# LASSO Regression as a Quadratic Program

MATH 5593 - Linear Programming

Kevin Ruiz  Brady Lamson

December 3, 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.
- Third, repeat the process for American Community Survey data.
- Lastly, we compare the model coefficients. Are they the same?

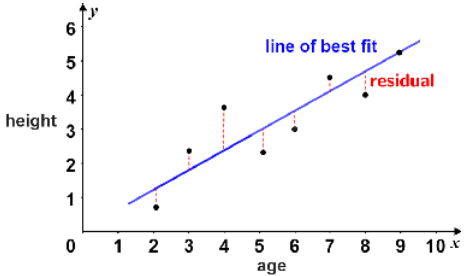
# Background information

# What is regression?

- Regression is a statistical technique that allows us to see the relationship between a **dependent variable** and a set of **independent variables**.
- It is commonly known as a line fit through a set of data points!
- The most basic form of it is $y = \beta_0 + \beta_1 x + \epsilon$ Where the $\beta$ coefficients are estimated from the data and $\epsilon$ represents random error unexplained by the model.

# How is the regression line chosen?

- But how is that line created? How are those $\beta$ coefficients estimated?
- In most situations, no line will perfectly fit the data so we want it to be as close as possible.
- The distance from the line to a given data point is called a **residual**, we want these to be small.
- So, linear regression seeks to minimize the **residual sum of squares**, or **RSS**.



# Why isn't Linear Regression an LP?

Our goal to minimize the RSS can be represented as:

$$\min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

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

Where $y_i$ is the actual response value, and $\hat{y}_i$ is that value estimated by our model.

That looks like an objective function, but it's missing something. The $\beta$ values are completely unconstrained! They could be anything.

So, simple linear regression can't be formulated as an LP. That's where penalized regression comes in.

# Penalized Regression

- In statistical modeling, it can often be advantageous to have as simple a model as possible. Thus, there are tools that penalize overly complex models.
- These tools come in different forms. There are metrics that take the number of dependent variables into account, as well as models with built in penalties that affect the coefficients.
- We're interested in an example of the latter: **LASSO regression**, a type of **penalized regression**.

# Penalized Regression

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

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

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

That penalty is the key here.

# LASSO Regression Penalty

The penalty has some useful features and consequences.

- The penalty can drag coefficients all the way down to zero.
- Because of this, LASSO regression can completely remove unimportant variables from a model!
- This makes it a convenient variable selection tool.
- When re-written, the penalty also gives us the **constraint** we need for an LP.

# 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.

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

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

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

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

# Implementation

# mtcars - 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.

# mtcars - 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 |

# ACS Data - Overview

- For more roboust testing we used an actual data.
- The ACS is a household survey providing information on social, economic, housing, and demographic charactersitics of the U.S. population.
- The ACS is very clean with clean identefiers for each indvidaul surveyed throught time.

# ACS Data - Example Rows

| age | uhrswork | female | wage | schoolyr | schoolyr2 |
|---|---|---|---|---|---|
| 25 | 20 | 0 | 16.666666 | 18.0 | 324.0 |
| 26 | 20 | 1 | 12.745098 | 13.0 | 169.0 |
| 33 | 40 | 0 | 13.725491 | 16.0 | 256.0 |
| 29 | 50 | 1 | 17.647058 | 13.0 | 169.0 |
| 27 | 40 | 0 | 13.725491 | 12.0 | 144.0 |

# The Regression Model mtcars

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 Regression Model for the ACS

Our model uses the following variables from the full dataset:

- wage: Current hourly wage reported
- age: Age of indvidaul
- female: Indicator for female
- schoolyr: The number of years in school
- schoolyr$^2$: The number of years in school squRed
- uhrswork: Hours worked per week

# 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.

# Model Results ACS Data

| Model | Age | Female | Schoolyr | Schoolyr$^2$ | uhrswork | intercept |
|---|---|---|---|---|---|---|
| statsmodels | 2.699 | -4.416 | 0.0 | 10.9886 | 2.9111 | 27.8711 |
| ampl | 2.699 | -4.348 | -0.0862 | 11.0109 | 2.8748 | 27.819 |

The ACS data tells us a different but similar story with some clear numerical differences.

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.

While this model might not be the fastest and can be optimized for larger datasets, it still allows us to truly let the tools learned in class help us formulate a solvable model.

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)}")
```