Best Subset Selection via Mixed-Integer Optimization: Implementation and Application to House Price Data

Bowen Hua

December 16, 2017

Abstract

Recent advances in mixed-integer optimization (MIO) algorithms have enabled the near-exact solution to the best subset selection problem in regression modeling via mathematical optimization. In this report, we implement the MIO-based best subset selection algorithm. Through numerical experiments on a real-world house price dataset, we compare the best subset selection algorithm with lasso and backward stepwise selection.

1 Introduction

Sparsity arises in many statistical problems. The *subset selection problem* often appears in many statistical models dealing with sparsity. In this problem, we identify a subset of the predictors that we believe to be related to the response.

The *best subset selection* (BSS) method usually appears in a textbook as the first method of subset selection introduced . Given a response vector $y \in \mathbb{R}^n$, a predictor matrix $X \in \mathbb{R}^{n \times p}$, and a subset cardinality k, BSS finds a subset of k predictors that produces the best t in terms of a given metric. Considering a regression problem where we use squared error as our metric, BSS can be formulated as the following nonconvex problem:

$$\min_{\beta \in \mathbb{R}^p} ||y - X\beta||_2^2 \quad \text{s.t.} \quad ||\beta||_0 \leq k. \tag{1}$$

An exhaustive search algorithm is typically given in a textbook (e.g., [Hastie et al., 2008]), and then there goes a discussion that refers to this method as conceptually appealing but usually computationally formidable in practical cases, because of the combinatorial nature of the problem.

When I was studying this method, I thought that this problem can be formulated as a mixed-integer optimization (MIO) problem, since most of my research has to do with constrained optimization and operations research.[1] Further literature search finds a paper

[1] As the saying goes: if the only tool you have is a hammer, it is tempting to treat everything as if it were a nail.

that proposes the very same idea: [Bertsimas et al., 2016] formulate BSS as an MIO problem and show that the MIO-based algorithm outperforms lasso [Tibshirani, 1995]. The main reason why we can cast BSS as an MIO problem and solve it much faster than exhaustive search lies in the recent progress of the efficiency of MIO solvers.[2]

In this report, we describe the MIO formulation of the BSS problem proposed in [Bertsimas et al., 2016]. We present some results that shed light on why some constraints that are seemingly redundant are added in the MIO problem. Next, we focus on a real-world dataset about house prices. We present some exploratory data analysis on this dataset and compare the performance of MIO-based BSS, lasso, and backward stepwise selection on this dataset.

2 Formulation of BSS as an MIO Problem

As mentioned in [Bertsimas et al., 2016], the combined machine-independent speedup factor in MIO solvers between 1991 and 2013 is 580,000. This section formulates the BSS problem (1) as an MIO problem.

2.1 Formulation via Specially Ordered Sets

Using a standard trick in integer programming, we introduce a binary indicator vector $z \in \{0,1\}^p$. For predictor i, if $z_i = 0$, then we do not choose that predictor in our subset (i.e., $\beta_i = 0$). This logical relationship implies

$$(1 - z_i)\beta_i = 0 \quad \forall i,$$

which can be modeled via integer optimization using Specially Ordered Sets of Type 1. In an SOS-1 constraint, at most one variable in the set can take a nonzero value.

This leads to the following formulation:[3]

$$\min_{\beta,z} \quad ||y - X\beta||_2^2 \tag{2}$$

$$\text{s.t.} \quad \sum_{i=1}^{p} z_i \leq k \tag{3}$$

$$z_i \in \{0,1\}, \quad \forall i \tag{4}$$

$$(1 - z_i, \beta_i) : \text{SOS-1}, \quad \forall i. \tag{5}$$

It is easy to see that this formulation is equivalent to (1). An advantage of this formulation is that it does not require any specication of any parameter other than k. However, this formulation is not scalable, and [Bertsimas et al., 2016] continue to propose another equivalent formulation.

[2] As far as I know, the fastest solution method to the traveling salesman problem is also MIO-based.

[3] This is exactly the same as formulation (5) in [Bertsimas et al., 2016].

2.2 Formulation via the big-M method

Another equivalent MIO formulation the BSS problem (1) is the following:

$$\min_{\beta,z} \quad ||y - X\beta||_2^2 \tag{6}$$

$$\text{s.t.} \quad \sum_{i=1}^{p} z_i \leq k \tag{7}$$

$$-M_i z_i \leq \beta_i \leq M_i z_i, \quad \forall i \tag{8}$$

$$z_i \in \{0, 1\}, \quad \forall i \tag{9}$$

where M_i are large constants.

We first show that formulation (6)–(9) is equivalent to (1). When the indicator variable for feature i is zero, the big-M constraints (8) become:

$$0 \leq \beta_i \leq 0,$$

which force β_i to be equal to zero. When $z_i = 1$, constraints (8) become

$$-M_i \leq \beta_i \leq M_i,$$

a constraint that needs to be made redundant by choosing M_i to be large enough.

We next explain how to pick the constants M_i. Although we could let M_i be a very large number, this poses a problem for the MIO solver because: 1) a large number causes numerical issues, and 2) perhaps more importantly, a large M_i slows down the solution speed.

We should choose M_i to be as tight as possible (i.e., the tightest upper bound of β_i) because MIOs are typically solved by branch-and-bound-based methods. In these methods, an integer relaxation[4] of the MIO is solved to get a lower bound of the optimal objective function, and a feasible solution provides an upper bound of the optimal objective function. A large M_i leads to weak integer relaxation and loose lower bounds.

2.3 Comparison of Two Formulations

We next investigate why big-M formulation (6)–(9) is preferred over the SOS-based formulation (2)–(5).[5]

As mentioned in the previous subsection, the strength of the integer relaxation of MIO is an important factor that is impactful on solution speed. The integer relaxation of the SOS-based formulation (2)–(5) is very weak because SOS-1 constraints do not appear in the integer relaxation.[6] In fact, without additional valid inequalities added by the solver, the integer relaxation of formulation (2)–(5) is just the unregularized least squares problem.

[4] A relaxation where we ignore the integrality requirement.

[5] Such comparison is not mentioned in the original paper, perhaps because the authors view such comparison as trivial. Nevertheless, I found this helpful in understanding the formulations.

[6] See `http://www-01.ibm.com/support/docview.wss?uid=swg21400084` for a discussion from CPLEX support.

On the other hand, the big-M constraints do appear in the integer programming relaxation. The lower bounds obtained in a branch-and-bound algorithm are better, and the convergence is faster.

We can solve a convex optimization problem to determine optimal values for M_i:

$$\min_{\beta_i, M_i} \quad M_i \tag{10}$$
$$\text{s.t.} \quad M_i \geq \beta_i \tag{11}$$
$$M_i \geq -\beta_i \tag{12}$$
$$||y - X\beta||_2^2 \leq UB, \tag{13}$$

where UB is an upper bound to the optimal objective value of (1). This bound can come from a feasible solution to the best subset selection problem.

3 Numerical Results

3.1 Implementation

We perform exploratory data analysis using Python and Scikit-learn. For lasso and backward stepwise selection, we use the implementation of Scikit-learn.

For best subset selection, we implement the big-M formulation (6)–(9) using MATLAB, as well as an optimization modeling language YALMIP [Lofberg, 2004]. We model the problem using YALMIP, which then translates our formulation before sending it to a solver. We use GUROBI to solve the problem.

As we can see below, the big-M formulation (6)–(9) can be implemented with about ten lines of code.

```
function [beta, obj] = bestSubset(X,y, bigM, k)

        % dimensions
        M = size(X, 1);
        N = size(X, 2);

        % variables
        beta = sdpvar(N, 1);
        z = binvar(N-1, 1);

        % constraints
        constraints = -bigM(1:N-1).*z <= beta(1:N-1) <= bigM(1:N-1) .* z;
        constraints = [constraints, sum(z) <= k];
        constraints = [constraints, -bigM <= beta <= bigM];

        % objective function
        obj = norm(y-X*beta,2);
```

```
        % optimize!
        diagnostics = optimize(constraints,obj);
        beta = value(beta);
        obj = value(obj);
end
```

3.2 House Price Dataset

I first tried some artificial data where we have a sparse model with k known as a ground truth. Without much surprise, BSS outperforms the other two methods. The performance comparison on a real-world dataset is much more interesting.

We use Ames Housing dataset [DeCock, 2011] which was compiled for use in data science education. This dataset has also been made into a Kaggle competition.[7]

This dataset describes the sale of individual residential property in Ames, Iowa from 2006 to 2010. It contains 2930 observations and a large number of features (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values. The features include various area dimensions, the number of kitchens, bedrooms, and bathrooms, the neighborhood where the houses locates, etc. The label (output) is the sale price of the house.

This dataset is mostly clean. However, it contains missing data, incorrect values, and many categorical features. We preprocess the data in the following steps:

[7] https://www.kaggle.com/c/house-prices-advanced-regression-techniques

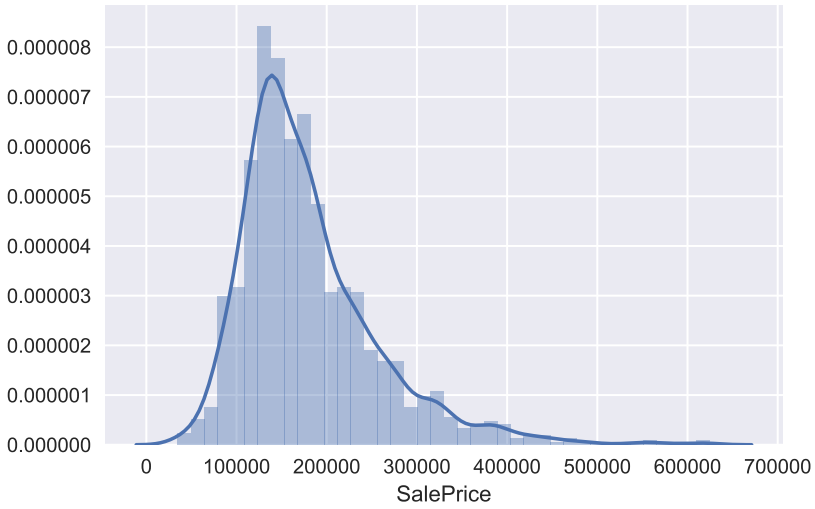

Figure 1: The distribution of sale prices is skewed. Therefore, we perform a log transformation.

- Fill in missing values.
- Delete outliers.
- Perform log transformation for skewed features, as well as the sale price (which is also skewed as seen from Fig. 1).
- Drop trivially irrelevant/highly correlated features.
- Standardize the features.
- Create dummy variables.

After preprocessing, we have 124 features.

3.3 Subset Selection with $k = 20$

As our first numerical experiment, we fix the cardinality of the selected subset, k, and compare the performance of the three models (BSS, Lasso, Backward Stepwise Selection). We pick a small number 20 for ease of comparison, although 20 is much less than the optimal value of k. The number k is a hyperparameter for BSS and backward stepwise selection, but not Lasso. Therefore, we manually pick the lasso penalty λ through trial and error, so that the effective k in the optimal solution to lasso equals 20. This might not be a fair comparison, but it facilitates our comparison. In the next subsection, we compare the performance of the three methods where we use a validation set to tune the hyper parameters.

In this experiment, we follow the training-testing allocation of the Kaggle competition; we have 1456 training samples and 1459 testing samples. The computational time of both lasso and backward subset selection are within a second. Interestingly, we spend 24 hours solving BSS, only to a 3.28% sub-optimal solution. That is, we get a feasible solution and a lower bound of the optimal loss, and the loss of the feasible solution is 3.28% away from the lower bound.[8]

Even for the small dataset that we have, the amount of time needed to solve BSS, with state-of-the-art solver such as GUROBI, is still formidable. We also note that the dataset we use is about as large as the largest dataset tested in the original paper [Bertsimas et al., 2016].

Fig. 2 shows the variable selection results. We see that, out of the 124 features, 36 features are chosen by at least one of the three methods. Among the 36 features, several are chosen by all three methods. These include:

- 3SsnPorch: three season porch area in square feet.
- GarageQual: quality of garage.
- GarageArea: area of garage.
- GrLivArea: above grade (ground) living area.
- HeatingQC: heating quality and condition.
- BsmtFinSF1: Type 1 finished square feet.
- OverallQual: overall quality.

[8]Typically, the feasible solution is closer to the true optimal solution (which we do not know) than 3.28% because it is usually hard to find a good lower bound in combinatorial optimization problems.

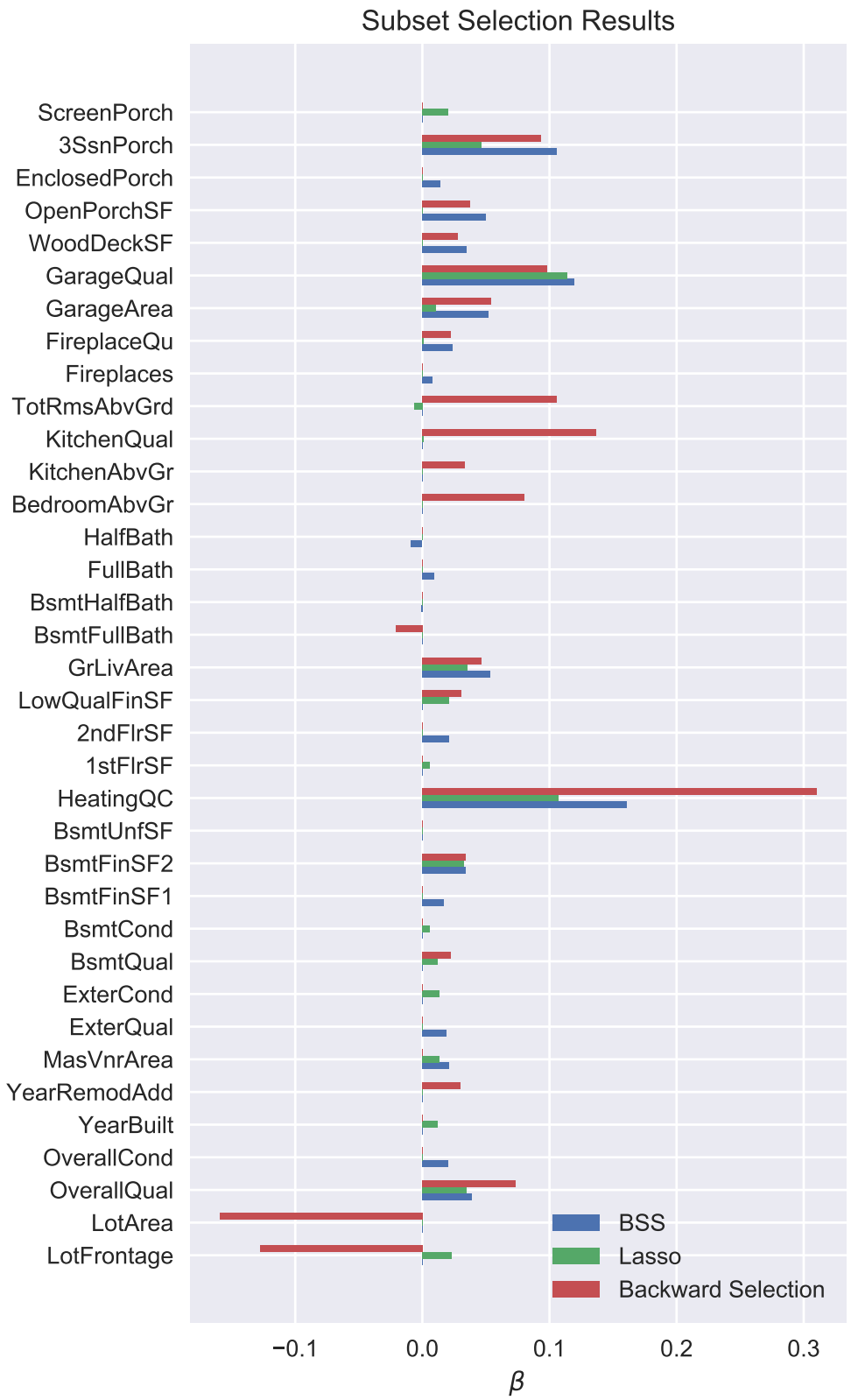

Figure 2: Subset selection via three methods.

Figure 3: Testing error of the three methods with $k = 20$.

In general, the subsets selected by BSS and lasso are somewhat similar, whereas the backward stepwise selection algorithm picks several features that none of the two other methods picks. In addition, backward stepwise selection assigns negative weights to BsmtFullBath (binary variable for full bathroom in basement), LotArea, and LotFrontage (linear feet of street connected to property), which seems strange.

Some of the differences between BSS and lasso can be explained by the correlation among features: lasso selects 1stFlrSF (area of first floor) whereas BSS selects 2ndFlrSF; lasso selects ExterCond (exterior condition), whereas BSS selects ExterQual (exterior quality).

Finally, Fig. 3 shows the testing performance of the three methods. We use root mean squared logarithmic error as our metric, since we performed log transformation on the output label. We can see that with fixed k, BSS outperforms the other two methods. In particular, BSS outperforms lasso. This might be explained by the fact that lasso leads to biased regression coefficient estimates, since the ℓ_1 norm penalizes the large coefficients more severely than the smaller coefficients.

3.4 Tuned Models

In this section, we use a validation set to tune the hyperparameters (k for BSS, backward stepwise selection, and λ for lasso), and compare the performance of the three tuned methods. Because the formidable computational time of BSS, we use a single validation set to tune the hyperparameters, instead of cross validating. We randomly select 1749 training samples, 583 validation samples, and 583 testing samples.

BSS. We choose k from the set $\{20, 40, 60, 80, 100\}$ due to the computational burden. For each choice of k, we set a time limit of 24 hours for solving the mixed-integer optimization problems. As a result, the BSS problems are solved to around 3% optimality gap. The validation process chooses $k = 80$.

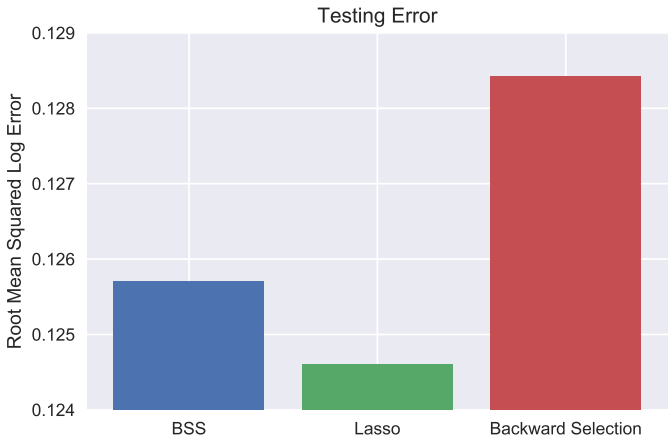

Figure 4: Testing error of the three tuned methods.

Lasso. We choose λ from the set $\{0.00005, 0.0001, 0.0003, 0.0005, 0.001, 0.01, 0.1\}$, and the validation process chooses $\lambda = 0.001$. In the optimal solution with $\lambda = 0.001$, there are 87 nonzero coefficients β_i.

Backward stepwise selection. We choose k from the set $\{20, 40, 60, 80, 100\}$. The validation process chooses $k = 60$.

Fig. 4 shows the testing performance of the three methods. It is interesting to see that, even after 24 hours of computation, BSS is outperformed by lasso on this dataset. This can partially explained by: 1) BSS is not solved to true optimum; 2) as pointed out by [Hastie et al., 2017], when the signal-to-noise ratio is low, and also depending on other factors like the correlations between predictor variables, BSS have a different bias-variance tradeoff characteristics; BSS does not dominate lasso, nor does lasso dominate BSS.

4 Discussion

In this report, we have implemented the best subset selection (BSS) method via mixed-integer optimization, originally proposed by [Bertsimas et al., 2016]. After analyzing two possible optimization formulations, we apply this method to a house price dataset, and compare its performance with lasso and backward stepwise selection.

We found that, for the dataset we use, although BSS consistently outperforms backward stepwise selection, it does not dominate lasso. In addition, the amount of computation required by BSS is still formidable, even given state-of-the-art solver and a moderately sized dataset. We also note the difficulty in fully parallelizing a mixed-integer optimization algorithm, which makes BSS via MIO even less scalable.

References

[Bertsimas et al., 2016] Bertsimas, D., King, A., and Mazumder, R. (2016). Best subset selection via a modern optimization lens. *Annals of Statistics*, 44(2):813–852.

[DeCock, 2011] DeCock, D. (2011). Ames , Iowa : Alternative to the Boston Housing Data as an End of Semester Regression Project. *Journal of Statistics Education*, 19(3):1–15.

[Hastie et al., 2008] Hastie, T., Tibshirani, R., and Friedman, J. (2008). *Elements of Statistical Learning*.

[Hastie et al., 2017] Hastie, T., Tibshirani, R., and Tibshirani, R. J. (2017). Extended Comparisons of Best Subset Selection, Forward Stepwise Selection, and the Lasso.

[Lofberg, 2004] Lofberg, J. (2004). YALMIP : a toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation*, pages 284–289.

[Tibshirani, 1995] Tibshirani, R. (1995). Regression shrinkage and selection via LASSO. *Journal of the Royal Statistical Society, Series B*, 73:273–282.