LASSO Regression as a Quadratic Program

MATH 5593 - Linear Programming

Kevin Ruiz Brady Lamson

November 29, 2025

College of Letters Arts and Sciences

Denver

Outline

1. Introduction
2. Background information
3. Implementation
4. Results and Conclusion

Introduction

Project Motivation

- We wanted to explore the statistical technique known as L_1, or, LASSO regression from a linear programming perspective.
- Vanderbei claims LASSO works as an LP, so surely it can be done in AMPL.
- If so, does this approach reach the same conclusions as standard statistical libraries?
- We thought these questions would be a fun way to bridge the gap between statistics and linear programming.

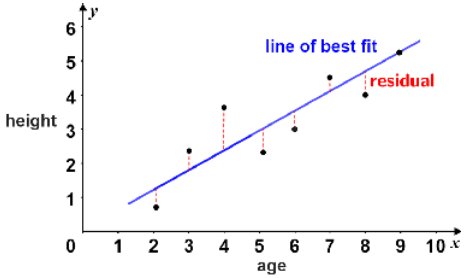
Project Goals

- To convince ourselves, we perfomed this regression in two ways.
- First, using the python library "statsmodels" on a simple dataset.
- Second, using our own AMPL model on the same dataset.
- Lastly, we compare the model coefficients. Are they the same?

Background information

What is regression?

- Regreesion fits a line, plane or hyperplane through a set of points in n dimensional space.
- This fitting is done by treating the "response" variable as a function of other "predictor" variables.
- The fit is improved by minimizing the lines overall distance from the points, otherwise known as the residuals.
- This is known as minimizing the "residual sum of squares", or, RSS.



Penalized Regression

- Having more predictors can often improve model performance, but at the cost of increased complexity.
- There are many consequences to overly complex models, so penalized regression models attempt to help with this.
- LASSO regression is surprise, a type of penalized regression.

Penalized Regression Structure

Penalized regression is made up of two chunks. The RSS part shared with linear regression, and a new penalty chunk.

$$\text{Linear Regression:} \quad \min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\text{LASSO Regression:} \quad \min \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{\text{RSS}} + \underbrace{\lambda \sum_{j=1}^{p} |b_j|}_{\text{Penalty}}$$

$$\hat{y}_i = \sum_{j=1}^{p} b_j x_{ij}$$

LASSO Regression Features

- The penalty is very useful because it can drag coefficients all the way down to zero.
- What this means is LASSO regression can completely remove unimportant variables from a model!
- This makes it a convenient variable selection tool.

LASSO as a Quadratic Program (QP) - Setup

- Converting this into the form of an LP is pretty straightforward, but some modifications are needed.
- The RSS portion becomes the objective function, but it's squared so this is actually a quadratic program (QP).
- The penalty will become the constraint, but the absolute value is inappropriate for an LP (or QP).
- To fix that, we let $b_j = b_j^+ - b_j^-$ where $b_j^+, b_j^- \geq 0$. The absolute value then is just $|b_j| = b_j^+ + b_j^-$.

LASSO as a Quadratic Program (QP) - Execution

This gives us the following program.

$$\min \sum_i (y_i - \hat{y}_i)^2$$

$$\text{subject to: } \sum_{j=1}^{p} (b_j^+ + b_j^-) \leq t$$

$$\hat{y}_i = \sum_j b_j^+ x_{ij} - \sum_j b_j^- x_{ij}$$

$$b_j^+, b_j^- \geq 0$$

Implementation

The Data - Overview

- For this project we'll be using the classic 'mtcars' toy dataset.
- "The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models)."
- This is perfect for as the relatively high number of variables to the low number of rows gives us a perfect environment to show the penalty in action.

The Data - Example Rows

model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1

The Regression Model

Our model uses the following variables from the full dataset:

- mpg: Miles per gallon. (Response variable)
- cyl: Number of cylinders
- disp: Cubic inches of displacement
- hp: Horsepower
- wt: Weight
- qsec: 1/4 mile time; a measure of acceleration

The AMPL Model - Parameters and Variables

```
set Car;
set Variables;

param y {i in Car};
param x {i in Car, j in Variables};
param t;

var bplus{j in Variables}  >= 0;
var bminus{j in Variables} >= 0;
```

The AMPL Model - Obective Function and Constraints

```
minimize SSE:
    sum {i in Car}
        ( y[i] - sum {j in Variables} (bplus[j] - bminus[j]) * x[i,j] )^2;

subject to L1_budget:
    sum {j in Variables: j != 'intercept'} (bplus[j] + bminus[j]) <= t;
```

Results and Conclusion

Model Results

Model	Intercept	cyl	disp	hp	wt	qsec
statsmodels	19.99	-1.64	0.0	-1.16	-2.99	0.0
ampl	20.09	-1.62	0.0	-1.08	-2.94	0.02

As can be seen from this table, our implementation reached nearly identical results.

Conclusion

As we've seen, tackling LASSO regression from the perspective of linear programming is very straightforward.

I think this demonstrates just how close many of our seemingly disparate domains truly are!

Thank You

Questions?

Backup Slides

The Python Model - Setup Code

```
mtcars = pd.read_csv("data/mtcars.csv")
X = mtcars[["cyl", "disp", "hp", "wt", "qsec"]]
y = mtcars['mpg']

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# Add constant for intercept
X_scaled_df = sm.add_constant(X_scaled_df)
```

The Python Model - Modeling Code

```
model = sm.OLS(y, X_scaled_df)
results = model.fit_regularized(method='elastic_net', L1_wt=1.0,
↪  alpha=0.1)

for variable, coefficient in results.params.items():
    print(f"{variable}: {round(coefficient, 5)}")
```