Finally, we have that optimizations in the DBMS are to reduce the number of scans to the input table, in order to have good scalability for large datasets.

Points in the input data are stored by row, when $n >> d$, to express the summarization matrices n, L, and Q in terms of the `SUM` aggregation. A single SQL statement is used to calculate all values of the summary matrices. A property of Q is to be a symmetric matrix, so it is enough to include the $d(d+1)/2$ upper or lower triangular elements. When including all summarization values, we have the following SQL statement:

```
SELECT SUM(1.0) /*n*/
    ,SUM($X_1$),SUM($X_2$)...,SUM($X_d$) /*L or $X1_n$*/
    ,SUM($X_1 * X_1$) /*Q or $XX^T$*/
    ,SUM($X_2 * X_1$),SUM($X_2 * X_2$)
    ⋮
    ,SUM($X_d * X_1$),SUM($X_d * X_2$),...,SUM($X_d * X_d$),
/*Variable of interest*/
    ,SUM($Y$) /*$Y1_n$*/
    ,SUM($Y * Y$) /*$YY^T$*/
    ,SUM($X_1 * Y$),SUM($X_2 * Y$),...,SUM($X_d * Y$) /*$XY^T$*/
FROM X;
```

The resulting table has a single row and a column for each value in n, L, and Q. Since a limitation exists for the number of columns in a table depending on the DBMS provider, there is also a limit for the number of dimensions that can be summarized with a single scan. Aggregate UDFs can compute summarization by packing dimension with a user-defined type [12]. However, parsing operations and serialization of the user-define type is time costly, and affects overall performance. Previous limitations are avoided by implementing multithreading inside a TVF (table-valued function) to distribute the workload, while keeping the hard drive access sequential for performance. Also, the TVF implementation follows the common API of DBMSs to access query result sets. To obtain sequential reading, one thread is uniquely in charge of retrieving records from the input table, caching blocks of records in main memory, and calling a monitor to dispatch the job to other thread that actually performs the calculations or working thread. We define techniques to control the number of threads executing simultaneously, and the amount of memory used by the aggregation process. The summarization matrices are used to compute posterior probabilities of models, and the TVF returns a table storing the generated sequence $\gamma_{0,\ldots,S}$. The storage layout by column is used to compute SSVS for data where $d >> n$. Therefore, points in X are stored as columns in the table X. Data access and aggregations are done dealing with vectors storing complete dimensions. Due to its size, X_γ is used to compute posterior probabilities of models. Each iteration, dimensions are evaluated one at a time by including/excluding it from X_γ. Since database programmability APIs provide access to tables by row, each dimension vector is used to modify X_γ, and compute the posterior probability of the new model. Like in storage by row, the output sequence is returned as a table by the TVF. An alternative implementation are stored procedures (SPs) which can create the output tables directly in the DBMS.

In order to improve execution, further optimizations use memory resources to avoid unnecessary model evaluations. A hash table is used inside the TVF or the SP to have quick access to values for the posterior probability $\pi(\gamma_a|Y, X, \gamma_{(a)})$ that have been calculated in previous iterations of SSVS. Since, the space requirements for the hash data structure could surpass memory limitations with a bad choice for the initial γ^0, the hash table is set with a memory space limitation. Whenever the space requirement goes above such boundary, the hash table can be restarted, or it can be pruned. If the hash data structure is cleaned, SSVS could also be restarted with a new γ^0 to explore a different region of parameter space.

D. Time and Space Complexity

Here we analyze complexity of the alternatives to perform SSVS. For each iteration I of SSVS, d estimates for the probability $\pi(\gamma_a^I|Y, X, \gamma_{(a)}^I)$ are computed. Without optimizations, each posterior probability requires a complete scan to the input dataset for the most representative operations: $\mathcal{X}_\gamma \mathcal{X}_\gamma^T$ or $O(nd_\gamma^2)$, and an inverse computation or $O(d_\gamma^3)$. Since d_γ varies for the d probabilities to compute each iteration, we have that the overall complexity of SSVS is $O(nd^4)$. Nevertheless, with the use of summarization matrices to avoid several scans to the input table, probability estimations are done over matrices of size $d \times d$ with complexity $O(d^3)$. Since, the complexity of computing the summarization matrices n, L, and Q is $O(nd^2)$, the overall complexity of SSVS with summarization is $O(nd^2 + d^4)$. Finally, we have that the time complexity analysis is reflected in the experimental results.

IV. Experimental Results

For this paper, we conducted our experiments on a commercial DBMS installed on a server with an Intel Core 2 Quad CPU with four cores of 2.83 GHz each, and 4GB of RAM. The hard drive had 320GB of capacity, with a SATA interface of 3.0Gb/s, and 7200 RPM. We used two real datasets: the $basket$ dataset of information about clients retained by a market store, and the $gene$ dataset of genetic information of patients. The $basket$ dataset has $n = 95570$ and $d = 35$; it was collected with the purpose of client classification and for linear regression; the output variable is the credit line of a new client. On the other hand, the $gene$ dataset was obtained from *GEO Gene Expression Omnibus: a gene expression/molecular abundance repository*. The dataset has $n = 176$ and $d = 36864$; samples are from kidney tissue with renal clear cell carcinoma and non cancer tissue. In our experiments, the output variable is to predict the one gene of interest given the rest of the gene values.

All experiments were repeated five times before reporting an average. First, we evaluate convergence of sequences obtained with our algorithms to compute variable selection. The second set of experiments shows the performance of our optimization to efficiently compute SSVS.

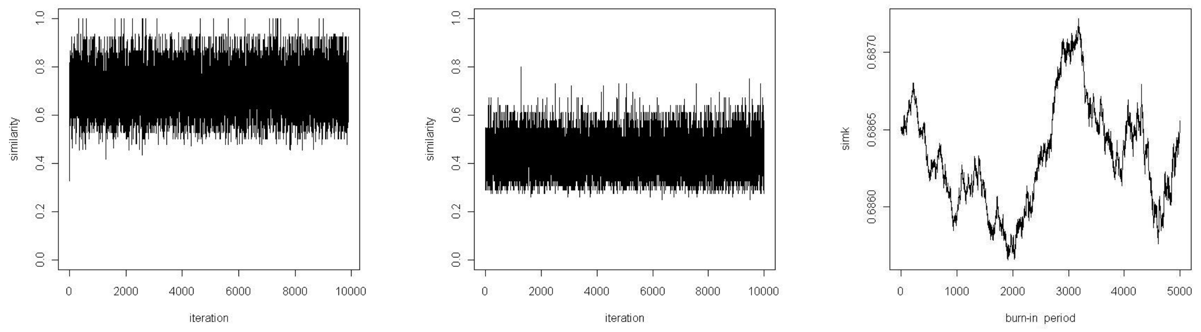
A. Evaluation of Convergence

In this section we analyze convergence of SSVS. Charts of the cosine similarity $sim(\gamma^I, \gamma^{I+k})$, of consecutive elements in a sequence with a lag k, are used to visualize dispersion. the importance of the initial model γ^0, and the influence of the burn-in period to estimate the posterior distribution.

Figure 1 presents charts of measurements to assess convergence of SSVS on the $basket$ dataset, where $d = 32$, $n = 90K$, and $S = 10k$ iterations were drawn. After plotting the similarity $sim(\gamma^I, \gamma^{I+100})$ of iterations I separated by a lag $k = 100$, it can be seen that the sequence stabilized, and models from the posterior distribution appeared repeatedly. Even though Figure 1.a provides valuable information to define a burn-in period, other measurements have also to be taken into consideration. The similarity of the initial γ^0 with the rest of the sequence is shown in Figure 1.b. Draws in the chain rapidly disperse from the initial parameter γ^0, assuring that exploration of the parameter space does not get stuck in a local optimal. The measurement $sim_{10}(\gamma^{burn,\ldots,10K})$ is the average similarity in the sequence $\gamma^{burn,\ldots,10K}$ when the lag $k = 10$; it is used to evaluate candidate burn-in periods. Figure 1.c provides little information to estimate a burn-in period, as values for the measurement vary in a tight range and there is no significant impact of the burn-in period between the first $5K$ iterations. The output analysis of the sequence has given enough information to define an appropriate burn-in period of few iterations, and to expect the sequence to be distributed from $\pi(\gamma|Y, X)$.

B. Analysis of Optimization

In this section, we analyze execution performance of SSVS inside the DBMS. Table I presents the execution time of the main steps of the summarization matrices, and the regression modeling. SSVS was performed over $basket$ information with $n = 1M$, and $S = 10K$ iterations were

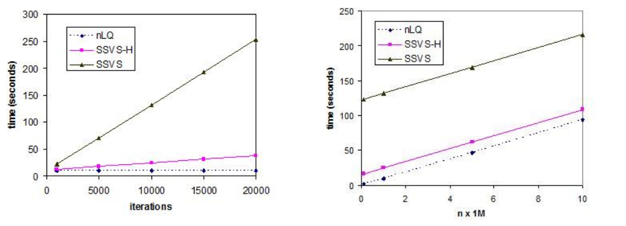


(a) $sim(\gamma^I, \gamma^{I+100})$ (b) $sim(\gamma^0, \gamma^I)$ (c) $sim_{10}(\gamma^{burn,...,10k})$

Figure 1. Output Analysis of an SSVS Execution ($basket$ dataset: $n = 90k$, $d = 32$, $S = 10k$)

Table I
EXECUTION PERFORMANCE OF SSVS ($n = 1M$, $S = 10K$) (TIME IN SECONDS)

d	nLQ	Model	Model with Hash
8	4.645	1.649	0.783
16	6.297	11.401	2.727
32	9.759	122.246	15.141
64	17.371	36.977	43.75



(a) iterations (b) records

Figure 2. Execution Performance of the SSVS from Summarization n, L, and Q ($n = 90K$, $d = 32$, $S = 10K$)

executed. The summarization step is efficiently computed with UDFs with linear scalability on the number of dimensions d. However, execution performance not only depends on the number of dimensions d, but also on the problem to solve. For SSVS from $d = 32$ to $d = 64$, there is no increment on the execution time. In the model sequence of $d = 64$ few variables are analyzed at a time. Therefore, matrix operations have little number of variables involved, and models are evaluated relatively fast. In contrast, when $d = 32$ the number of variables which appear in the models to evaluate are greater than $d = 64$. Consequently, the average selection probability in $d = 64$ is computed faster than in the sequence for $d = 32$. On the other hand, we have that optimizations to the model computation improve performance when the sequence converges. Since SSVS converges within the $S = 10K$ iterations for problems where $d \leq 32$, execution performance is improved with the hash table optimization. Nevertheless, the sequence for the problem $d = 64$ does not reach convergence; execution performance of the optimization decreases in the model computation step, due the overhead of looking for inexistent probabilities in the hash table. Moreover, further iterations must be done for $d = 64$ to reach the state of convergence, and probably to see some benefit from the hash table optimization. Since data summarization is computed once, the number of iterations does not affect the summarization step. It can be seen in Figure 2.a the impact of the hash table optimization during the model computation step. SSVS has linear scalability on the number of iterations in the sequence. Once the summarization step is completed, the model computation does not require more scans to the input table. Consequently, SSVS has also linear scalability on the number of records n (see Figure 2.b). Since summarization matrices of our experiments have size 32×32, the model computation step is unaffected by the number of records n.

C. SSVS for High-Dimensional Data

In this set of experiment, we test execution performance of our algorithm to compute SSVS for high-dimensional data. Since the number of dimensions in the model is not expected to surpass certain threshold set as a parameter, SSVS-HD caches X_γ in main memory. The execution performance of SSVS-HD is presented in Table II. Only when $n = 32$, $d = 1000$, and $S = 100$ there is no time improvement by hash table optimization, due the little exploration of the posterior distribution. If S is increased, convergence of the sequence is reached and the benefit of the hash table optimization can be seen again when $n = 32$, $d = 1000$, and $S = 1000$. Since convergence of SSVS is reached relatively fast, the number of iterations can be increased with almost linear scalability. During our experimental session, SSVS-HD was restarted whenever the number of dimensions in X_γ surpassed the threshold $d_\gamma \leq 40$. By restarting SSVS-

Table II
EXECUTION PERFORMANCE OF SSVS FOR HIGH-DIMENSIONAL DATA (IN SECONDS)

n	d	S	SSVS-HD	SSVS-HD with HASH
32	100	100	3.38	3.05
32	500	100	23.42	7.61
32	1000	100	52.47	62.24
32	100	1000	31.76	27.06
32	500	1000	233.60	36.69
32	1000	1000	949.49	173.98
64	100	100	10.805	10.225
64	500	100	61.965	34.128
64	1000	100	141.208	38.525

HD we avoid getting stuck into local optima, and search is guided toward models bellow the desired threshold.
The experimental evaluation of our algorithms and measurements for SSVS has shown the feasibility to efficiently implement Bayesian statistics for large data inside the DBMS. Converge evaluation is sufficient to define adequate burn-in periods to estimate the posterior distribution of interest. Finally, all the algorithms have good scalability on the number of records n, and the number of dimensions d.

V. RELATED WORK

Variable or feature selection is a well researched topic in both data mining and statistical literature, leading to a variety of algorithms for searching the model space and selection criteria for choosing between competing models [9]. In the Bayesian framework, Markov chain Monte Carlo methods can be used for inference and estimation, such in [2], [4]. The primary task then is to estimate the marginal posterior probability, such a setup has been used in various variable selection settings [8], but were limited to cases when the size and dimensionality of the data is relative small.
Integrating data mining and statistical techniques into a DBMS has been paid little attention. Most of the work has focused on efficiently implementing sufficient statistics for linear models [11], [12], [10]. Summarization of matrix multiplication to compute models and probabilities in large datasets has been explored in [3]. We have extended previous work to yield the DBMS with Bayesian model computations which turned out to be quite challenging to incorporate into the DBMS.

VI. CONCLUSIONS

In this paper, we propose integrating SSVS into the DBMS for processing large amounts of data. Our algorithms show to efficiently perform variable selection for a large number of record, and high-dimensional data. The algorithm for a large n expresses model selection probabilities in terms of summary matrices, where only one table scan is done before computing SSVS. The model computation is not affected by the number of records, while the summarization step shows linear scalability on both n and d. Since computations of order d are demanding for high-dimensional data, we introduced an algorithm which exploits caching to select models from a large number of explanatory variables. Finally, the storage layout is defined to reduce I/O operations, and to suit the data access of the database APIs. There are several issues for future research. It is important to investigate methods to improve the models obtained with SSVS for high-dimensional data. Classification is an important problem in data mining, which under the Bayesian framework can be performed with logistic regression. Furthermore, yielding the DBMSs with the basic operations in Bayesian statics would greatly aid the analysis of large data.

REFERENCES

[1] J. Blakeley, V. Rao, I. Kunen, A. Prout, M. Henaire, and C. Kleinerman, ".NET database programmability and extensibility in microsoft sql server," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 1087–1098, ACM, 2008.

[2] V. Baladandayuthapani, B. Mallick, M. Hong, J. Lupton, N. Turner, and R. Carroll, "Bayesian hierarchical spatially correlated functional data analysis with application to colon carcinogenesis," *Biometrics*, vol. 64, pp. 64–73, March 2008.

[3] P. Drineas, M. Mahoney, and S. Muthukrishnan, "Sampling algorithms for l2 regression and applications," in *Proc. SODA*, pp. 1127–1136, 2006.

[4] E. I. George and R. E. Mcculloch, "Variable selection via gibbs sampling," *Journal of the American Statistical Association*, vol. 88, no. 423, pp. 881–889, 1993.

[5] G. Graefe, U. Fayyad, and S. Chaudhuri, "On the efficient gathering of sufficient statistics for classification from large SQL databases," in *Proc. ACM KDD Conference*, pp. 204–208, 1998.

[6] M. Jaedicke and B. Mitschang, "On parallel processing of aggregate and scalar functions in object-relational DBMS," in *ACM SIGMOD Conference*, pp. 379–389, 1998.

[7] J. M. Marin and C. P. Robert, *Bayesian Core: A Practical Approach to Computational Bayesian Statistics*. Springer Publishing Company, Incorporated, 2007.

[8] T. J. Mitchell and J. J. Beauchamp, "Bayesian variable selection in linear regression," *Journal of the American Statistical Association*, vol. 83, no. 404, pp. 1023–1032, 1988.

[9] A. Miller, *Subset Selection in Regression*. Chapman & Hall/CRC, 2002.

[10] M. Navas and C. Ordonez, "Efficient computation of PCA with SVD in SQL," in *KDD Workshop on Data Mining using Matrices and Tensors (DMMT)*, 2009.

[11] C. Ordonez, "Statistical model computation with UDFs," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 22, 2010.

[12] C. Ordonez and S. Pitchaimalai, "Fast UDFs to compute sufficient statistics on large data sets exploiting caching and sampling," *Data & Knowledge Engineering*, vol. 69, no. 4, pp. 383 – 398, 2010.