SN Computer Science (2021) 2:396
https://doi.org/10.1007/s42979-021-00788-1

ORIGINAL RESEARCH

Backward Stepwise Elimination: Approximation Guarantee, a Batched GPU Algorithm, and Empirical Investigation

Benjamin Sauk[1] · Nikolaos V. Sahinidis[2]

Received: 17 February 2021 / Accepted: 19 July 2021 / Published online: 31 July 2021

Abstract
Best subset selection is NP-hard and expensive to solve exactly for problems with a large number of features. Practitioners often employ heuristics to quickly obtain approximate solutions without any accuracy guarantees. We investigate solving the best subset selection problem with backward stepwise elimination (BSE). We prove an approximation guarantee for BSE that bounds its performance by applying the concept of approximate supermodularity. This guarantee provides conditions that suggest the backward stepwise elimination algorithm will return a near-optimal solution, or when another technique should be used. To improve computational performance of the algorithm, we develop a graphics processing unit (GPU) parallel BSE that averages up to 5x faster than an efficient CPU implementation on a collection of over 1.8 million problems; larger problems resulted in the largest speedups. Finally, we demonstrate the benefit of BSE with empirical results, comparing against several state-of-the-art feature selection approaches. For certain classes of problems, BSE generates solutions with lower relative test error than the lasso, the relaxed lasso, and forward stepwise selection. BSE thus deserves a place in the data modeling toolset along with these other more popular methods. All codes and data used for computations in this paper can be obtained from https://github.com/bsauk/BackwardStepwiseElimination.

Keywords Regression · Model selection/variable selection · Algorithms · GPU computing

Introduction

Feature selection is the problem of identifying a subset of features that succinctly and accurately relate a set of input observations to output measurements. A popular way to address the problem of feature selection is by solving, often approximately, the best subset selection problem, i.e., the problem of finding a small subset of features, so that the resulting linear model provides an accurate representation of the measurements [23].

The best subset selection problem is known to be NP-hard [1]. When solving with branch-and-bound [13], mixed-integer optimization [4, 8], or exhaustive enumeration, optimal subset selection can become intractable for problems with a large number of features. Instead, heuristics, such as forward stepwise selection (FSS), backward stepwise elimination (BSE) [11], or the lasso [31], are commonly used to identify near-optimal subsets for large instances [19, 28]. Although heuristic approaches are significantly faster than exact methods, there are few studies that have investigated the accuracy of these methods.

Even when it is possible to solve the subset selection problem exactly, the mathematically optimal model may not be the best choice in practice. Hastie et al. [16] compared the performance of FSS, the lasso [31], the relaxed lasso [22], and an exact mixed-integer formulation [4]. The comparisons did not consider BSE, thus leaving a gap in the understanding of this technique in comparison to other approaches.

We investigate the benefits of solving the best subset selection problem with a backward stepwise elimination algorithm. The contributions of this paper are:

1. We obtain an approximation guarantee for BSE using the concept of supermodularity ratio. The derived guar-

✉ Nikolaos V. Sahinidis
nikos@gatech.edu

1 Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

2 H. Milton Stewart School of Industrial & Systems Engineering and School of Chemical & Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

antee provides a bound on the worst-case performance of backward stepwise elimination.
2. We develop a GPU parallel batched BSE algorithm that is a factor of 5x faster than a CPU implementation of BSE for a range of problem sizes.
3. We compare the accuracy of BSE and other state-of-the-art subset selection methodologies. We demonstrate that, for certain classes of problems, BSE generates models that are simpler and have less out-of-sample test error than the lasso or forward selection.

The remainder of this paper is organized as follows. In the next section, we review the literature related to best subset selection, stepwise selection, approximate submodularity, and supermodularity. In the third section, we rely on submodularity to prove an approximation guarantee for BSE. In the fourth section, we propose a batched GPU BSE algorithm, describe our implementation, and compare the performance of the proposed GPU algorithm against a CPU implementation. In the fifth section, we compare BSE against other popular subset selection techniques in terms of solution quality. We provide conclusions in the last section.

Literature Review

Best Subset Selection Problem Formulation

Given a response vector $\boldsymbol{y} \in \mathbb{R}^m$, predictor matrix $\mathbf{A} \in R^{m\times n}$, whose rows correspond to measurements and columns correspond to features, and a subset size $k \leq n$, the best subset selection problem is defined as the following optimization model:

$$\min_{\boldsymbol{x}} \|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|_2^2, \text{ subject to } \|\boldsymbol{x}\|_0 \leq k, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ are the coefficients of a linear predictive model. The ℓ_0 norm limits the number of nonzero coefficients and adds nonconvexity to an otherwise convex problem.

Many techniques have been developed to solve this problem, including branch-and-bound [13, 14], mixed-integer optimization [4, 8], and screening rules to reduce the size of the solution space [29]. Several heuristic approaches consider relaxing the cardinality constraint, producing the following problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|_2^2, \tag{2}$$

which can be solved in closed form. A least squares estimator of $\boldsymbol{x}$ can be found to solve $\boldsymbol{y} = \mathbf{A}\boldsymbol{x} + \epsilon$, where $\epsilon \in \mathbb{R}^m$. For the remainder of the paper, we will assume that $m \geq n$ There are many techniques to solve the linear least squares problem, with QR factorization being one of the most commonly used. QR factorization involves decomposing a matrix $\mathbf{A} \in R^{m\times n}$ into the product of an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{m\times m}$ and an upper triangular matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$:

$$\mathbf{Q}^{\mathrm{T}}\mathbf{A} = \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix}. \tag{3}$$

Let

$$\mathbf{Q}^{\mathrm{T}}\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{w} \end{pmatrix}, \tag{4}$$

where $\boldsymbol{y}_1 \in \mathbb{R}^n$ and $\boldsymbol{w} \in \mathbb{R}^{m-n}$. The least squares estimator is obtained by solving the following optimization problem

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|_2^2 = \arg\min_{\boldsymbol{x}} \left\|\mathbf{Q}^T(\boldsymbol{y} - \mathbf{A}\boldsymbol{x})\right\|_2^2. \tag{5}$$

From this, the residual sum of squares (RSS) can be calculated from

$$\mathrm{RSS} = \|\boldsymbol{y} - \mathbf{A}\hat{\boldsymbol{x}}\|_2^2 = \left\|\mathbf{Q}^{\mathrm{T}}(\boldsymbol{y} - \hat{\boldsymbol{x}})\right\|_2^2 = \|\boldsymbol{w}\|_2^2. \tag{6}$$

If QR factorization is used, the residual sum of squares is calculated from the Euclidean norm of $\mathbf{Q}^T\boldsymbol{y}$ for the vector of values from $n+1$ to m.

Stepwise Selection and Elimination

A common technique for solving (1) involves selecting or eliminating variables, in a stepwise fashion. Forward stepwise selection initially generates a model that minimizes RSS by selecting a single variable. Then, in each subsequent iteration, a new variable is included in the solution until $\|\boldsymbol{x}\|_0 = k$. In every iteration, a new model is obtained by identifying the variable that minimizes RSS when added to the previously obtained model. Forward stepwise selection (FSS) is a greedy selection algorithm, which has a provable worst performance for certain classes of problems [24]. Forward selection can also be used for problems when the optimal subset size is not known a priori. Stopping rules for FSS aim to find a balance between accuracy and model complexity [3].

Backward stepwise elimination (BSE) starts from the standard least squares solution and removes one feature at a time until the cardinality constraint is satisfied. Given the initial least squares solution $\boldsymbol{x}_0$, the error for the model after s iterations and the corresponding subset $\boldsymbol{x}_s$ are obtained via factorization or QR downdating [6]. QR downdating refers to updating the solution to the linear least squares problem when a column or a row is removed from $\mathbf{A}$. For subset selection, a model with k columns is selected, then RSS is calculated, where $\boldsymbol{y}_2 \in \mathbb{R}^{m-k}$. QR downdating reduces the number of floating point operations by removing a column from $\mathbf{R}_s$ and updating $\mathbf{Q}$ to maintain an upper triangular structure in $\mathbf{R}_{s-1}$ without having to perform a QR factorization at every iteration. When a column is removed from $\mathbf{R}_{s-1}$, the upper triangular structure is only destroyed in the columns to the right of the deleted column. As the computational complexity of QR factorization is $O(2mn^2)$, in many cases, the complexity of the update is significantly less than the cost of factorizing $\mathbf{A}_{s-1}$. The sum of squared errors is computed by left multiplying $\mathbf{Q}^\mathrm{T}$ to restore the upper triangular structure $\mathbf{R}_{s-1}$ with $\boldsymbol{y}_2$. This procedure is repeated to calculate the best $i = n, \ldots, n_{\min}$ models, where $n_{\min} \in [1, k]$. In each iteration, after the best model has been identified, all suboptimal solutions are discarded, and the next iteration begins. This algorithm is outlined in Algorithm 1.

Algorithm 1 Generic backward stepwise elimination algorithm

```
1: Given a set of data points $\boldsymbol{x}_{ij}, \boldsymbol{y}_i$ for $i = 1, ..., m$, $j = 1, ..., n$
2: Generate a set of basis functions from input features $\mathbf{A}$
3: procedure BackwardElimination($\mathbf{A}$)
4:     $\mathbf{Q}_n\mathbf{R}_n \leftarrow \mathbf{A}$
5:     for $k = n-1, ..., 0$ do
6:         for $h = 1, ..., k$ do
7:             $\mathbf{A}_{k,h} \leftarrow \mathbf{R}_{k+1}$
8:             $\mathbf{Q}_{k,h}\mathbf{R}_{k,h} \leftarrow QR(\mathbf{A}_{k,h}(h:m, h:n))$
9:             $\boldsymbol{w}_{k,h} \leftarrow \mathbf{Q}^T_{k,h}\boldsymbol{w}_{k+1}$
10:            $\mathbf{RSS}_{k,h} \leftarrow \|\boldsymbol{w}_{k,h}(k+1:m)\|_2^2$
11:        end for
12:        $i \leftarrow \arg\max_h \boldsymbol{w}_{k,h}$
13:        $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_{k,i}$
14:        $\mathbf{A}_k \leftarrow \mathbf{A}_{k,i}$
15:    end for
16: end procedure
```

Both stepwise techniques have been extensively used for the last 50 years [12]. In most cases, BSE requires more floating point operations than FSS. However, the accuracy of BSE has been observed to be better than FSS for certain classes of problems [7, 21]. While some have argued that neither approach should be used [26], for problems with millions of observations and thousands of features, stepwise approaches quickly generate approximately accurate and sparse solutions.

FSS and BSE are heuristic hill climbing strategies that obtain locally optimal solutions to the subset selection problem. As both methods require fewer computations than exact strategies, researchers have investigated if any guarantees exist for these methods. Several authors have proven statistical bounds on the accuracy of FSS [9, 24]. Using the notion of the submodularity ratio, it is possible to obtain a worst-case bound on the performance of FSS. Under certain conditions, BSE identifies an optimal subset [7]. Unfortunately, checking the conditions in [7] requires the solution of an NP-hard problem.

Submodularity and Supermodularity

A function f that maps a set to a real number is called submodular if it satisfies the following property:

$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T), \tag{7}$$

for $S \subset T$ and $\{v\} \subset T \setminus S$. The results of Nemhauser et al. [24] prove that the greedy algorithm achieves a $(1 - 1/e)$-approximation for the maximization of any monotone, submodular set function over a cardinality constraint. Here, e is the base of the natural logarithm. This approximation result provides a lower bound on the performance of greedy algorithms for solving NP-hard problems subject to cardinality constraints. However, subset selection does not involve a submodular objective function. To develop an approximation guarantee for subset selection, the work of [9] defines the submodularity ratio as a way to measure how close a function is to being submodular:

$$\gamma_{U,k}(f) = \min_{L \subset U, S : |S| \leq k, S \cap L = \emptyset} \frac{\sum_{x \in S} f(L \cup \{x\}) - f(L)}{f(L \cup S) - f(L)}, \tag{8}$$

where f is a set function, $L \subset U$ and $S \cap L = \emptyset$. The submodularity ratio is a function of the maximum subset size k, and the set U. It reflects how much the value of f increases by adding any subset S of size k to L, compared to the benefit of $f(S \cup L)$. If the function f is submodular, then the submodularity ratio is defined to be 1, otherwise if $\gamma < 1$, the function is defined as weakly submodular. Das and Kempe [9] prove that FSS has a worst-case approximation guarantee of $1 - \exp(-\gamma) \cdot OPT$, where OPT is the R^2 of the optimal best subset solution. For $\gamma = 1$, the guarantee in [9] recovers the guarantee of Nemhauser et al.; the bound is loose as γ approaches zero.

A function f is supermodular if $-f$ is submodular. Several authors have defined a supermodularity ratio [18, 20, 27]. Inspired by the work of Sakaue [27], we define the following supermodularity ratio:

$$\beta_{U,k}(f) \geq \frac{\sum_{x \in S} f(L \setminus \{x\}) - f(L)}{f(L \setminus S) - f(L)}, \tag{9}$$

where $\beta_{U,k} \in [1, k]$ is selected as the maximum value for each combination of $S, L \subseteq U$. Like the submodularity ratio, the supermodularity ratio captures how close a function is to being supermodular.

Algorithmic Analysis of BSE

While there exist approximation guarantees for forward selection, no such bound is currently known for backward stepwise elimination. To determine such a bound, we use the concept of the supermodularity ratio.

Let f be a nonnegative monotonically increasing set function. The problem we seek to solve is

$$\max_S f(S), \text{ subject to } \|S\|_0 \leq k. \tag{10}$$

Our theoretical contribution is an approximation guarantee on the performance of backward stepwise elimination.

Theorem 1 *Let f be a nonnegative, monotonically increasing set function, OPT be the maximum value of f possible for a set of size k, and k^* be the size of the subset for OPT. Then, the set selected by BSE, S^{BSE}_{n-k}, has the following approximation guarantee:*

$$f(S^{\text{BSE}}_{n-k}) \geq \left(1 - \frac{\beta}{k^*}\right)^{n-k^*} \cdot \text{OPT}. \tag{11}$$

Proof Let S^B_0 be the initial set of all variables considered, and S^*_{n-k} be an optimal set of k variables that has a value of OPT. Let S^B_i be the set of variables that remain in S after i iterations of BSE. We begin by rearranging the supermodularity ratio to ensure that the numerator and denominator in (9) are both positive:

$$\sum_{x_j \in S^B_i} \left(f\left(S^B_i\right) - f\left(S^B_i \setminus \{x_j\}\right)\right) \leq \beta\left(f\left(S^B_i\right) - f\left(S^B_i \setminus S^B_i\right)\right). \tag{12}$$

In every iteration of BSE, $\hat{x}$ is selected to minimize $f(S^B_i) - f(S^B_i \setminus \{\hat{x}\})$. As the minimum size of S^B_i is $|S^B_i| \geq k$ and $\sum_i^n x \geq n \cdot x_{\min}$, we have

$$k^*\left(f(S^B_i) - f(S^B_{i+1})\right) \leq |S^B_i|\left(f(S^B_i) - f(S^B_{i+1})\right) \leq \beta f(S^B_i). \tag{13}$$

Letting $A(i)$ be the loss in f in iteration i, $A(i) = f(S^B_{i-1}) - f(S^B_i)$. Let $f(S^B_0)$ be the value of f when all variables in the set are included. Then, $\sum_{j=1}^i A(j) = f(S^B_0) - f(S^B_i)$ extends from the definition of $A(i)$. Rewriting (13) in terms of $A(\cdot)$, we get

$$A(i+1) \leq \frac{\beta}{k}\left(f\left(S^B_0\right) - \sum_{j=1}^{i} A(j)\right). \tag{14}$$

Using the inequality above, we will prove by induction that

$$f(S^B_0) - \sum_{j=1}^{t} A(j) \geq f\left(S^B_0\right)\left(1 - \frac{\beta}{k}\right)^{n-k} \geq \left(1 - \frac{\beta}{k}\right)^{n-k} \cdot OPT \tag{15}$$

For $t = 0$, the inequality is trivial. Assume that the inequality holds for t iterations. Then, for iteration $t + 1$:

$$\begin{aligned} f\left(S^B_0\right) - \sum_{j=1}^{t+1} A(j) &= f\left(S^B_0\right) - \sum_{j=1}^{t} A(j) - A(t+1) \\ &\geq f\left(S^B_0\right) - \sum_{j=1}^{t}\left(A(j) - \frac{\beta}{k}\left(f\left(S^B_0\right) - \sum_{j=1}^{t} A(j)\right)\right) \\ &\geq \left(f\left(S^B_0\right) - \sum_{j=1}^{t} A(j)\right)\left(1 - \frac{\beta}{k}\right) \\ &\geq f\left(S^B_0\right)\left(1 - \frac{\beta}{k}\right)^{t+1} \\ &\geq \left(S^B_0\right)\left(1 - \frac{\beta}{k}\right)^{n-k} \end{aligned}$$

where $t + 1$ is less than or equal to $n - k$. Finally, $f(S^B_0) \geq OPT$ and from the definition of $f(S^{BSE}_{n-k^*}) = f(S^B_0) - \sum_{j=1}^{n-k} A(j)$:

$$f(S^{\text{BSE}}_{n-k}) \geq \left(1 - \frac{\beta}{k}\right)^{n-k} \cdot \text{OPT}. \tag{16}$$

This completes the proof for the approximation guarantee. □

We apply this theorem to the best subset selection problem by defining $f(S) = R^2_S$. When $\beta = 1$, our approximation is the tightest, and deteriorates until $\beta = k$. This implies that the proposed guarantee is stronger for functions that are closer to supermodular, similar to the submodularity ratio for submodular functions. In addition, the proposed bound is stronger as k approaches n, where in the case that $k = n$, BSE returns the linear least squares solution.

A Batched GPU Algorithm for BSE

One major criticism against BSE is that it is computationally expensive. To address this shortcoming, in this section, we develop a parallel BSE algorithm using batched GPU computing.

Background

To reduce the computational time of BSE, we parallelized the QR downdate operations in each iteration of Algorithm 1. Unfortunately, in every iteration, each downdate task is unique. As a result, it cannot be accelerated with a data-parallel framework. In particular, the problem size and batch size decrease in each iteration, and every downdate requires a different number of matrix update operations. Instead of a data-parallel approach, we parallelized tasks with batched GPU computing.

GPUs are powerful accelerators that are designed for single instruction multiple data parallelism—not task-level parallelism. GPU hardware is designed for rendering graphics and performing the same set of operations on different sets of data. There are many cases where thousands of small, independent problems need to be solved. To take advantage of GPU hardware for scientific computing, "batching" is a technique that solves groups of problems in parallel [15, 25]. While algorithms designed for batched computing do not fully utilize the hardware, batched methods have been observed to be a factor of 2x faster than optimized CPU kernels for performing the same set of instructions [10].

Despite the clear need to solve problems in batches, developing software to execute task-level parallelism on a GPU efficiently is challenging. To fill this gap, two batched dense linear algebra libraries have been developed. In CUBLAS, NVIDIA developers have created a set of batched basic linear algebra subroutines (BLAS) and batched kernels for QR factorization, LU factorization, and matrix-matrix multiplication [25]. The Innovative Computing Laboratory developed MAGMA, and implemented efficient batched BLAS routines to accelerate batched linear algebra kernels [15]. MAGMA has demonstrated that, with proper algorithm-specific optimizations, it is possible to develop algorithms that are twice as fast as CUBLAS, for problems with a large batch size. In addition, for small- to medium-sized problems, batched BLAS approaches are reported as 2–3x faster for batched matrix multiplication compared to traditional GPU code [15].

Batched BSE

We employed the batched QR factorization routine available in the MAGMA 2.5.0 library [17]. While batched QR factorization is the most time consuming portion of the backwards stepwise algorithm, a BSE algorithm also needs to perform downdates on the output $\boldsymbol{y}$ to calculate the sum of squared errors for every problem in a batch. Unfortunately, MAGMA does not have a batched implementation to perform QR downdates. In LAPACK, this functionality corresponds to the routine DORMQR, which uses $\mathbf{Q}$ generated from DGEQRF and calculates $\mathbf{Q}^T\boldsymbol{y}$.

We augmented the DGEQRF routine to include the update operation on $\boldsymbol{y}$. We modified the batched routine to update $\boldsymbol{y}$ when the rest of the matrix $\mathbf{A}$ is updated. When $\boldsymbol{y}$ was a vector, updating $\boldsymbol{y}$ added a negligible amount of time. We conducted experiments, and observed that the computational time of the modified code did not increase compared to that of the original batched DGEQRF code.

The batched BSE algorithm computes a solution to the linear least squares problem. Then, in parallel, the algorithm removes different single features from the linear least squares solution. Each task then downdates $\boldsymbol{y}$ to calculate the updated sum of squared errors. After factorizing and updating $\boldsymbol{y}$ for the removal of each candidate variable, the column with the smallest change in SSE is removed. This process is repeated until terminating at a predefined minimum matrix size.

The backwards selection algorithm relies on the QR factorization kernel in MAGMA for tall and skinny matrices. To optimize the performance of this routine, we performed parameter tuning on the block size parameter in MAGMA. From previous work [30], it was observed that varying the block size parameter in MAGMA has a significant impact on performance. We discovered that changing the block size to 16, from a default value of 32 improved the performance of the batched kernel for problem sizes of interest.

By definition, a batch is made up of problems of the same size. In each iteration, we perform matrix updates that operate on a different number of columns ranging from zero to $n - s_i$ columns, where n is the number of columns in the matrix and s_i is the current iteration. As a result, for every task in the same iteration, the number of operations in each downdate operation is different, depending on which column is removed. If the feature removed from $\mathbf{R}$ is the furthest to the right, no work is needed to downdate the solution. However, if the first column is removed, the entire matrix needs to be downdated to restore the upper triangular structure of $\mathbf{R}$. This uneven distribution of work creates a batch size of one, where all jobs require different computations. To make BSE amenable to batched computing, we decided to perform downdate operations for every feature as if the entire matrix is to be downdated. By assuming that all problems are the same size, we greatly increase the total number of computations in every batch. This design choice allows us to set the batch size equivalent to the number of features that are candidates to remove in each iteration, i.e., $n - s_i$. Even though doing so increases the total count of operations, our early computational experimentation demonstrated that the proposed batch methodology is a reasonable option. In particular, the proposed algorithm is faster than a sequential CPU BSE implementation and a GPU BSE implementation. An outline of the algorithm listed above is detailed in Algorithm 2

For each problem size, we generated 10 instances and calculated the average execution time. In all, over 1.8 million instances were solved in these experiments. We utilized the MAGMA-2.5.0 [17] library to perform batched least squares calculations, with the modification to the DGEQRF routine that we detailed above to perform QR downdating. We compared the proposed batched GPU algorithm against

Algorithm 2 Batched backward stepwise elimination algorithm

```
1: Given a set of data points $\boldsymbol{x}_{ij}, \boldsymbol{y}_i$ for $i = 1, ..., m$, $j = 1, ...n$
2: Generate a set of basis functions from input features $\mathbf{A}$
3: procedure BatchBackwardElimination($\mathbf{A}$)
4:     $\mathbf{Q}_n\mathbf{R}_n \leftarrow \mathbf{A}$
5:     for $k = n - 1, ..., 0$ do
6:         $\mathbf{A}_k \leftarrow \mathbf{R}_{k+1}$
7:         $\mathbf{Q}_k\mathbf{R}_k \leftarrow QR(\mathbf{A}_k)$
8:         $\boldsymbol{w}_k \leftarrow \mathbf{Q}_k^T \boldsymbol{w}_{k+1}$
9:         $\mathbf{RSS}_k \leftarrow \|\boldsymbol{w}_k(k+1:m)\|_2^2$
10:        $i \leftarrow \arg\max_k \boldsymbol{w}_k$
11:        $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_{k,i}$
12:        $\mathbf{A}_k \leftarrow \mathbf{A}_{k,i}$
13:    end for
14: end procedure
```

Computational Results

We conducted experiments on a machine running CentOS7, with an Intel Xeon E5-1630 at 3.7 GHz and 8 GB of RAM. The machine was equipped with a NVIDIA Tesla K40 GPU with 15 streaming multiprocessors, 12 GB of RAM, and a peak memory bandwidth of 288 GB/s. The algorithms were compiled with the NVCC CUDA 9.1 compiler, using the -03 optimization flag. We generated subset selection problems with randomly generated values between zero and one. We compare the computational time to solve the best subset selection problem for problems with m = 500–1000 over a range of n = 200–600. We consider problems where the number of rows is larger than the number of columns.

a CPU implementation of BSE that relies on LAPACK [2] to perform factorization and QR downdating.

As seen in Fig. 1, the parallel backward elimination algorithm is 2–5x faster than the CPU implementation. In the figure, we report the GPU speedup as a function of the number of columns, for three matrix sizes. We observe that the speedup levels off as the number of columns, or equivalently the batch size, increases. A leveling off of performance is indicative that the computing resources are completely saturated.

The speedup obtained when the number of rows is increased is not as significant as when the number of columns is increased. As the computational complexity of QR factorization scales with the square of the number of

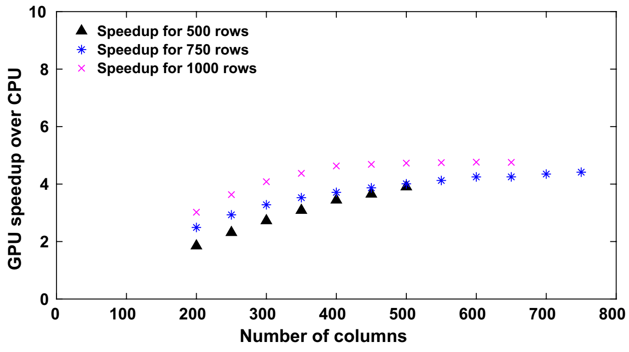


Fig. 1 Speedup as a function of problem size for the backward stepwise elimination algorithm

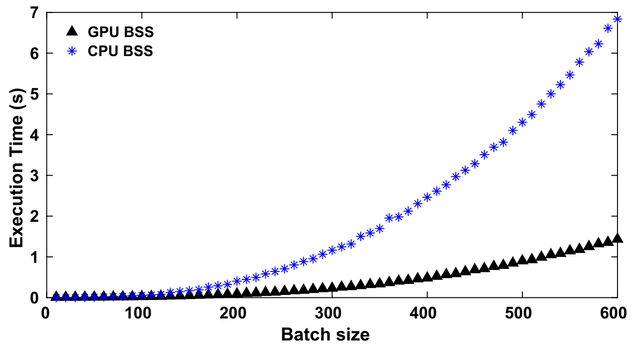


Fig. 2 Execution time as a function of batch size for the backward stepwise selection algorithm for a problem with 1000 rows and 600 columns with a variable batch size between one and 600

columns and linearly with the number of rows, our speedups are in line with the computational complexity of the underlying algorithm.

To reinforce the observation that the speedup for BSE was limited by computational efficiency of the computing resources, we investigated the performance of the algorithm as a function of the batch size. Figure 2 displays the execution time of the CPU and GPU BSE algorithms as a function of batch size. In every iteration of BSE, the batch size was decreased by one. From Fig. 2, we see that the benefits of batched GPU computing decrease as the problem size decreases. For batch sizes of 600, the GPU outperforms the CPU by a factor of 5x. For large batch sizes, above 300, the execution time increases linearly as the problem size increases. A linear relationship between execution time and problem size suggests that performance is limited by a computational bottleneck. Even though the GPU outperforms the CPU for large batch sizes, the speedup decreases to one around a batch size of 50. The CPU is faster than the batched GPU algorithm for small problems. For small problems, the overhead of transferring data to the GPU outweighs the benefits of batched computing.

The astute reader must have noticed that, for problems such as those shown in Fig. 2, the proposed GPU algorithm reduces computational time from 7 to 1.4 s. Obviously, this reduction in computational time is uninteresting in the context of a single problem. The GPU algorithm becomes practically relevant when a large number of problems must be solved. For instance, in the next section, we conduct a comprehensive experiment that relies on the results from 2,020,000 BSE runs.

Accuracy of Backward Stepwise Elimination

Background

Recently, several articles have been published on the topic of best subset selection. With advances in integer programming solvers, researchers have investigated this problem with mixed-integer programming techniques [4, 8]. However, in the statistics community, several have postulated that it may not be worthwhile to solve this problem to optimality on training data [16, 32]. Instead, the use of heuristic approaches like the lasso and forward selection have been investigated and found to perform well for various problems [16]. There also has been work to solve this problem with penalized L1-regression methodologies [5]. In the work of Hastie et al. [16], both the execution time and several out-of-sample statistical metrics are used to compare the lasso, a mixed-integer programming formulation, forward selection, and the relaxed lasso. They discovered that each of the methods obtained the best solution under different problem sizes and data characteristics. In terms of computational time, the mixed-integer programming formulation was the most computationally expensive for all problems considered.

The work of Hastie et al. raised two questions that we investigate in this paper. First, the examples formulated in their work sought to identify a sparse algebraic representation for models with five variables in the true model. However, in practice, modeling complex systems may require complex non-linear equations with more terms. Second, while forward and backward selection have been compared empirically in the literature, we are interested in determining when BSE should be used for subset selection. To facilitate a comparison between these methods, we performed experiments with four techniques:

1. the proposed batched BSE algorithm,
2. forward selection in the R best subset package,
3. the lasso in the R best subset package,
4. the relaxed lasso in the R best subset package.

Despite recent advances in solving mixed-integer problems, for problems of sufficient size, solving the best subset selection problem exactly is still costly. We do not include results for the mixed-integer formulation as the approximate best subset solutions obtained from preliminary experiments were comparable to forward selection.

Experimental Setup

In this section, we make use of the notation proposed in [16]. Our experiments followed a similar procedure to those presented in the Hastie et al. paper, and were conducted on the same machine as in the previous section. Data in our experiments were drawn from distributions that were defined by several parameters. Our matrices were generated by defining a problem size (m, n), a sparsity level s, to indicate the number of nonzeros in the model, and a beta-type, to create a sparsity pattern. In addition, ρ is used to control the correlation level between variables when generating input data, and a signal-to-noise-ratio (SNR) term was used to control the level of noise in the data. Matrices were generated from a true model parameterized by ρ and s. A response vector $\boldsymbol{y}$ was also drawn by sampling points from the true model while adding noise that satisfied a specified SNR.

To compare approaches, several test metrics were evaluated: relative risk, relative test error, proportion of variance explained, and the number of nonzeros in the chosen model. As studied in Hastie et al., relative risk (RR) is a measure of predictive performance:

$$\mathrm{RR}(\hat{\beta}) = \frac{(\hat{\beta} - \beta_0)^T \Sigma (\hat{\beta} - \beta_0)}{\beta_0^T \Sigma \beta_0}. \tag{17}$$

Here, $\hat{\beta}$ is the vector of coefficients selected from regression, β_0 is the vector of true coefficients that are used to generate the data, and Σ represents the correlation between the predictor variables. A perfect RR score for relative risk is zero, corresponding to $\hat{\beta} = \beta_0$. A bad score corresponds to one. Relative test error (RTE) is an out-of-sample procedure for measuring accuracy, which measures the expected test error relative to the Bayes error rate:

$$\text{RTE} = \frac{(\hat{\beta} - \beta_0)^T \Sigma (\hat{\beta} - \beta_0) + \sigma^2}{\sigma^2} \tag{18}$$

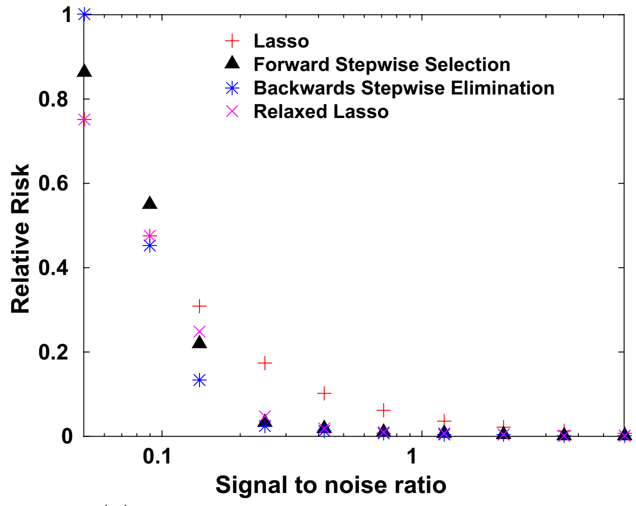
A perfect RTE score is one, while a score of zero corresponds to $\hat{\beta} = 0$. In this formula, σ^2 is the variance used to generate the matrices while satisfying a predetermined SNR.

Proportion of variance explained is the amount of variance explained by the proposed model in the output variable y_0:

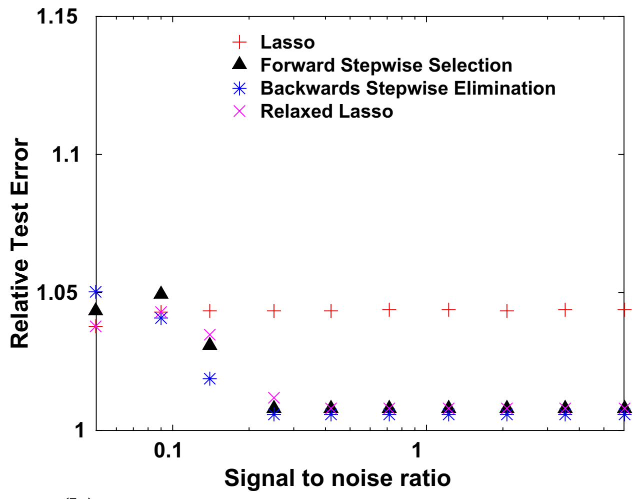
$$\text{PVE} = 1 - \frac{(\hat{\beta} - \beta_0)^T \Sigma (\hat{\beta} - \beta_0) + \sigma^2}{\beta_0^T \Sigma \beta_0 + \sigma^2}. \tag{19}$$

If the true model is selected, PVE equals $\frac{\text{SNR}}{1+\text{SNR}}$, while a null model has a score of zero. The last metric considered is the number of nonzero coefficients selected. In general, sparser models generalize better to validation data.

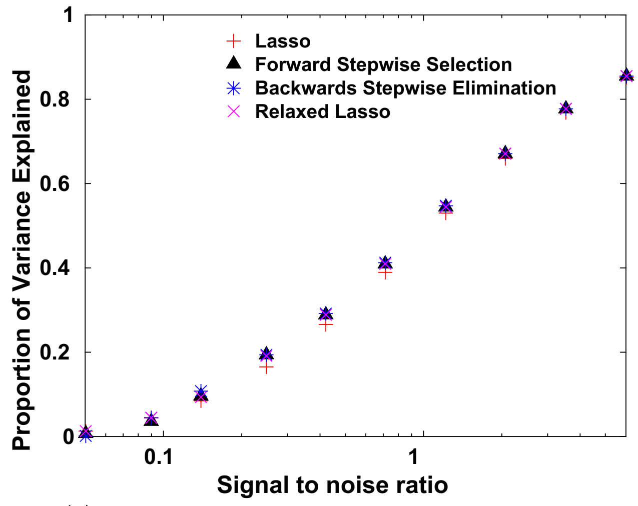
To compare BSE against other subset selection strategies, we conducted experiments with matrices of size $m = 500$, $n = 100$, and $s = 5$. We were also interested in determining which methods are better suited for developing more complex models. We consider s over a range of 10–70 in multiples of



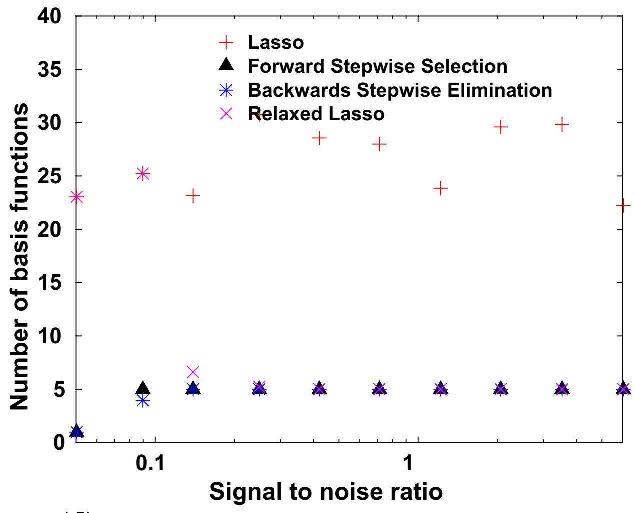
(a) Relative risk as a function of SNR



(b) Relative test error as a function of SNR

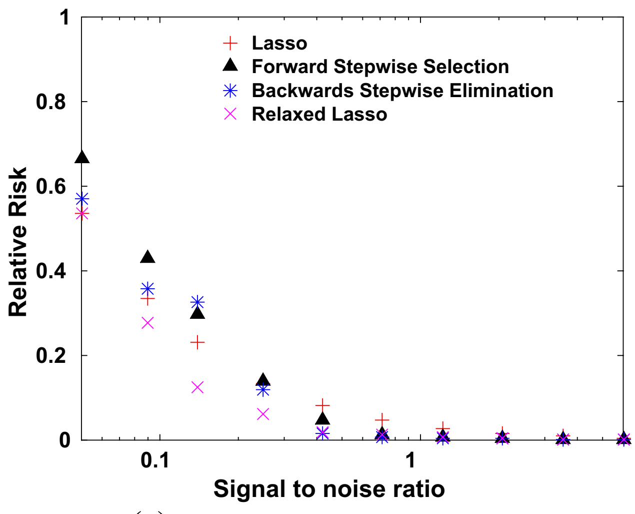


(c) Proportion of variance explained as a function of SNR

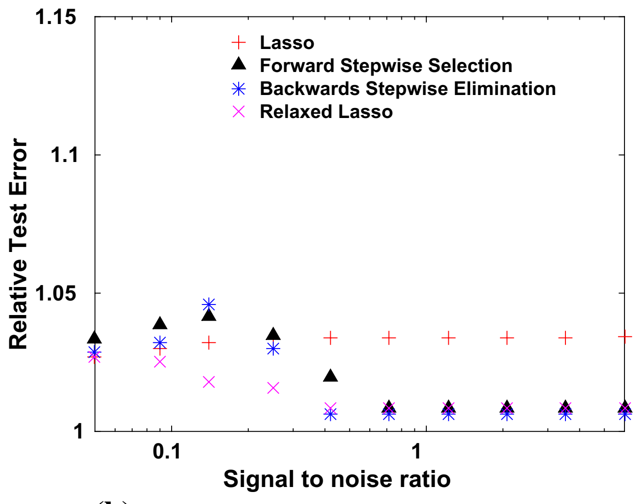


(d) Number of nonzero coefficients as a function of SNR

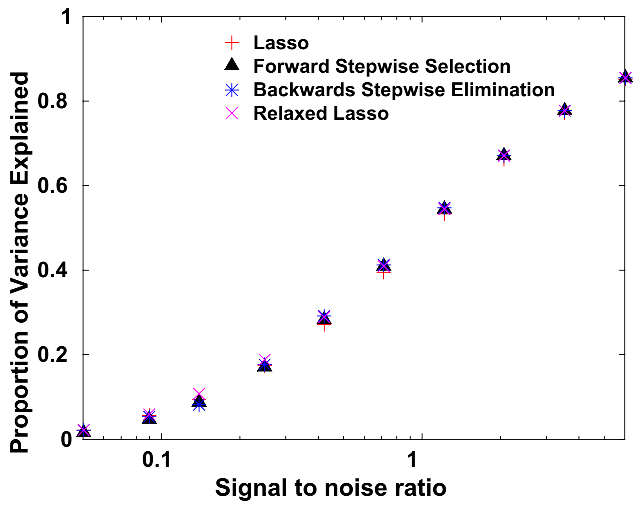
Fig. 3 Four accuracy metrics for the performance of different subset selection techniques for $\rho = 0$



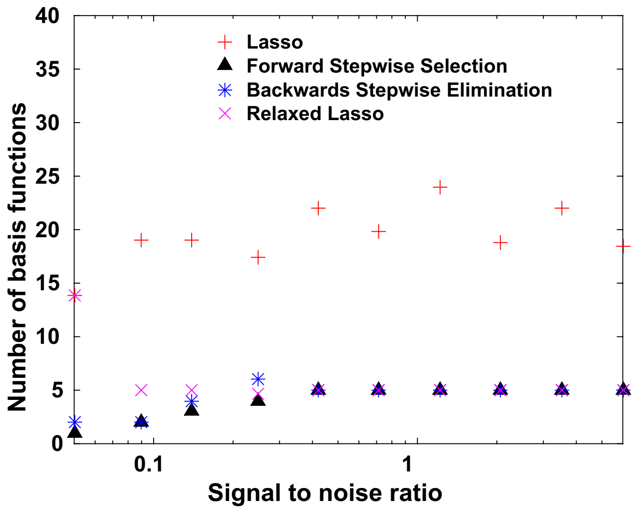
(a) Relative risk as a function of SNR



(b) Relative test error as a function of SNR



(c) Proportion of variance explained as a function of SNR



(d) Number of nonzero coefficients as a function of SNR

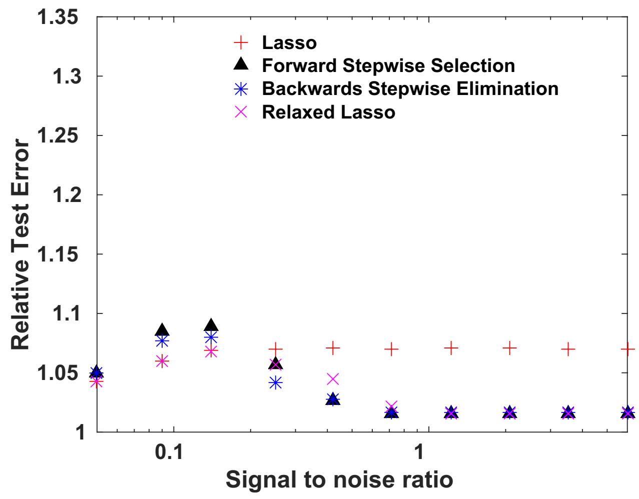
Fig. 4 Four accuracy metrics for the performance of different subset selection techniques for $\rho = 0.35$

20. Experiments for all problem types were conducted over $SNR \in [0.05, 0.09, 0.14, 0.25, 0.42, 0.71, 1.22, 2.07, 3.52, 6]$.

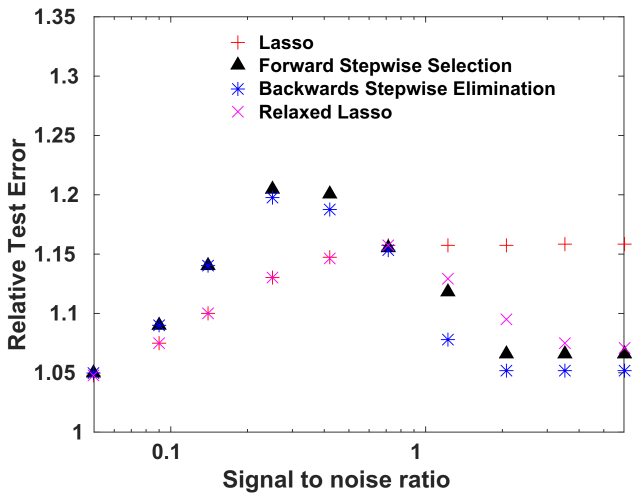
In the work of Hastie et al., multiple methods were used to generate matrices. We used beta-type 2, where β_0 has the first s parameters equivalent to one, with the rest set to zero. Experiments were conducted with ρ equivalent to either 0 or 0.35. All values reported below are an average over five repetitions. For each technique, a solution path was generated for every SNR considered. The results reported below are from the models that minimized the desired test metric from each solution path.

Computational Results

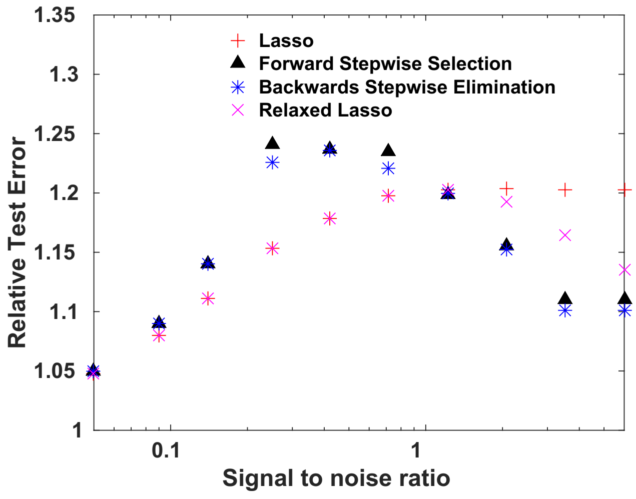
Figures 3 and 4 relate SNR to the accuracy metrics for different correlation levels. The uncorrelated case was unique from the other cases that were observed. BSE and FSS performs differently when SNR$<$ 0.16. In particular, BSE in the low SNR cases outperforms all other methods in regards to RR and RTE. For SNR$>$ 0.16, all the methods except for the lasso converge to low error solutions. The lasso selects denser models than all of the other methods, selecting a 25-term model as opposed to a five-term model.



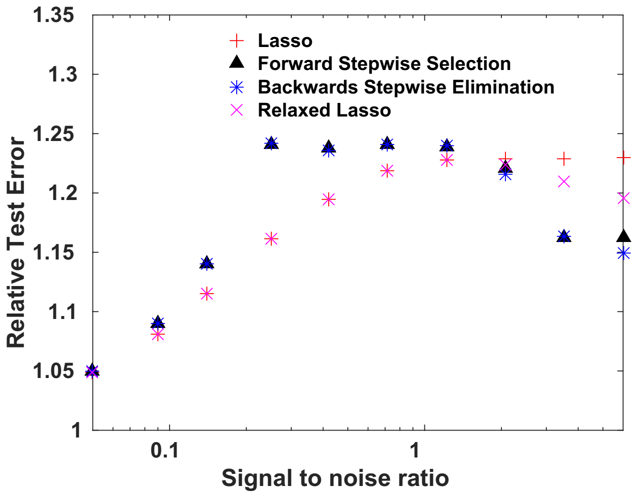
(a) RTE as a function of SNR for $s = 10$



(b) RTE as a function of SNR for $s = 30$



(c) RTE as a function of SNR for $s = 50$



(d) RTE as a function of SNR for $s = 70$

Fig. 5 Four accuracy metrics for the performance of different subset selection techniques when the number of nonzero coefficients in the real model changes for problems with $\rho = 0$

The results suggest that BSE outperforms the other methods at SNR< 0.16. The relaxed lasso and lasso both select denser solutions than BSE for these problems. BSE outperforms FSS because FSS selects several variables in early iterations that hinder its overall performance as k increases. For this case of noisy data with no correlation, BSE selects a sparser model than the relaxed lasso, leading to a smaller RTE.

At a larger correlation of $\rho = 0.35$, the advantage demonstrated by BSE at the low SNR regime vanishes. BSE and FSS perform similarly except for small deviations in RTE observed at SNR= 0.42. All the methods converge to a similar RTE around SNR= 0.71, except for the lasso. The lasso selects a denser solution than all of the other methods, and does not converge to the RTE obtained by the other methods. The relaxed lasso does not have this problem as it manipulates a second tuning parameter γ to control the aggressiveness of the relaxed lasso to shift its performance from that of the lasso to that of best subset and forward selection. The results demonstrate that either BSE or the relaxed lasso are the best methods for problems of this size. The choice of which method to select depends on the correlation in the underlying data. The correlation of the features affects the critical transition value after which BSE, best subset, and FSS outperform the lasso and are competitive with the relaxed lasso method. BSE has a lower RTE than the relaxed lasso only in the case of $\rho = 0$.

In Fig. 5, we report results relating RTE to s. RTE is affected most by a change in the number of nonzeros in the model. Similar to the results of Hastie et al., models generated have a critical transition value at which point the RTE of BSE and FSS decreases below that of the lasso.

The performance of the lasso is worse than all of the other methods above the critical transition value, while that of the relaxed lasso is similar to that of the stepwise methods. Unlike in the $s = 5$ case, in all of the results, the relaxed lasso outperforms BSE for SNR less than the critical transition value. The most notable result from this study is that, in certain cases above the critical transition value, BSE and FSS outperform the relaxed lasso. For $s = 30$, BSE outperforms FSS and the relaxed lasso for SNR= 1.22. We also investigated whether the RTE converges for all methods if the SNR value is increased beyond six. At larger SNR values approaching 20, BSE still outperforms FSS and the relaxed lasso. From this comparison, it appears that, in the case of low correlation in the input data and regardless of how large the underlying model is, BSE is competitive with other methods at any SNR. The relaxed lasso and FSS also generate accurate models for problems of this size.

Depending on the problem structure, different subset selection strategies are optimal. We expected that BSE would outperform forward selection when the number of terms in the true model approaches n as suggested by the proposed approximation guarantee in Section “Algorithmic Analysis of BSE”. This trend was observed for $\rho = 0$. Overall, the best technique to use depends on the underlying data. For certain classes of problems, especially those that are uncorrelated, BSE produces an accurate and sparse model.

Conclusions

We investigated using backward stepwise elimination to solve the subset selection problem. Our main theoretical result is the proof of the existence of a bound on the accuracy of a solution selected by backward stepwise elimination related to how close the function is to being supermodular. Using the concept of the supermodularity ratio, we obtained an approximation guarantee for backward stepwise elimination. Our computational results demonstrate that the performance of backward stepwise elimination is dependent on the difference between n and k, and more unexpectedly, the supermodularity ratio. We developed a GPU parallel batched BSE algorithm. This algorithm reduces the execution time of solving the subset selection problem for matrices with 1000 rows and 600 columns by a factor of 5x.

We demonstrated that BSE performs as well as other state-of-the-art subset selection strategies that are commonly employed in practice. For certain problems at SNR below 0.5, BSE generated sparser models and achieved a lower relative test error than forward selection and the lasso. Results demonstrated that BSE also achieved a lower relative test error than the relaxed lasso, the lasso, or forward stepwise selection for problems with no correlation and for signal to noise ratios above zero.

Our primary conclusion is that BSE is a technique that should be considered by practitioners who want to develop sparse and accurate models.

Funding This work was conducted as part of the Institute for the Design of Advanced Energy Systems (IDAES) with funding from the Office of Fossil Energy, Cross-Cutting Research, U.S. Department of Energy. We also gratefully acknowledge the support of the NVIDIA Corporation with the donation of the NVIDIA Tesla K40 GPU used for this research.

Availability of Data All data used for computations in this paper can be obtained from https://github.com/bsauk/BackwardStepwiseElimination

Code Availability All codes used for computations in this paper can be obtained from https://github.com/bsauk/BackwardStepwiseElimination. This repository includes all scripts necessary to reproduce the results of this article.

Declarations

Conflict of Interest None.

References

1. Amaldi, E., Kann, V.: On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. Theoretical Computer Science 209, 237–260 (1998).
2. Anderson, E., Bai, Z., Bischof, C., Blackford, L.S., Demmel, J., Dongarra, J., Croz, J.D., Hammarling, S., Greenbaum, A., McKenney, A., Sorensen, D.: LAPACK Users’ Guide (Third ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1999).
3. Bendel, R., Afifi, A.: Comparison of stopping rules in forward “stepwise” regression. Journal of the American Statistical Association 72, 46–53 (1977).
4. Bertsimas, D., King, A., Mazumder, R.: Best subset selection via a modern optimization lens. The Annals of Statistics 44, 813–852 (2016).
5. Bertsimas, D., Pauphilet, J., Parys, B.V.: Sparse regression: Scalable algorithms and empirical performance. Statistical Science 35, 555–578 (2020).
6. Björck, A., Park, H., Eldén, L.: Accurate downdating of least squares solutions. SIAM Journal Matrix Analysis and Applications 15, 549–568 (1994).
7. Couvreur, C., Bresler, Y.: On the optimality of the backward greedy algorithm for the subset selection problem. SIAM Journal on Matrix Analysis and Applications 21, 797–808 (2000).
8. Cozad, A., Sahinidis, N.V., Miller, D.C.: Automatic learning of algebraic models for optimization. AIChE Journal 60, 2211–2227 (2014).
9. Das, A., Kempe, D.: Approximate submodularity and its applications: subset selection, sparse approximation and dictionary selection. The Journal of Machine Learning Research 19, 74–107 (2018).
10. Dong T, Haidar A, Luszczek P, Tomov S, Abdelfattah A, Dongarra J. MAGMA batched: a batched BLAS approach for small matrix factorizations and applications on GPUs. Tech. rep., Technical Report (2016).

11. Efroymson, M.: Multiple regression analysis. In A. Ralston and H.S. Wilf (eds.), Mathematical Methods for Digital Computers, Wiley, New York pp. 191–203 (1960).
12. Furnival, G., Wilson, R.: Regressions by leaps and bounds. Technometrics 16, 499–511 (1974).
13. Gatu, C., Kontoghiorghes, E.J.: Branch-and-bound algorithms for computing the best-subset regression models. Journal of Computational and Graphical Statistics 15, 139–156 (2006).
14. Gatu, C., Kontoghiorghes, E.J.: A fast algorithm for non-negativity model selection. Statistics and Computing 23, 403–411 (2013).
15. Haidar, A., Dong, T., Luszczek, P., Tomov, S., Dongarra, J.: Batched matrix computations on hardware accelerators based on GPUs. The International Journal of High Performance Computing Applications 29, 193–208 (2015).
16. Hastie, T., Tibshirani, R., Tibshirani, R.J.: Best subset, forward stepwise or lasso? Analysis and recommendations based on extensive comparisons. Statistical Science 35, 579–592 (2020).
17. ICL. MAGMA. 2020. http://icl.cs.utk.edu/projectsfiles/magma/doxygen/. Accessed Feb 2020.
18. Karaca O, Kamgarpour M. Exploiting weak supermodularity for coalition-proof mechanisms. In: Proceedings 2018 IEEE Conference on decision and control (CDC), IEEE, Miami Beach, FL, 2018; p. 1118–123.
19. Kohavi, R., John, G.: Wrappers for feature subset selection. Artificial intelligence 97, 273–324 (1997).
20. Liberty, E., Sviridenko, M.: Greedy minimization of weakly supermodular set functions. In: Jansen, K., Rolim, J., Williamson, D., Vempala, S. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017). Leibniz International Proceedings in Informatics (LIPIcs), pp. 19:1–19:11 (2017).
21. Mantel, N.: Why stepdown procedures in variable selection. Technometrics 12, 621–625 (1970).
22. Meinshausen, N.: Relaxed lasso. Computational Statistics & Data Analysis 52, 374–393 (2007).
23. Miller, A.: Subset selection in regression. CRC Press, Boca Roton (2002).
24. Nemhauser, G., Wolsey, L., Fisher, M.: An analysis of approximations for maximizing submodular set functions-I. Mathematical Programming 14, 265–294 (1978).
25. NVIDIA Corporation: cuBLAS. 2020. https://docs.nvidia.com/cuda/cublas/index.html. Accessed Feb 2020.
26. Ratner, B.: Variable selection methods in regression: Ignorable problem, outing notable solution. Journal of Targeting, Measurement and Analysis for Marketing 18, 65–75 (2010).
27. Sakaue S. Weak supermodularity assists submodularity-based approaches to non-convex constrained optimization. Arxiv preprint arXiv pp. 1–26 (2019).
28. Sarwar, O., Sauk, B., Sahinidis, N.V.: A discussion on practical considerations with sparse regression methodologies. Statistical Science 35, 593–601 (2020).
29. Sauerbrei, W., Holländer, N., Buchholz, A.: Investigation about a screening step in model selection. Statistics and Computing 18, 195–208 (2008).
30. Sauk, B., Ploskas, N., Sahinidis, N.V.: GPU paramter tuning for tall and skinny dense linear least squares problems. Optimization Methods and Software 35, 638–660 (2020).
31. Tibshirani, R.: Regression shrinkage and selection via the lasso. Royal Statistical Society 58, 267–288 (1996).
32. Tibshirani, R.: Degrees of freedom and model search. Statistica Sinica 25, 1265–1296 (2015).