WSABI Summer Research Lab **Summer 2025**

Lab 15: Regularization and Ridge Regression

Instructor: Jonathan Pipping *Author: Ryan Brill*

15.1 Measuring Player Impact in the NBA

15.1.1 Motivation

We can measure player's offensive skill fairly well using box score data, but it doesn't tell the full story. For example, the impact of role players will be diminished by their reduced minutes, and the defensive impact of non-bigs will be underestimated due to lack of counting stats. We want to use statistical modeling to measure each player's impact, which motivates the **Adjusted Plus-Minus (APM)** model.

15.1.2 Adjusted Plus-Minus

Consider the available data:

- i: index of the i^{th} possession in the dataset.
- $OP1(i), OP2(i), \ldots, OP5(i)$: the 5 offensive players on the court for the i^{th} possession.
- $DP1(i), DP2(i), \ldots, DP5(i)$: the 5 defensive players on the court for the i^{th} possession.
- y_i: our outcome representing the points of the i^{th} possession (positive if scored, negative if allowed).

We want to estimate two parameters for each player $j \in 1, \ldots, m$: one for their offensive impact (β_j) and one for their defensive impact (γ_j). We will include an intercept term α_0 to account for the baseline points scored in a possession. Then our model is as follows:

$$y_i = \alpha_0 + \beta_{OP1(i)} + \beta_{OP2(i)} + \ldots + \beta_{OP5(i)} + \gamma_{DP1(i)} + \gamma_{DP2(i)} + \ldots + \gamma_{DP5(i)} + \epsilon_i$$

where ϵ_i is a mean-zero error term ($\mathbb{E}[\epsilon_i] = 0$). We can define this in matrix form using one-hot encoding for the offensive and defensive players. For each possession i,

$$x_i = [\overset{\text{intercept}}{\bullet}, \overset{\text{Player 1 (off)}}{\bullet}, \ldots, \overset{\text{Player } m \text{ (off)}}{\bullet}, \overset{\text{Player 1 (def)}}{\bullet}, \ldots, \overset{\text{Player } m \text{ (def)}}{\bullet}]$$

Where each offensive $\bullet$ is a 1 if player j is on offense this possession and 0 otherwise, and each defensive $\bullet$ is a 1 if player j is on defense this possession and 0 otherwise. Then the model can be written compactly for each possession i as

$$y_i = x_i^T \boldsymbol{\beta} + \epsilon_i$$

or for the entire dataset as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \alpha_0 \\ \beta_1 \\ \vdots \\ \beta_m \\ \gamma_1 \\ \vdots \\ \gamma_m \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

15.1.3 Fitting the Model

If you fit this model using OLS, you get Adjusted Plus-Minus (APM). However, this will overfit significantly, largely due to extreme multicollinearity (teammates who frequently play together) and having players with very few possessions. We can improve our model by using regularization, which we learned about in lecture. **Regularized Adjusted Plus-Minus (RAPM)** fits this model using **ridge regression**, which shrinks coefficients towards 0. This limits overfitting by penalizing large coefficients.

15.1.4 Your Task

1. One-hot encode the players in your dataset to create the design matrix $\boldsymbol{X}$.
2. Fit the Adjusted Plus-Minus model using OLS.
3. Fit the Adjusted Plus-Minus model using ridge regression.
4. Visualize the two models and compare their performance.

15.1.5 How to Fit a Ridge Regression Model

Below are instructions for fitting a ridge regression model in `R` using the `glmnet` package. Note that the `cv.glmnet` function takes x and y as separate inputs and has built-in cross-validation to get the best lambda. Also, the `glmnet` package standardizes $\boldsymbol{X}$ by default, so we need to set `standardize = FALSE` to avoid this.

```
# load glmnet
library(glmnet)
# set lambdas to cross-validate
lambdas = 10^seq(-3, 3, by = 0.2)
# fit the model
ridge_model = cv.glmnet(x = model_matrix, y = outcome_vector,
                        nfolds = 5, alpha = 0, family = "gaussian",
                        lambda = lambdas, standardize = FALSE)
# get the best lambda
ridge_model$lambda.min
# plot the model
plot(ridge_model)
```