Optimization – I
Project 3: Non-Linear Programming
Group: 23

Team: Palak Agarwal (pa9797), Mahika Bansal (mb62835), Brandt Green (bwg537), Chandler Wann (clw4642)

Introduction

High dimensional data analysis has always been a huge challenge in the fields of data mining and machine learning. Feature selection thus becomes important to eliminate redundancy and irrelevant data. It also enables faster training, reduces complexity, and improves accuracy if the right subset is chosen.

Overview

One of the most common problems in predictive analytics is variable selection for regression. 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. However, in the recent past there have been tremendous advancements in optimization software, specifically the ability to solve mixed integer quadratic programs (MIQP). This project will pose the variable selection problem for regression as an MIQP which has been solved using gurobi. The results have been further compared with results from LASSO to see if the additional 'shrinkage' component of LASSO really is more beneficial than finding the 'best' set of variables to include in your regression.

Techniques & Algorithms

MIQP

MIQP or Mixed Integer Quadratic Programming is one of the most important class of mixed integer nonlinear programming that consists of: discrete decision variables and nonlinear objective function and thus subsumes both mixed integer linear programming and quadratic programming.

For feature selection, we use MIQP with the goal of choosing the appropriate values for all of the βs with the goal of minimizing the test set error. The optimization problem to solve is formulated as follows:

Objective:

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

Constraints:

$$s.t. -Mz_j \leq \beta_j \leq Mz_j \quad for\ j = 1, 2, 3, \ldots, m$$
$$\sum_{j=1}^{m} z_j \leq k$$
$$z_j \ are\ binary.$$

Where X refers to the set of m independent variables and Y the dependent variable. We also included some binary variables in the problem, a Z vector, to facilitate feature selection. The variable *k* is an input that allows us to specify the maximum number of features selected, where we choose *k* to be (*k* = {5,10,15….50}).

Code:

Data snippet:

	y	X0	X1	X2	X3	X4	X5	X6	X7	X8	...	X41	X42	X43	X44	X45
0	8.536145	1.0	-1.535413	0.718888	-2.099149	-0.442842	-0.598978	-1.642574	0.207755	0.760642	...	0.361866	1.793098	-0.631287	-0.061751	0.511049
1	4.808344	1.0	-1.734609	0.551981	-2.147673	-1.552944	1.514910	-1.143972	0.737594	1.321243	...	-0.677985	-0.165679	0.065405	0.137162	1.258197
2	-1.530427	1.0	0.097257	0.107634	-0.194222	0.335454	-0.408199	0.133265	0.706179	0.394971	...	1.108801	0.333791	0.282055	-1.086294	-0.115354
3	-0.428243	1.0	-0.067702	0.557836	0.700848	-1.121376	1.722274	0.613525	0.700909	-0.417976	...	0.692511	-0.350990	0.624558	0.434520	-0.367409
4	0.566694	1.0	0.488729	0.211483	0.568389	0.646837	0.163868	-0.002152	0.125137	0.493571	...	-0.000605	1.075280	0.182281	-1.138458	0.106092

5 rows × 52 columns

Note: Complete data set consisted of 250 rows. Also note this snippet includes a column for the independent y variable, and a column of all 1s to represent the intercept term.

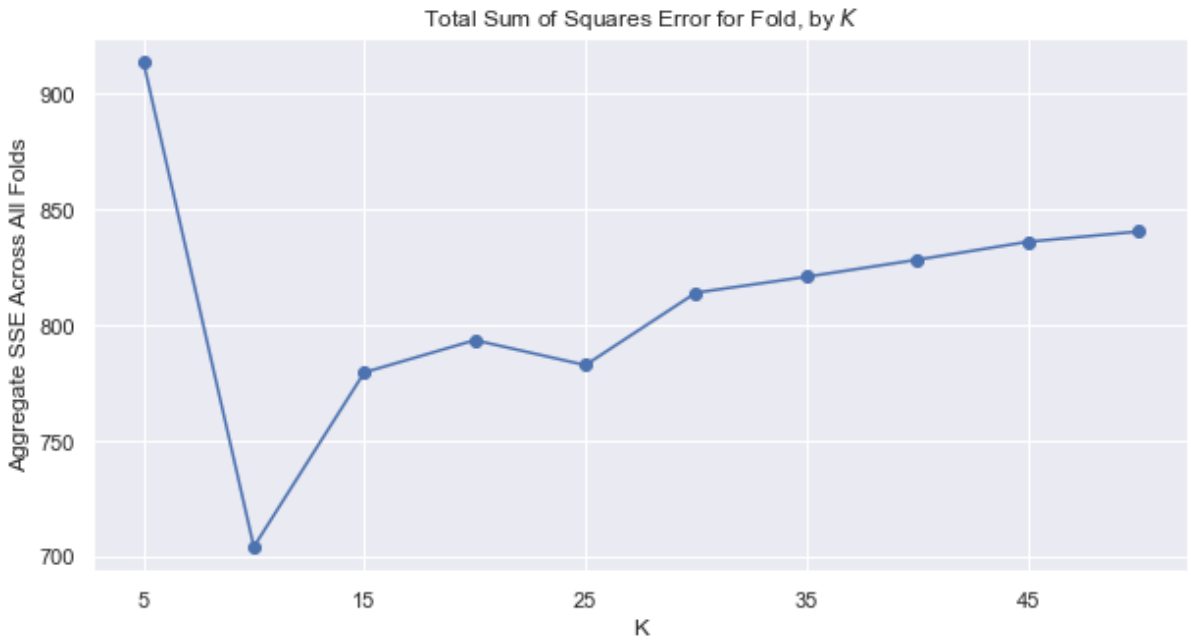
In order to find the optimal k to use in the optimization, we use K-fold cross validation where K = 10 (Note, this is not the same *k* referenced in the optimization problem formulation). This involves first shuffling our training data set and then splitting it into 10 distinct subsets. Following that segmentation, we iterate through each subset where choose one of the folds to be testing set and the other 9 are our training data. Once we have this training/testing data segmented, we iterate through all of our possible values for *k* and solve the MIQP problem with each different value of k. The output of this Gurobi optimization will yield us an intercept coefficient and *k* non-zero betas. We use these betas and the test set to predict our out of sample sum of squared errors (SSE) which we store in a pandas data frame. The final result of these successive iterations is displayed below:

	5	10	15	20	25	30	35	40	45	50
0	121.08483	83.22601	84.98515	82.43595	87.58834	95.66110	97.65884	97.40345	96.66076	97.57031
1	76.27096	68.72295	85.44507	91.70347	87.32364	95.86751	96.42208	96.91586	100.55214	101.13772
2	76.40846	62.50184	70.91066	83.65107	75.81232	73.46238	78.72707	80.83133	79.28233	79.96562
3	72.64028	62.72359	71.70770	71.71293	70.69936	79.46967	80.39510	77.86652	77.20627	78.33711
4	118.95958	82.00438	83.84794	80.40048	80.64813	86.74073	90.89855	87.42634	85.88534	86.93229
5	82.97925	61.88108	65.54801	59.53692	75.45312	67.36771	61.14406	66.13959	67.73551	67.36372
6	76.18434	65.08857	69.81931	70.62222	68.35383	70.47346	67.89549	71.89055	75.20079	75.06871
7	60.29282	48.66334	59.89032	58.02635	56.43211	55.66685	54.96695	60.79733	64.49536	63.55508
8	133.75785	93.39369	99.85986	105.95280	98.82562	105.08918	104.69604	103.96380	104.83067	104.08194
9	95.53616	75.76057	87.45220	89.38065	81.48977	84.18368	88.06178	84.99322	84.14878	86.44153

Where each row represents the values corresponding to when the Kth fold was the test set and each column represents a value of *k* corresponding to the maximum number of features that can be included.

Results:

To choose the best value of *k,* we sum the SSE number across all folds for each *k* and pick the *k* with the smallest aggregate error.



After running the code, we found that *k* = 10 is the optimal value, i.e., 10 variables were chosen as the best subset.

We use this value of *k* to re-run the optimization of the entire training data set, and then use this model to predict on the never-before-seen holdout set.

LASSO

LASSO or Least Absolute Shrinkage & Selection Operator uses shrinkage parameter for variable selection. The model encourages simple, sparse models and thus is suitable for models with high multicollinearity or automation of selection of best subset of the variables.

Lasso follows L1 regularization, which adds a penalty equivalent to absolute value of the magnitude of coefficients multiplied by shrinkage parameter, λ, i.e.

Objective:

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

Code:

The beauty of using scikit-learn is that they abstract away a ton of the complexity associated with the cross validation. We are able to carry out a similar cross validation process to the Gurobi optimization with just the following lines of code!

```
lasso_model_cv = linear_model.LassoCV(cv=10).fit(X_train_og,y_train_og)
best_lambda = lasso_model_cv.alpha_
print(f'The best lambda = {best_lambda}')
```

```
The best lambda = 0.07638765995113514
```

You can see the best regularization parameter to use when fitting our complete training data set is about .0764. We use this value to fit a lasso model on the entire training set, and then use this model to predict on the never-before-seen holdout set.

Results:

With LASSO, we were able to get the best subset with 17 variables.

Comparing the results from MIQP & LASSO

	SSE	MSE	R_Squared
Gurobi_Method_Metrics	116.82720	2.33654	0.85867
Lasso_Metrics	117.48174	2.34963	0.85788

We see that MIQP performs slightly better than LASSO with a less complex model.

Advantages and Limitations

Though we see that MIQP performs slightly better than LASSO here, both the techniques have some advantages and limitations over each other which need to be taken into account while choosing the right model depending on the situation.

Benefits & Disadvantages of MIQP:

MIQP helped in getting better test set accuracy, but it requires a lot of computational power and time, which can be a problem with bigger datasets. It requires manual setup and doesn't have established and sophisticated library with built-in functions like scikit-learn for other machine learning algorithms. It also might not return the global optima, given the complexity of the problem.

Benefits & Disadvantages of LASSO:

LASSO helps in automation variable selection, by shrinking coefficients to zero and helps in avoiding overfitting. But the selected features might be highly biased and the selection of only one feature from a set of high multicollinearity bunch is highly arbitrary. The prediction performance might be worse than other regularization techniques like ridge, and doesn't work well with high dimensionality data ($n << p$)

Feature selection, though helps in reducing complexity, can lead to loss of information if not used properly. The techniques rely on approximation, which might not lead to the best selection at times. They also might lead to only one solution, where in reality multiple solutions might be possible. This is true especially in the cases with inherent redundancy in the underlying problem. In such cases, other methods like dimensionality reduction can also be used depending on the situation.

Final Recommendation & Conclusion

Our final recommendation on which methodology to utilize is of course "it depends". The results are quite similar, with only a small accuracy advantage going to the MIQP approach. Because of the similarity, we suggest continuing to use LASSO for the majority of tasks, because of its simplicity, easier implementation, and community familiarity with the following exceptions:

- Incremental accuracy improvements are highly valuable.
- Computational time and additional complexity are not a constraint.
- The number of potential features to select from is relatively small.