RM 294 – Optimization I

Project 3 – Mixed-integer quadratic programming

Nittala Venkata Sai Aditya (vn5227), Muskaan Singhania (ms88283), Varun Kausika (vsk394), Yingjia Shang (ys23737)

Table of Contents

I. INTRODUCTION

One of the predominant problems in the predictive analytics space is feature selection. we have to pick the subset k features from a total of m features. The total number of models that can be picked is:

$$\binom{m}{0} + \binom{m}{1} + \ldots + \binom{m}{m} = 2^m$$

If features are carefully selected, it could help improve the model performance on unseen data substantially resulting in more accurate predictions. One could say that we're grappling with an optimization problem, and this problem is more often than not tackled using advanced methods such as Lasso and Ridge Regression. Reiterating, variable selection helps extract the useful features that explain the variance in our target while keeping the model complexity to a minimum.

Direct variable selection using optimization has long been dismissed by the statistics/analytics community because of computational difficulties. This computational issue was part of the motivation for the development of LASSO and Ridge regression. The LASSO regression model selects a subset of features to be included in the model to minimize prediction error. By shrinking the insignificant variables' coefficients to closer to 0, lasso regression can achieve variance reduction, thus preventing overfitting.

However, with the advancement of optimization software recently, we have the ability to solve mixed integer quadratic programs (MIQP) for variable selection. In this project, we're setting out to compare the Direct variable selection method to Lasso. We want to validate if the Direct variable selection method has substantially evolved over time with better solvers. Our primary goal of the project is to evaluate whether we should shift from the current method of indirect variable selection using Lasso to direct variable selection using MIQP.

II. METHODOLOGY

We try to solve the problem with two different approaches and compare the outcomes. The first one is via Gurobi where we will use the Mixed Integer Quadratic Problem, and the second approach is via LASSO regression.

A. Direct Variable Selection – MIQP

We first applied Mixed Integer Quadratic Problem using direct variable selection. There are a few steps we need to consider while formulating the MIQP problem:

Objective Function

Given the training dataset of m independent variables X and a dependent variable y, our objective is to minimize the loss function

$$min_{\beta} \Sigma_{i=1}^{n} (\beta_0 + \beta_1 x_{i1} + \ldots + \beta_m x_{im} - y_i)^2$$

where m is the number of variables to be selected.

In order to pose this question in the standard framework of a quadratic programming objective, we need to transform our matrices. First, let β be an (m+1) x 1 column vector that constraints the intercept and all coefficients. Let X be the n x (m+1) matrix with the first column made up entirely 1s, and columns 2 to (m+1) have the data from the m independent variables. Let y be the n x 1 column vector that has the dependent variable data. Applying some matrices calculations from linear algebra, we can pose the objective function as

$$\min_{\beta} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_m x_{im} - y_i)^2 + \lambda \sum_{j=1}^{m} |\beta_j|,$$

Decision variables

We would include the following decision variables in our optimization problems:

The *m* coefficients of the variables:

$$\beta_0, \beta_1, \beta_2, \ldots, \beta_m$$

The binary *zj* variables which decide which βj variables are 0

$$z_j \text{ are binary where } j \in \{1, 2, \ldots m\}$$

The size of our decision variables should be (2m+1) x 1. Our decision variables are made up of m+1 values of β and the m values of z. Since the objective function written above has only m+1 values of β, we would need to format the Q matrix to be a (2m+1) x (2m+1) matrix with the upper left corner of the matrix being X^TX, and all other values as zeros. We will also have a linear term of the objective to be a (2m+1) x 1 vector with the first (m+1) terms as $-2y^TX$ and the rest as zeros.

Constraints

1. **First *m* set of Big-M constraints:** in order to implement non-negative constraint for all β coefficients, we are setting the lower bound to -M.

$$-M z_j \leq \beta_j \text{ for } j \in \{1, 2, 3, \ldots, m\}$$

2. **Second *m* set of Big-M constraints:** since we included the binary variables, zj, we are forcing the corresponding β to be zero if zj is zero.

$$M z_j \geq \beta_j \; for \; j \in \{1, 2, 3, \ldots, m\}$$

We choose the value of *M* to arbitrarily be 100 (sufficiently large value).

3. **k Constraints:** the total number of variables picked should be less than or equal to k. The main objective while training this model is to select the optimal value of k which gives the lowest error. We will select the value of k by devising our own cross-validation function.

$$\Sigma_{j=1}^{m} z_j \leq k$$

10 – Fold Cross Validation to Find Optimal k

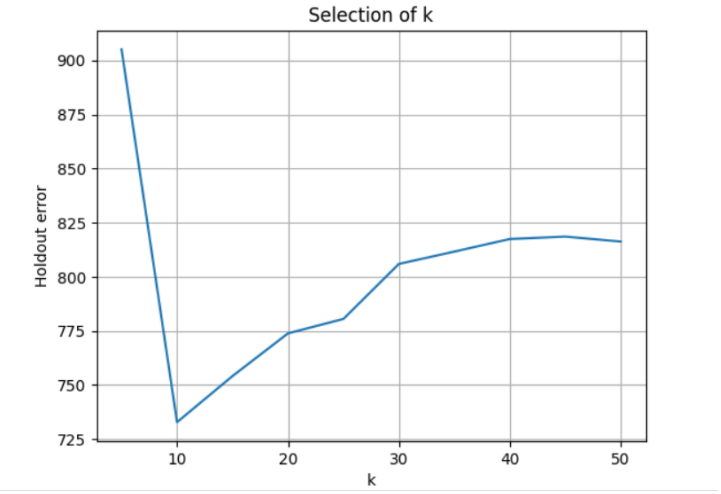
The main advantage of cross-validation is that we use the same dataset for getting the best fit model by plotting 9 parts into a training set and keeping the last one as a validation dataset. This ensures that the model is not overfitted and is able to cater to all data points and insights. Once all parts are modeled and tested, the average of the errors between the prediction obtained from the one validation set and the actual value is calculated. The aim is to select the value of K (number of variables selected) which gives the lowest error. We have 50 predictor variables in our dataset. We will start our k value from 5 to 50 with an increment of 5 i.e., k = 5,10,15,20,...,50.

We used the np.split() function on a sample created from a training dataset and split it into 10 folds. The loss metric used was the Sum of Squared Errors. The aim was to reduce this by the formula

$$SSE_{train} = (X_{train}\beta - y)^T * (X_{train}\beta - y)$$
$$SSE_{test} = (X_{test}\beta - y)^T * (X_{test}\beta - y)$$

We plotted the SSE against various values of k as seen below and found that the best number of variables for the MIQP model came out to be 10, the other variables are noise and contribute to overfitting.

	k	Holdout_error
0	5	905.047754
1	10	732.848361
2	15	754.143193
3	20	773.865825
4	25	780.541226
5	30	805.951129
6	35	811.646425
7	40	817.453951
8	45	818.574317
9	50	816.277840



Fit Model on Entire Training Set

Once we get the optimal number of variables (k = 10), we use that to fit the MIQP model and get the coefficients of betas:

```
In [11]:
1 bet_miqp = MIQP(train, 10)
2 print('Beta coefs:')
3 bet_miqp

Beta coefs:

Out[11]: array([ 0.97252408,  0.        ,  0.        ,  0.        ,  0.        ,
                 0.        ,  0.        ,  0.        ,  0.        , -2.30820726,
                 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
                -0.51832612, -0.20416201,  0.        ,  0.        ,  0.        ,
                 0.        ,  0.        ,  0.        , -1.55914318,  0.86697336,
                 0.        , -1.31191942,  0.        ,  0.        ,  0.        ,
                 0.        ,  0.        ,  0.        ,  0.        ,  0.4081653 ,
                 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
                 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
                 1.78147489,  0.        ,  0.88738292, -0.28229213,  0.        ,
                 0.        ])
```

Then, we used the result beta coefficients to predict the y variable in the test set. Our resulting MSE was 2.33.

```
The values of y predicted from MIQP problem are:

[ 6.17985878  5.09524299  3.28559532  3.75848539 -0.33297526 -5.14273683
 -3.14454357 -1.23806288  1.38511093 -0.44173854 -1.69500225  2.73035027
  0.74744903 -0.97192232 -0.68681528  8.04522381 -7.94698471  3.89063974
 -4.58142919 -3.21992082 -2.16211454  3.21686318 -3.19810533  0.19740731
 -2.35988844 -0.41999885 -1.9125216  -3.32418587 -3.14170972 -3.55379324
 -1.80842543 -0.37134301  1.8670808   5.04927886 -1.80005614  3.09427675
  4.38154309  2.6988627   1.6132886   5.97584637 -1.1973583   5.2232542
 -5.84899891 -1.14461528  4.51802998  4.18774866  4.12046008  0.61483809
  1.95723246 -1.54904383]
```

B. Indirect Variable Selection – LASSO

The second model we will try is by doing Lasso Regularisation. The main purpose of the Lasso Regularisation is to penalize high overfitting. Lasso shrinks the coefficient estimates towards zero and it has the effect of setting variables exactly equal to zero. When the lambda is small, the result is essentially the least squares estimate. As lambda increases, shrinkage occurs so that variables that are at zero can be thrown away.

In Lasso, the variables are selected as

$$y_{predicted} = min_{\beta}(\Sigma_{i=1}^{n}(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_m x_{im} - y_i)^2 + \lambda \Sigma_{j=1}^{m} |\beta_j|)$$

Here, variables are set to 0 when lambda is large, penalizing the cost function. Note that β0 is not included in the penalty term. When lambda is very large, the prediction of the regression is just the mean of the target variables.

Similar to the MIQP problem, we need to tune our hyperparameter (in this case lambda) to give the best possible fit which will reduce the SSE.

We used the LassoCV function in python which has a built-in cross-validation function that performs the same task as in the MIQP problem and helps select the best lambda.
On training the model, we found the best parameters as below:

```
In [12]:  1 # Lasso
          2 bet_lasso = LASSO(train, test)
```

```
Best lambda from LASSO: 0.0057453437864455085
Intercept:  1.0061963544057866
Number of X Variables: 18
Beta coefs:  [-0.          -0.           0.           0.          -0.           0.
 -0.          -0.          -2.11561506  0.          -0.06043079 -0.
 -0.          -0.          -0.41674549 -0.18155256  0.           0.
 -0.           0.           0.          -0.19710223 -1.3655275   0.73510021
 -0.          -1.30018578  0.           0.           0.06390289  0.
 -0.           0.          -0.10737966  0.25392747  0.02138366  0.
  0.           0.          -0.21159473  0.          -0.          0.
  0.           0.01152326  1.53171531 -0.01408773  0.6504778  -0.09757869
  0.           0.        ]
Train MSE:  2.4009897950790076
Test MSE:  2.3597086675768337
```

The optimal number of variables with non-zero coefficients in Lasso is 18, and the lambda that came out was 0.0057. Additionally, our resulting test MSE was 2.3597. We also used the beta coefficients obtained from the lasso regression to predict the y variable in the test set:

```
The values of y predicted from LASSO are:

[ 6.03825222  4.8649441   3.18119483  3.53168053 -0.45811526 -4.85193624
 -2.79173278 -1.44600095  1.39579841 -0.17881191 -1.95903628  2.70532632
  0.62547897 -0.45474384 -0.24671651  7.26275263 -7.4420549   3.60548489
 -4.34845018 -2.95351631 -1.89112059  3.33269462 -2.34452757  1.18116807
 -2.45191722 -0.21885517 -1.49545244 -2.22345125 -3.47518537 -3.64118586
 -1.64685854  0.0656198   1.41989248  5.49729599 -1.16134637  2.32078083
  4.7252849   3.20434951  1.56532897  5.83470219 -0.81974158  4.62664564
 -5.64573204 -1.15917996  4.17943946  4.09115511  3.82923833  0.6400209
  1.86558547 -0.97272978]
```

III. Comparing Lasso and MIQP Models

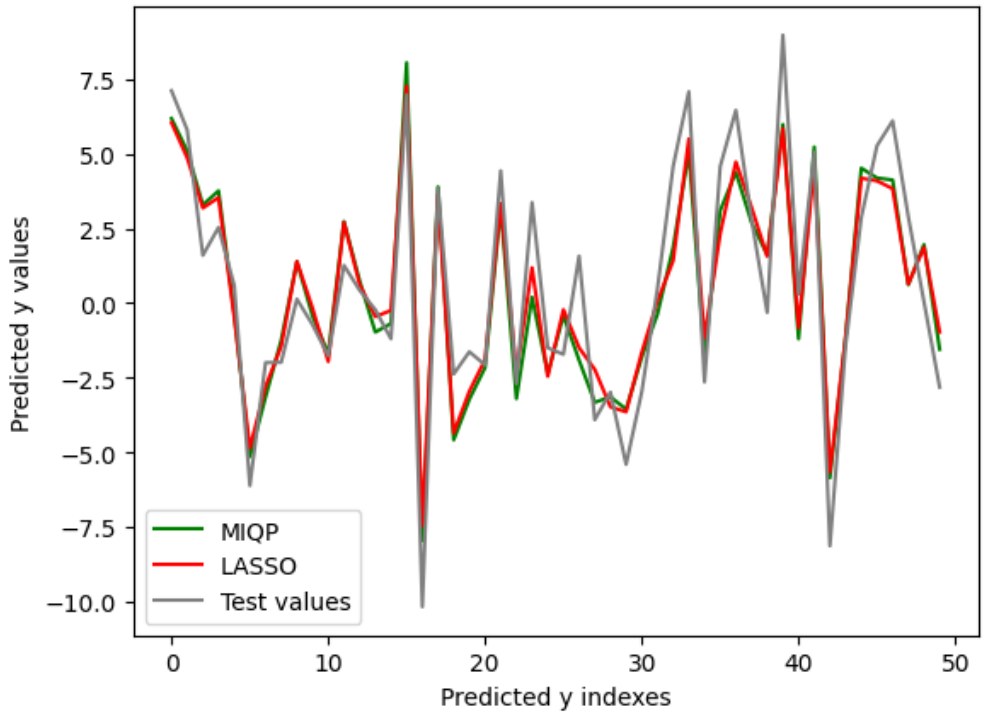
For the MIQP model, we have selected k=10 as our best model, and for the lasso model, we eventually resulted in 18 non-zero coefficients. On comparing the two methods, we found that the predicted values for both models are very similar with some minor differences. We would compare the model prediction accuracy, selected variable coefficients, and model run-time efficiency.

Model Accuracy

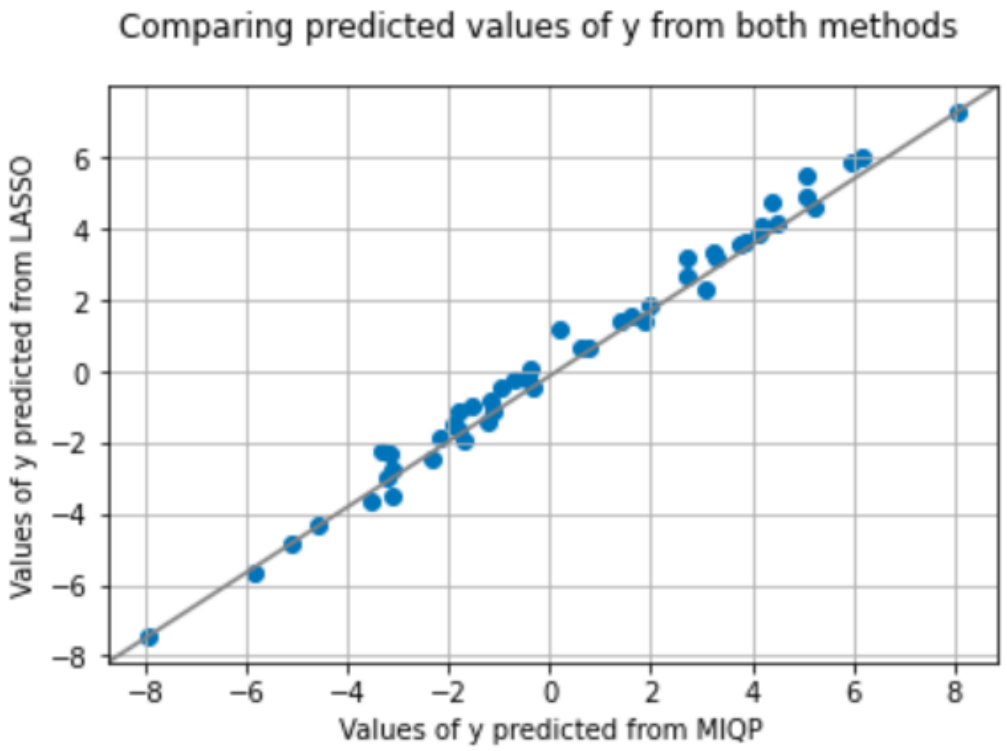
First, we are comparing the predicted value of dependent variable y using the LASSO and MIQP models. Our obtained MSE for the two models is 2.33 for the MIQP model and 2.359 for the Lasso model. The test MSE values were really similar for the two models. The predicted values of y for each model are listed below.

```
The values of y predicted from MIQP problem are:

[ 6.17985878  5.09524299  3.28559532  3.75848539 -0.33297526 -5.14273683
 -3.14454357 -1.23806288  1.38511093 -0.44173854 -1.69500225  2.73035027
  0.74744903 -0.97192232 -0.68681528  8.04522381 -7.94698471  3.89063974
 -4.58142919 -3.21992082 -2.16211454  3.21686318 -3.19810533  0.19740731
 -2.35988844 -0.41999885 -1.9125216  -3.32418587 -3.14170972 -3.55379324
 -1.80842543 -0.37134301  1.8670808   5.04927886 -1.80005614  3.09427675
  4.38154309  2.6988627   1.6132886   5.97584637 -1.1973583   5.2232542
 -5.84899891 -1.14461528  4.51802998  4.18774866  4.12046008  0.61483809
  1.95723246 -1.54904383]
The values of y predicted from LASSO are:

[ 6.03825222  4.8649441   3.18119483  3.53168053 -0.45811526 -4.85193624
 -2.79173278 -1.44600095  1.39579841 -0.17881191 -1.95903628  2.70532632
  0.62547897 -0.45474384 -0.24671651  7.26275263 -7.4420549   3.60548489
 -4.34845018 -2.95351631 -1.89112059  3.33269462 -2.34452757  1.18116807
 -2.45191722 -0.21885517 -1.49545244 -2.22345125 -3.47518537 -3.64118586
 -1.64685854  0.0656198   1.41989248  5.49729599 -1.16134637  2.32078083
  4.7252849   3.20434951  1.56532897  5.83470219 -0.81974158  4.62664564
 -5.64573204 -1.15917996  4.17943946  4.09115511  3.82923833  0.6400209
  1.86558547 -0.97272978]
```

To visualize the model comparison, we have plotted the predicted values of each model versus the actual y values on the same graph. The graph is presented below.

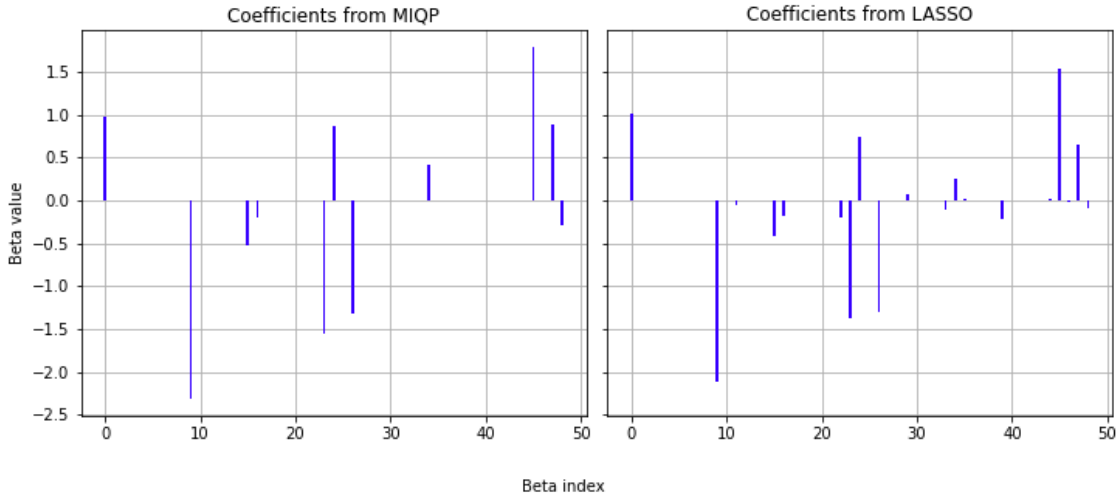


From the graph comparing the model prediction, we can observe that the predicted values of the two models are very similar. We further plotted a scatter plot between the predicted values of both methods as shown below:



As can be seen from the graph, there is not much difference between the two in terms of predictions. The points plotted from the same, methods are very close to the diagonal.

Variable Coefficients



On comparing the beta values that were obtained from both methods, we find that the variable selections for the two methods are very different. While the MIQP method selected 10 variables as the optimal answer, the Lasso model selected 18 variables and shrank the coefficients of all other variables to zero. Additionally, we found out that some variable coefficients in the Lasso regression model are relatively small as compared to the variable coefficients using the MIQP method. There are many coefficients that are almost zero in the lasso method.

Run-Time Efficiency

Comparing the model run-time for the MIQP model and the Lasso model, the run time for the prior model is significantly longer than the latter model. Using a dataset with 50 features and 250 records, it took approximately 1.5 hours for the model to run, whereas the Lasso model finished training within seconds. As we increase the size of the dataset or the number of cross-validation, we can reasonably predict that the run-time for the MIQP model will increase. This would be one shortcoming that the MIQP model is suffering.

IV. Recommendations

In terms of subset selection, we find that solving the MIQP is more accurate and exactly zeros out the required variables, whereas if only prediction is necessary, there is not much use to solving the harder problem (MIQP) and would be more advantageous to select the computationally cheaper LASSO. It would be worthwhile to note that the Lasso model isn't scaled invariant whereas MIQP is. Hence it would make sense to use MIQP over Lasso if the dataset isn't very big or computation time is not a constraint for more accurate results.

Additionally, in most real-world scenarios collinearity among variables while running regression seems to be a major concern as collinearity violates linear regression assumptions. It turns out Lasso does not have an elegant method of dealing with this issue, as it eliminates the variable at random, whereas MIQP seems not to suffer from collinearity.

In conclusion, if computation time is not a major concern, the MIQP method has gotten a lot faster since it was introduced. Thus, we would recommend using MIQP for better accuracy due to the aforementioned reasons. However, if the ask is for faster results with decent accuracy, we would recommend relying on the Lasso method.