Penalized Models

Fitting Models to Data, Not Data to Models

Model Fitting Series - With Applications in R

Jesse Ghashti

Source code by Stefano Mezzini

November 18, 2025

Centre for Scholarly Communication

The University of British Columbia | Okanagan Campus | Syilx Okanagan Nation Territory

Workshop Series Overview

Session	Topic	Date/Time
1	Simple Linear Regression	Oct 7, 9:00 AM
2	Fitting Linear Models in R	Oct 8, 10:30 AM
3	Multiple Linear Regression in R	Oct 16, 4:00 PM
4	Interaction Terms & Hierarchical Linear Models	Oct 21, 11:00 AM
5	Generalized Linear Models	Oct 23, 4:00 PM
6	Generalized Additive Models (GAMs)	Oct 28, 11:00 AM
7	Interpreting & Predicting from GAMs	Oct 29, 10:30 AM
8	Hierarchical GAMs	Nov 4, 12:00 PM
9	Penalized Models	Nov 18, 11:00 AM
10	Survival Models	Nov 25, 11:00 AM
11	Nonparametric Models	Dec 2, 11:00 AM

New Here?

<——————————-

New to R? Check out the Fundamentals of R series!

GitHub code and slides for today's workshop (and previous workshops)

——————————>

Alternatively, code/slides available at the bottom of `https://csc-ubc-okanagan.github.io/workshops/`

Last Time (Workshop 8) — Quick Recap

Key Concepts

- Reviewed smooth types in `mgcv`: `tp`, `cr`, `cc`, `ad`
- Learned factor smooths (`fs`, `sz`) and random effects (`re`, `mrf`)
- Built interactions with `s()`, `te()`, and `ti()`
- Applied to seasonal × long-term trends in temperature data
- Key insight: `ti()` separates main effects from interactions!

Today: Penalized Regression Models

Today we will...

- Understand why we need regularization (the bias-variance tradeoff)
- Learn Ridge, Lasso, and Elastic Net penalties
- Fit penalized models with `glmnet`
- Choose tuning parameters via cross-validation
- Compare variable selection approaches
- Apply to high-dimensional data (many predictors)

Today we require...

```r
library('dplyr')     # data wrangling
library('tidyr')     # data reshaping
library('ggplot2')   # plotting
library('glmnet')    # penalized regression
library('ISLR')      # example datasets
theme_set(theme_bw(base_size = 15))
```

The Problem: Too Many Predictors

When Standard Regression Fails

Issues with many predictors:

- **Overfitting:** Model fits training data perfectly but predicts poorly
- **Multicollinearity:** Correlated predictors $\Rightarrow$ unstable coefficients
- **High variance:** Small changes in data $\Rightarrow$ large coefficient changes
- $p > n$ **problem:** More predictors than observations $\Rightarrow$ no unique solution

What we want:

- Stable coefficient estimates
- Better out-of-sample predictions
- Variable selection (which predictors matter?)
- Interpretable models

The Bias-Variance Tradeoff

Prediction Error Decomposition

Expected prediction error:

$$\text{MSE} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Unpenalized regression:

- Low bias (fits training data well)
- High variance (sensitive to data changes)
- Poor generalization to new data

Penalized regression:

- Slightly higher bias (shrinks coefficients)
- Much lower variance (more stable)
- Better overall prediction error

Penalized Regression: General Framework

Objective Function

Standard least squares:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2$$

Penalized least squares:

$$\hat{\beta}_\lambda = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \cdot \text{Penalty}(\beta) \right\}$$

where:

- $\lambda \geq 0$ is the **tuning parameter** (controls strength)
- Larger $\lambda \Rightarrow$ more shrinkage toward zero
- $\lambda = 0 \Rightarrow$ ordinary least squares

Ridge Regression (L2 Penalty)

Ridge Penalty

Objective:

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Properties:

- Penalizes the sum of *squared* coefficients
- Shrinks all coefficients toward zero
- **Does NOT set coefficients exactly to zero**
- Good for multicollinearity
- Keeps all predictors in the model

You might consider using ridge regression when all predictors might be relevant, or you have severe multicollinearity

Lasso Regression (L1 Penalty)

Lasso Penalty

Objective:

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Properties:

- Penalizes the sum of *absolute values* of coefficients
- Shrinks coefficients toward zero
- **Can set coefficients exactly to zero** (variable selection!)
- Produces sparse models (fewer predictors)
- Performs automatic feature selection

You might consider using lasso regression when you want a simpler model and believe many predictors are irrelevant

Elastic Net: Best of Both Worlds

Elastic Net Penalty

Objective:

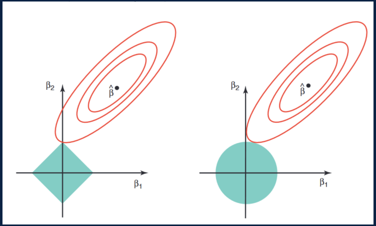
$$\hat{\beta}^{\text{enet}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \right) \right\}$$

Properties:

- Combines Ridge and Lasso penalties
- $\alpha \in [0,1]$ controls the mix: $\alpha = 1$ (Lasso), $\alpha = 0$ (Ridge)
- Can select groups of correlated variables together
- More stable than Lasso when $p > n$

You might consider using Elastic Net when you want variable selection but have correlated predictors

Geometric Intuition: Why Lasso Gives Sparse Solutions



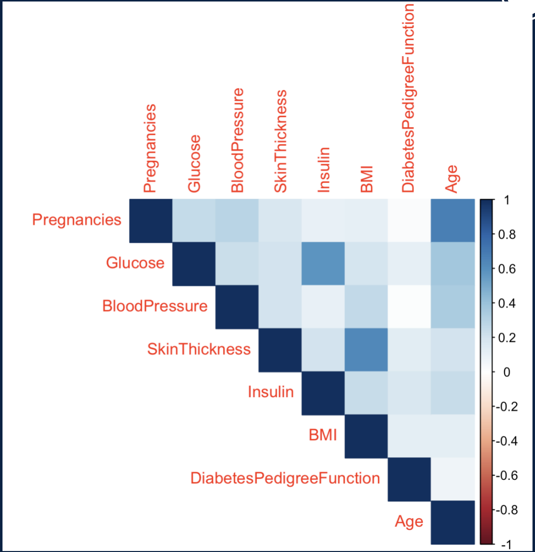
Ridge: Circular constraint $\Rightarrow$ solution rarely at axis

Lasso: Diamond constraint $\Rightarrow$ corners at axes $\Rightarrow$ exact zeros

Exploring the Diabetes Dataset

```
library(ISLR)
library(corrplot)

data(diabetes)
Diabetes <- na.omit(diabetes) # remove missing
dim(Diabetes)

# Check correlations
numVars <- Diabetes[, sapply(Diabetes, is.numeric)]
numVars <- numVars[, !colnames(numVars) %in% "Outcome"]
corMat <- cor(numVars) # get correlations
corrplot(corMat, method = 'color',
         type = 'upper') # plot correlations
```

Many predictors are highly correlated — multicollinearity...



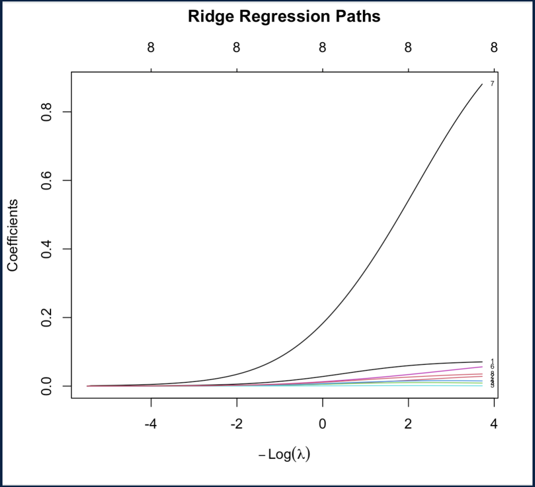
Preparing Data for `glmnet`

```r
# glmnet requires MATRIX input (not data frame)
# model.matrix automatically creates dummy variables for factors
X <- model.matrix(Outcome ~ ., data = Diabetes)[, -1] # remove intercept
y <- Diabetes$Outcome

# split into 70/30 train/test set
set.seed(2025)
trainIDX <- sample(1:nrow(X), size = floor(0.7 * nrow(X)))
testIDX <- setdiff(1:nrow(X), trainIDX)
Xtrain <- X[trainIDX, ]
ytrain <- y[trainIDX]
Xtest <- X[testIDX, ]
ytest <- y[testIDX]
```

`glmnet` requires matrix input and automatically standardizes predictors.

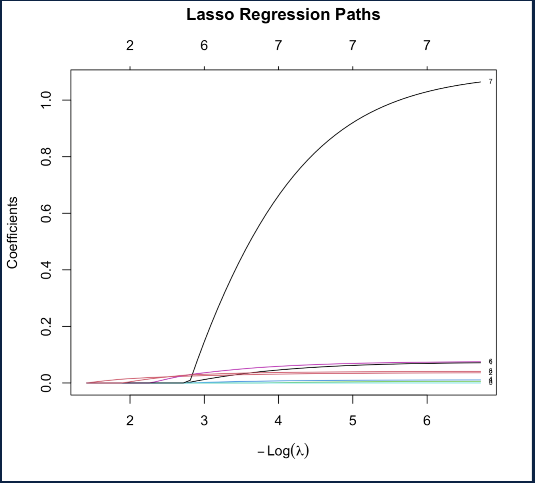
Fitting Ridge Regression

```
# Fit ridge regression
# alpha = 0 for ridge

ridgeModel <- glmnet(X, y, alpha = 0, family = "binomial")
plot(ridgeModel, xvar = 'lambda', label = TRUE)
title('Ridge Regression Paths', line = 3)

# coefficients at specific lambdas
coef(ridgeModel, s = 50)
coef(ridgeModel, s = 500)
coef(ridgeModel, s = 5000)
# Take a look at these :)v
```



Fitting Lasso Regression

```r
# Fit lasso regression
# alpha = 1 for lasso
lassoModel <- glmnet(X, y, alpha = 1, family = "binomial")
plot(lassoModel, xvar = 'lambda', label = TRUE)
title('Lasso Regression Paths', line = 3)

# coefficients at specific lambdas (play around with these!)
coef(lassoModel, s = 50)
coef(lassoModel, s = 500)
coef(lassoModel, s = 5000)
# Notice that many coefficients are
# EXACTLY zero (which no longer contribute to the model)
```

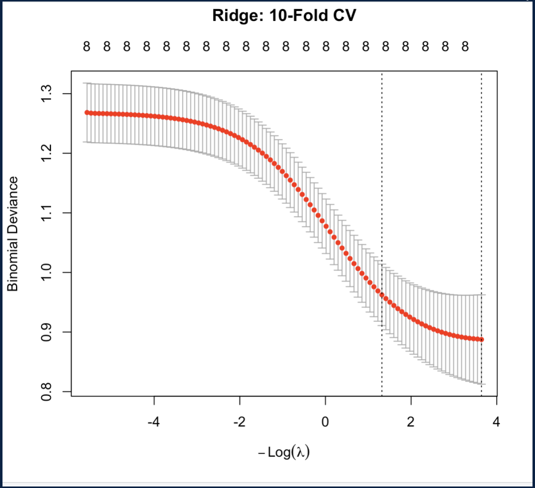


Cross-Validation: Choosing λ for Ridge

```r
# 10-fold cross-validation for Ridge
set.seed(2025)
cvRidge <- cv.glmnet(Xtrain,
                      ytrain,
                      alpha = 0,
                      nfolds = 10,
                      family = "binomial)

plot(cvRidge)
title('Ridge: 10-Fold CV', line = 3)

# best lambda values
cvRidge$lambda.min   # min CV error is 0.02615955
cvRidge$lambda.1se   # 1 SE rule is 0.2677511

# optimal coefficients
coef(cvRidge, s = 'lambda.min')
```

lambda.min: Minimizes CV error

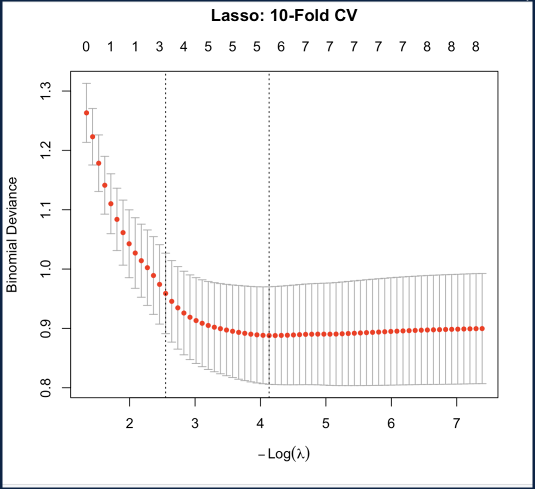
lambda.1se: Simpler, more stable



Cross-Validation: Choosing λ for Lasso

```r
# 10-fold cross-validation for Lasso
set.seed(2025)
cvLasso <- cv.glmnet(Xtrain,
                     ytrain,
                     alpha = 1,
                     nfolds = 10,
                     family = "binomial")

plot(cvLasso)
title('Lasso: 10-Fold CV', line = 3)

# best lambda values
cvLasso$lambda.min # 0.01605126
cvLasso$lambda.1se # 0.07805087

# optimal coefficients
coef(cvLasso, s = 'lambda.min')
```

Number at top shows how many variables included



Making Predictions and Comparing Models

```r
# predict on test set using optimal lambda
ridgePredict <- predict(cvRidge, s = 'lambda.min', newx = Xtest, type = "response")
lassoPredict <- predict(cvLasso, s = 'lambda.min', newx = Xtest, type = "response)

# we will compare to multiple linear regression as well.
lmModel <- glm(Salary ~ ., data = Hitters[trainIDX, ], family = "binomial")
lmPredict <- predict(lmModel, newdata = Hitters[testIDX, ], type = "response")

accuracy <- function(actual, predicted) {
  predicted <- ifelse(predicted >= 0.5, 1, 0)
  matches <- sum(diag(Thresher::matchLabels(table(actual, predicted))))
  return(matches/length(predicted))
}

accuracyLM <- accuracy(as.numeric(ytest), lmPredict)
accuracyRIDGE <- accuracy(as.numeric(ytest), ridgePredict)
accuracyLASSO <- accuracy(as.numeric(ytest), lassoPredict)

data.frame(
  Model = c('LM', 'Ridge', 'Lasso'),
  accuracy = c(accuracyLM, accuracyRIDGE, accuracyLASSO)
)
```

Test Set Performance Comparison

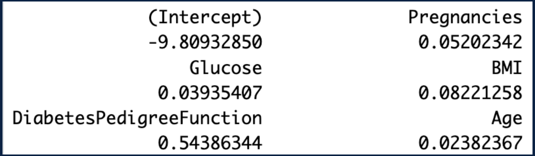
Model	Test Accuracy	# Variables
LM	0.7223	9
Ridge	0.7426	9
Lasso	0.7223	6

Lasso does better as good as LM with predictions but with fewer variables (simpler, more interpretable)

Both penalized methods outperform LM on out-of-sample data, but for different reasons

Variable Selection: Which Predictors Matter?

```r
lassoCoefficients <- coef(cvLasso,
                          s = 'lambda.min')
lassoCoefficients
# which variables selected?
varsSelected <-
  lassoCoefficients[lassoCoefficients[, 1] != 0, ]
varsSelected
```



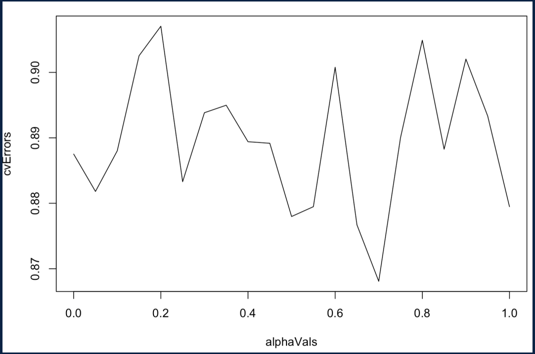
```
             (Intercept)              Pregnancies 
             -9.80932850               0.05202342 
                 Glucose                      BMI 
              0.03935407               0.08221258 
DiabetesPedigreeFunction                      Age 
              0.54386344               0.02382367 
```

Elastic Net: Combining Ridge and Lasso

```r
set.seed(2025)
cvENET <- cv.glmnet(Xtrain, ytrain, alpha = 0.5, nfolds = 10, family = "binomial")
plot(cvENET)
title('Elastic Net ( = 0.5): 10-Fold CV', line = 3)
enetPredict <- predict(cvENET, s = 'lambda.min', newx = Xtest, type = "response")
accuracyENET <- accuracy(as.numeric(ytest), enetPredict)
data.frame(
  Model = c('LM', 'Ridge', 'Lasso', 'Elastic Net'),
  accuracy = c(accuracyLM, accuracyRIDGE, accuracyLASSO, accuracyENET)
)

# how many variables selected?
enetCoefficients <- coef(cvENET, s = 'lambda.min')
sum(enetCoefficients[-1] != 0) # 7 selected
```

Tuning α: Finding the Best Mix

```r
alphaVals <- seq(0, 1, by = 0.05)
cvResults <- list()
set.seed(2025)
for (i in seq_along(alphaVals)) {
  cvResults[[i]] <-
    cv.glmnet(Xtrain, ytrain,
              alpha = alphaVals[i],
              nfolds = 10,
              family = "binomial")
}
# minimum CV error
cvErrors <- sapply(cvResults,
                   function(x) min(x$cvm))
alphaOptimal <-
  alphaVals[which.min(cvErrors)]
alphaOptimal # 0.7
```



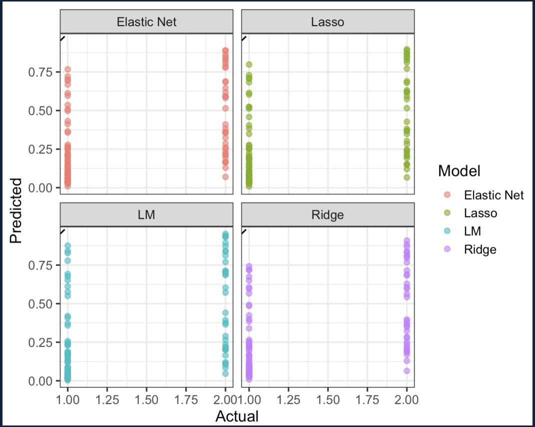
Final Model Comparison

Model	Test Accuracy	# Variables	α
LM	0.7228	9	N/A
Ridge	0.7426	9	0
Lasso	0.7228	6	1
Elastic Net (α=0.5)	07327	7	0.5
Optimal (α=0.7)	0.7228	5	0.7

Optimal $\alpha \approx 0.7$: Closer to Lasso, emphasizing variable selection

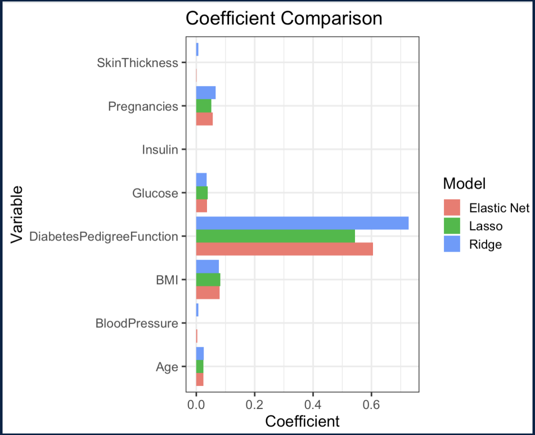
Visualizing Predictions: All Models

```
predictionCompare <- tibble(
  Actual = as.numeric(ytest),
  LM = as.vector(lmPredict),
  Ridge = as.vector(ridgePredict),
  Lasso = as.vector(lassoPredict),
  `Elastic Net` = as.vector(enetPredict)
) %>%
  pivot_longer(cols = -Actual,
               names_to = 'Model',
               values_to = 'Predicted')

ggplot(predictionCompare,
       aes(Actual, Predicted,
           color = Model)) +
  geom_point(alpha = 0.6) +
  geom_abline(slope = 1,
              intercept = 0,
              linetype = 'dashed') +
  facet_wrap(~ Model)
```



Coefficient Comparison Across Methods

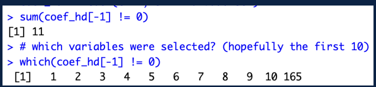
```r
coefficientComparison <- tibble(
  Variable = rownames(lassoCoefficients)[-1],
  Ridge = as.vector(coef(cvRidge, s = 'lambda.min')[-1]),
  Lasso = as.vector(lassoCoefficients[-1]),
  `Elastic Net` = as.vector(enetCoefficients[-1])
) %>%
  pivot_longer(cols = -Variable,
               names_to = 'Model',
               values_to = 'Coefficient')
ggplot(coefficientComparison,
       aes(Variable, Coefficient,
           fill = Model)) +
  geom_col(position = 'dodge') +
  coord_flip() +
  labs(title = 'Coefficient Comparison')
```



High-Dimensional Data: $p > n$

```r
# simulate high-dimensional data where p > n
set.seed(2025)
n <- 100     # observations
p <- 200     # predictors

# only first 10 predictors are truly important
Xhd <- matrix(rnorm(n * p), n, p)
trueCoefficients <- c(rep(2, 5), rep(-2, 5), rep(0, p - 10))
yhd <- Xhd %*% trueCoefficients

# try the linear model, it should fail!
# lmHD <- lm(yhd ~ Xhd)

set.seed(2025)
cvHD <- cv.glmnet(Xhd, yhd, alpha = 1, nfolds = 10)

# how many variables selected?
coef_hd <- coef(cvHD, s = 'lambda.1se')
sum(coef_hd[-1] != 0)

# which variables were selected? (hopefully the first 10)
which(coef_hd[-1] != 0)
```

High-Dimensional Lasso ($p > n$)



```
> sum(coef_hd[-1] != 0)
[1] 11
> # which variables were selected? (hopefully the first 10)
> which(coef_hd[-1] != 0)
 [1]   1   2   3   4   5   6   7   8   9  10 165
```

Lasso identifies relevant variables even when $p > n$ (impossible for LM)

Choosing Between `lambda.min` and `lambda.1se`

The One Standard Error Rule

lambda.min:

- Minimizes cross-validation error
- Best predictive performance
- May overfit slightly
- More variables selected (less sparse)

lambda.1se:

- Largest λ within 1 SE of minimum
- More regularization (simpler model)
- Better generalization in practice
- Fewer variables (more sparse)
- Preferred for interpretability

Rule of thumb: Use `lambda.1se` for simpler, more stable models

Additional Information

Important Details

Standardization:

- `glmnet` standardizes by default (`standardize = TRUE`)
- Essential for fair penalization across different scales
- Final coefficients returned on original scale

Categorical predictors:

- Use `model.matrix()` to create dummy variables
- `glmnet` doesn't handle factors directly
- Can group dummies with `penalty.factor`

Response transformations:

- For GLM families: set `family` argument
- Logistic: `family = 'binomial'`
- Poisson: `family = 'poisson'`

Decisions, Decisions, Decisions

Consideration	**Recommendation**
Severe multicollinearity	Ridge (keeps all variables)
Want interpretable model	Lasso (variable selection)
$p > n$ problem	Lasso or Elastic Net
Correlated predictors + selection	Elastic Net ($\alpha \approx 0.5 - 0.7$)
All predictors relevant	Ridge
Many irrelevant predictors	Lasso
Uncertain	Tune α via CV

Best Practices for Penalized Regression

Do's

- Always use cross-validation to choose λ
- Consider `lambda.1se` for simpler, more stable models
- Standardize predictors (glmnet does this automatically)
- Split data into train/test for honest evaluation
- Try multiple α values for Elastic Net
- Use `model.matrix()` to handle categorical predictors

Don'ts

- Don't use a single λ without CV
- Don't forget to check test set performance
- Don't interpret Ridge coefficients as "importance"
- Don't ignore the geometric interpretation

Connection to GAMs: Penalization Everywhere!

Unified Framework

Remember from Workshop 6-8:

- GAMs use penalization on smooth functions
- Penalty controls "wiggliness" of curves
- REML/ML chooses optimal smoothing parameter

Penalized regression:

- Similar idea applied to coefficients
- Penalty controls "size" of coefficients
- Cross-validation chooses optimal λ

Common theme:

- Bias-variance tradeoff
- Automatic complexity control
- Better out-of-sample prediction

You Try: Penalized Regression Practice

Exercise: Compare methods

```r
# Use the College dataset from ISLR
library(ISLR)
data(College)
?College

# Task: Predict Graduation Rate (Grad.Rate) from all other variables

# 1. Prepare data
#    - Create X matrix and y vector
#    - Split into train (70%) and test (30%)

# 2. Fit three models
#    - Ridge (alpha = 0)
#    - Lasso (alpha = 1)
#    - Elastic Net (alpha = 0.5)
#    - Use CV to find optimal lambda for each

# 3. Compare performance
#    - Calculate test RMSE for each
#    - How many variables does Lasso select?
#    - Which model performs best?

# 4. Interpret
#    - Plot coefficient paths
#    - Which predictors are most important?
#    - Try different alpha values - does it matter?
```

Key Takeaways

- Penalized regression addresses overfitting, multicollinearity, and $p > n$
- Ridge shrinks coefficients but keeps all variables
- Lasso performs variable selection (sets coefficients to zero)
- Elastic Net combines both approaches (tune α)
- Always use cross-validation to choose λ
- Consider `lambda.1se` for simpler, more stable models
- Penalized methods usually outperform LM on test data
- The bias-variance tradeoff is central to statistical learning
- `glmnet` is fast, efficient, and works when $p > n$
- Geometric intuition explains why Lasso gives sparse solutions

What's Next?

Additional Questions?
Book an Appointment!

Next Workshop:

Survival Models
–> November 25, 11:00 AM

Thank You!

Questions?

Workshop Materials:

`https://github.com/csc-ubc-okanagan/ubco-csc-modeling-workshop`

Contact:

jesse.ghashti@ubc.ca