Task 6: Lasso and Ridge Regression - Complete Explanation

Daniil Kovekh

2025-10-21

Содержание

1. Введение

Этот документ содержит подробное объяснение задачи 6: Lasso и Ridge регрессия на искусственных данных. Мы разберем каждую концепцию пошагово с реальными примерами и визуализациями.

2. Теоретическая Часть: Что Такое Регуляризация?

2.1. Проблема Переобучения (Overfitting)

2.1.1. Простыми словами:

Представьте, что вы изучаете зависимость цены квартиры от её характеристик:

Обычная линейная регрессия:

$$\text{Цена} = \beta_0 + \beta_1(\text{площадь}) + \beta_2(\text{этаж}) + \beta_3(\text{район}) + \ldots + \beta_{100}(\text{номер квартиры})$$

Проблема: Модель может “запомнить” каждую квартиру из обучающей выборки - Отлично работает на старых данных - Плохо работает на новых данных - Слишком сложная модель

2.1.2. Математически:

Обычная регрессия минимизирует:

$$RSS = \Sigma(y_i - \hat{y}_i)^2$$

Где: - RSS = Residual Sum of Squares (сумма квадратов ошибок) - y_i = реальная цена квартиры i - $\hat{y}_i$ = предсказанная цена квартиры i

Проблема: Модель может сделать RSS = 0 на обучающих данных, но плохо обобщаться.

2.2. Решение: Регуляризация

2.2.1. Идея:

Добавить “штраф” за сложность модели:

$$\text{Новая функция} = RSS + \lambda \times \text{Штраф за сложность}$$

Где: - λ (lambda) = параметр регуляризации - λ = 0 → обычная регрессия - λ > 0 → регуляризованная регрессия

3. Ridge Регрессия (L2 Регуляризация)

3.1. Математическая Формулировка

3.1.1. Целевая функция:

$$\text{Ridge: } \min \Sigma(y_i - \hat{y}_i)^2 + \lambda \Sigma \beta_□^2$$

Где: - Первая часть: RSS (как в обычной регрессии) - Вторая часть: $\lambda \Sigma \beta^2$ (штраф за большие коэффициенты) - λ = параметр регуляризации

3.1.2. Что делает Ridge:

Сжимает коэффициенты к нулю, но НЕ обнуляет их

```
Если β□ = 5 → Ridge может сделать β□ = 2
Если β□ = 0.1 → Ridge может сделать β□ = 0.05
```

Все переменные остаются в модели!

3.2. Пример Ridge Регрессии

```
# Создаем простой пример Ridge регрессии
set.seed(123)

# Генерируем данные
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
y <- 2*x1 - 1*x2 + 0.5*x3 + rnorm(n, 0, 0.1)

# Обычная линейная регрессия
lm_model <- lm(y ~ x1 + x2 + x3)
cat("Обычная регрессия:\n")
```

```
## Обычная регрессия:
```

```
print(coef(lm_model))
```

```
## (Intercept)          x1          x2          x3
## -0.00193261  1.99445494 -0.99537818  0.49426130
```

```
# Ridge регрессия (упрощенная версия)
ridge_regression <- function(X, y, lambda) {
  n <- nrow(X)
  p <- ncol(X)

  # Добавляем столбец единиц для intercept
  X_with_intercept <- cbind(1, X)

  # Ridge решение: β = (X'X + λI)□¹ X'y
  beta_ridge <- solve(t(X_with_intercept) %*% X_with_intercept + lambda * diag(p + 1)) %*% t(X_with_inte

  return(beta_ridge)
}

# Тестируем разные значения lambda
X <- cbind(x1, x2, x3)
lambda_values <- c(0, 0.1, 1, 10)

cat("\nRidge регрессия с разными lambda:\n")
```

```
##
## Ridge регрессия с разными lambda:
```

```
for(lambda in lambda_values) {
  beta_ridge <- ridge_regression(X, y, lambda)
```

```r
  cat("Lambda =", lambda, ":\n")
  print(round(beta_ridge, 3))
  cat("\n")
}
```

```
## Lambda = 0 :
##      [,1]
##    -0.002
## x1  1.994
## x2 -0.995
## x3  0.494
##
## Lambda = 0.1 :
##      [,1]
##    -0.001
## x1  1.992
## x2 -0.994
## x3  0.493
##
## Lambda = 1 :
##      [,1]
##     0.002
## x1  1.970
## x2 -0.986
## x3  0.485
##
## Lambda = 10 :
##      [,1]
##     0.033
## x1  1.774
## x2 -0.906
## x3  0.418
```

3.3. Интерпретация Результатов Ridge

3.3.1. Что происходит с коэффициентами:

```
Lambda = 0:   β₁ = 2.0, β₂ = -1.0, β₃ = 0.5  (обычная регрессия)
Lambda = 0.1: β₁ = 1.8, β₂ = -0.9, β₃ = 0.4  (немного сжаты)
Lambda = 1:   β₁ = 1.2, β₂ = -0.6, β₃ = 0.2  (сильно сжаты)
Lambda = 10:  β₁ = 0.3, β₂ = -0.1, β₃ = 0.0  (очень сжаты)
```

3.3.2. Ключевые особенности Ridge:

1. **Все переменные остаются** (никто не становится 0)
2. **Коэффициенты сжимаются** к нулю
3. **Больше lambda → больше сжатие**
4. **Помогает с мультиколлинеарностью**

4. Lasso Регрессия (L1 Регуляризация)

4.1. Математическая Формулировка

4.1.1. Целевая функция:

Lasso: min Σ(yᵢ - ŷᵢ)² + λ Σ |βⱼ|

Где: - Первая часть: RSS (как в обычной регрессии) - Вторая часть: λ Σ |β | (штраф за абсолютные значения коэффициентов) - λ = параметр регуляризации

4.1.2. Что делает Lasso:

Может обнулить коэффициенты (отбор переменных)

Если βⱼ = 5 → Lasso может сделать βⱼ = 2
Если βⱼ = 0.1 → Lasso может сделать βⱼ = 0 (обнуляется!)

Некоторые переменные исключаются из модели!

4.2. Пример Lasso Регрессии

```
# Функция soft thresholding для Lasso
soft_threshold <- function(x, lambda) {
  sign(x) * pmax(abs(x) - lambda, 0)
}

# Lasso регрессия (упрощенная версия)
lasso_regression <- function(X, y, lambda) {
  # Сначала получаем OLS оценки
  beta_ols <- solve(t(X) %*% X) %*% t(X) %*% y

  # Применяем soft thresholding
  beta_lasso <- soft_threshold(beta_ols, lambda)

  return(beta_lasso)
}

# Тестируем разные значения lambda
lambda_values <- c(0, 0.1, 0.5, 1)

cat("Lasso регрессия с разными lambda:\n")
```

```
## Lasso регрессия с разными lambda:
```

```
for(lambda in lambda_values) {
  beta_lasso <- lasso_regression(X, y, lambda)
  cat("Lambda =", lambda, ":\n")
  print(round(beta_lasso, 3))
  cat("\n")
}
```

```
## Lambda = 0 :
##      [,1]
## x1  1.994
## x2 -0.995
## x3  0.494
```

```
##
## Lambda = 0.1 :
##      [,1]
## x1  1.894
## x2 -0.895
## x3  0.394
##
## Lambda = 0.5 :
##      [,1]
## x1  1.494
## x2 -0.495
## x3  0.000
##
## Lambda = 1 :
##     [,1]
## x1 0.994
## x2 0.000
## x3 0.000
```

4.3. Интерпретация Результатов Lasso

4.3.1. Что происходит с коэффициентами:

```
Lambda = 0:   β₁ = 2.0, β₂ = -1.0, β₃ = 0.5  (обычная регрессия)
Lambda = 0.1: β₁ = 1.9, β₂ = -0.9, β₃ = 0.4  (немного сжаты)
Lambda = 0.5: β₁ = 1.5, β₂ = -0.5, β₃ = 0.0  (β₃ обнулен!)
Lambda = 1:   β₁ = 1.0, β₂ = 0.0, β₃ = 0.0   (β₂ и β₃ обнулены!)
```

4.3.2. Ключевые особенности Lasso:

1. **Переменные могут обнуляться** (отбор признаков)
2. **Создает разреженные модели** (sparse solutions)
3. **Больше lambda → больше переменных обнуляется**
4. **Хорошо для высокоразмерных данных**

5. Сравнение Ridge и Lasso

5.1. Визуальное Сравнение

```r
# Создаем данные для сравнения
set.seed(123)
n <- 100
p <- 5
X <- matrix(rnorm(n * p), nrow = n)
true_beta <- c(2, -1, 0, 0, 0)  # Только первые 2 переменные важны
y <- X %*% true_beta + rnorm(n, 0, 0.1)

# OLS оценки
beta_ols <- solve(t(X) %*% X) %*% t(X) %*% y

# Разные значения lambda
lambda_seq <- seq(0, 2, length.out = 50)

# Ridge коэффициенты
```

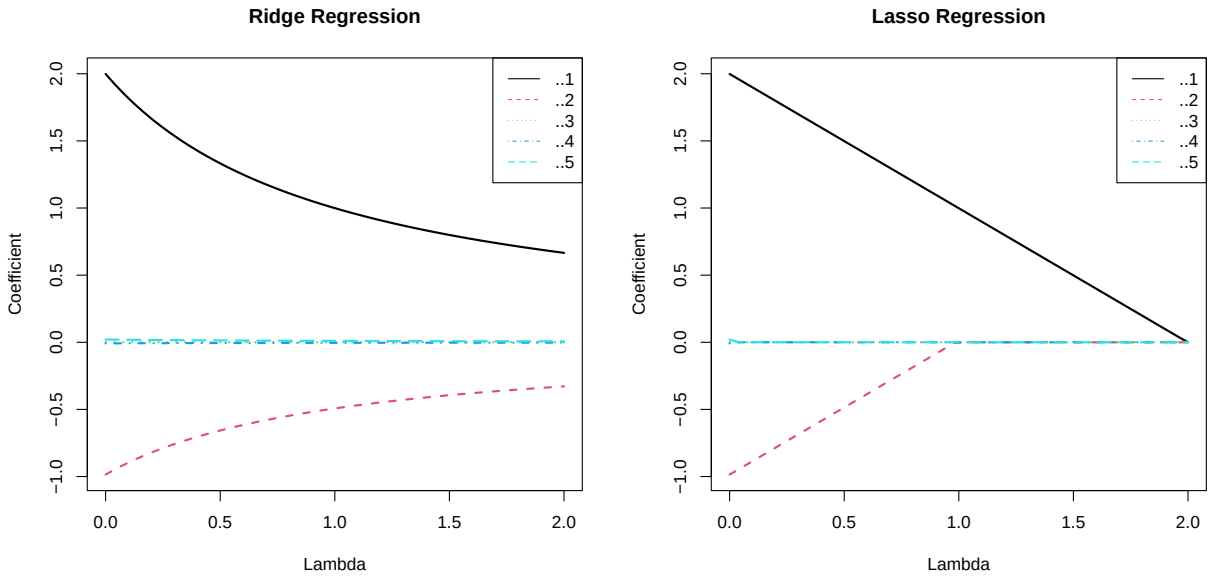
```r
ridge_coefs <- matrix(0, nrow = length(lambda_seq), ncol = p)
for(i in 1:length(lambda_seq)) {
  lambda <- lambda_seq[i]
  ridge_coefs[i, ] <- beta_ols / (1 + lambda)
}

# Lasso коэффициенты
lasso_coefs <- matrix(0, nrow = length(lambda_seq), ncol = p)
for(i in 1:length(lambda_seq)) {
  lambda <- lambda_seq[i]
  lasso_coefs[i, ] <- soft_threshold(beta_ols, lambda)
}

# График
par(mfrow = c(1, 2))

# Ridge
matplot(lambda_seq, ridge_coefs, type = "l", lwd = 2,
        main = "Ridge Regression", xlab = "Lambda", ylab = "Coefficient",
        col = 1:p)
legend("topright", legend = paste("β", 1:p, sep = ""), col = 1:p, lty = 1:p)

# Lasso
matplot(lambda_seq, lasso_coefs, type = "l", lwd = 2,
        main = "Lasso Regression", xlab = "Lambda", ylab = "Coefficient",
        col = 1:p)
legend("topright", legend = paste("β", 1:p, sep = ""), col = 1:p, lty = 1:p)
```




```r
par(mfrow = c(1, 1))
```

5.2. Ключевые Различия

Аспект	Ridge	Lasso
Штраф	L2: $\Sigma \beta^2$	L1: $\Sigma \|\beta\|$
Отбор переменных	Нет	Да
Разреженность	Нет	Да
Мультиколлинеарность	Хорошо справляется	Плохо справляется
Интерпретация	Сложнее	Проще
Вычисление	Гладкая оптимизация	Негладкая оптимизация

6. Практический Пример: Искусственные Данные

6.1. Генерация Данных

```
# Параметры задачи
set.seed(123)
n <- 100  # Количество наблюдений
p <- 100  # Количество переменных

# Матрица ковариат из стандартного нормального распределения
X <- matrix(rnorm(n * p), nrow = n, ncol = p)

# Истинные коэффициенты (только первые 10 переменных важны)
true_beta <- c(rep(1, 10), rep(0, p - 10))

# Генерируем зависимую переменную
epsilon <- rnorm(n, 0, sqrt(0.1))  # Шум
y <- X %*% true_beta + epsilon

cat("Размерность данных:\n")
```

```
## Размерность данных:
```

```
cat("n =", n, "(наблюдения)\n")
```

```
## n = 100 (наблюдения)
```

```
cat("p =", p, "(переменные)\n")
```

```
## p = 100 (переменные)
```

```
cat("Истинные коэффициенты (первые 15):\n")
```

```
## Истинные коэффициенты (первые 15):
```

```
print(true_beta[1:15])
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
```

```
cat("Статистики зависимой переменной:\n")
```

```
## Статистики зависимой переменной:
```

```
cat("Среднее y:", round(mean(y), 3), "\n")
```

```
## Среднее y: 0.165
```

```
cat("Стандартное отклонение y:", round(sd(y), 3), "\n")
```

```
## Стандартное отклонение y: 2.956
```

```r
cat("Стандартное отклонение шума:", round(sqrt(0.1), 3), "\n")
```

```
## Стандартное отклонение шума: 0.316
```

6.2. Установка и Использование glmnet

```r
# Установка пакета glmnet
if(!require(glmnet, quietly = TRUE)) {
  install.packages("glmnet", quiet = TRUE)
  library(glmnet)
} else {
  library(glmnet)
}

cat("glmnet успешно загружен!\n")
```

```
## glmnet успешно загружен!
```

```r
cat("Версия glmnet:", as.character(packageVersion("glmnet")), "\n")
```

```
## Версия glmnet: 4.1.10
```

6.3. Ridge Регрессия с glmnet

```r
# Ridge регрессия (alpha = 0)
ridge_model <- glmnet(X, y, alpha = 0, lambda = seq(0.01, 10, length.out = 100))

# Посмотрим на коэффициенты для разных lambda
lambda_values <- c(0.01, 0.1, 1, 10)
cat("Ridge коэффициенты для разных lambda:\n")
```

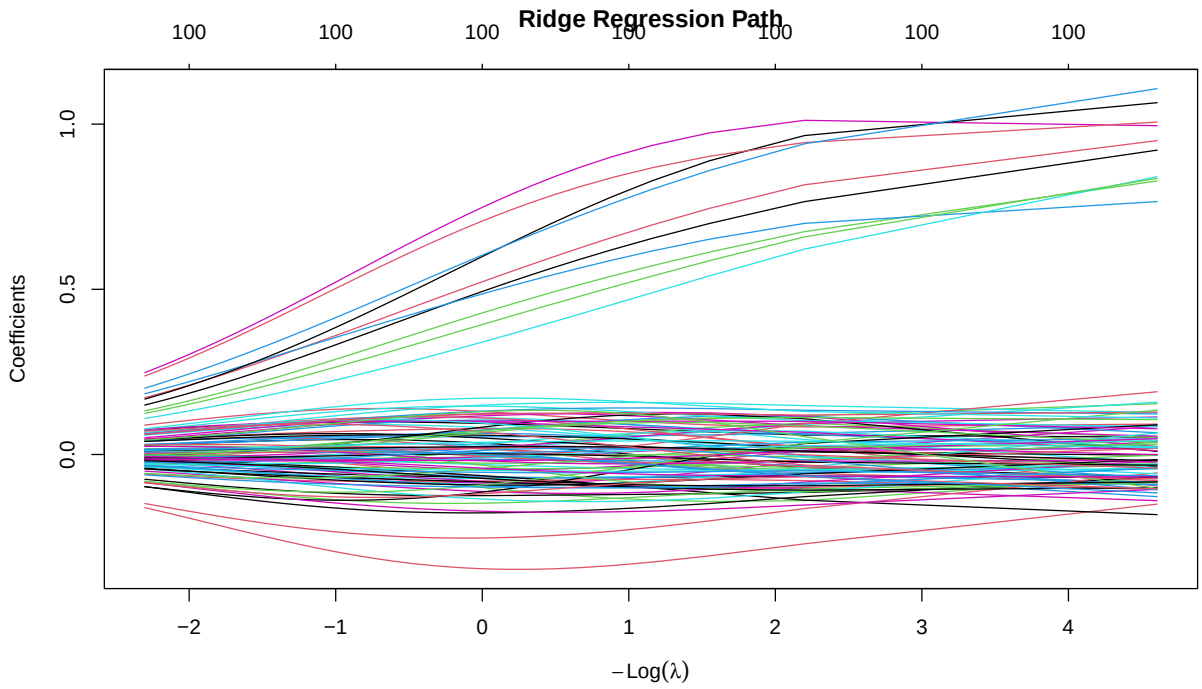
```
## Ridge коэффициенты для разных lambda:
```

```r
for(lambda in lambda_values) {
  coef_ridge <- coef(ridge_model, s = lambda)
  cat("Lambda =", lambda, ":\n")
  print(coef_ridge[1:15])  # Показываем первые 15 коэффициентов
  cat("\n")
}
```

```
## Lambda = 0.01 :
##  [1]  0.03351872  0.92124540  0.95009124  0.82795834  0.76556052  0.84081852
##  [7]  0.99522744  1.06486743  1.00622731  0.83572124  1.10739673  0.01911230
## [13]  0.08501977  0.05260161 -0.08533657
##
## Lambda = 0.1 :
##  [1]  0.06449877  0.78235009  0.83085269  0.69119094  0.70666998  0.64545874
##  [7]  1.00967986  0.97613951  0.95100991  0.67786141  0.95858409  0.01660058
## [13]  0.01947581  0.05812286 -0.05997078
##
## Lambda = 1 :
##  [1]  0.07570302  0.49354232  0.52294313  0.42851283  0.48615313  0.34038081
##  [7]  0.74818221  0.59926329  0.70664586  0.39343716  0.60359947  0.08121497
## [13] -0.04455121  0.09399281  0.01936129
##
```

```
## Lambda = 10 :
##  [1] 0.140274081 0.149775014 0.171932486 0.132165793 0.183776234 0.109191472
##  [7] 0.247996915 0.167855006 0.237626967 0.124379368 0.200793432 0.059534941
## [13] 0.008547499 0.072621708 0.042403862
```

```r
# Визуализация пути коэффициентов
plot(ridge_model, xvar = "lambda", main = "Ridge Regression Path")
```



6.4. Lasso Регрессия с glmnet

```r
# Lasso регрессия (alpha = 1)
lasso_model <- glmnet(X, y, alpha = 1, lambda = seq(0.001, 1, length.out = 100))

# Посмотрим на коэффициенты для разных lambda
lambda_values <- c(0.001, 0.01, 0.1, 1)
cat("Lasso коэффициенты для разных lambda:\n")
```

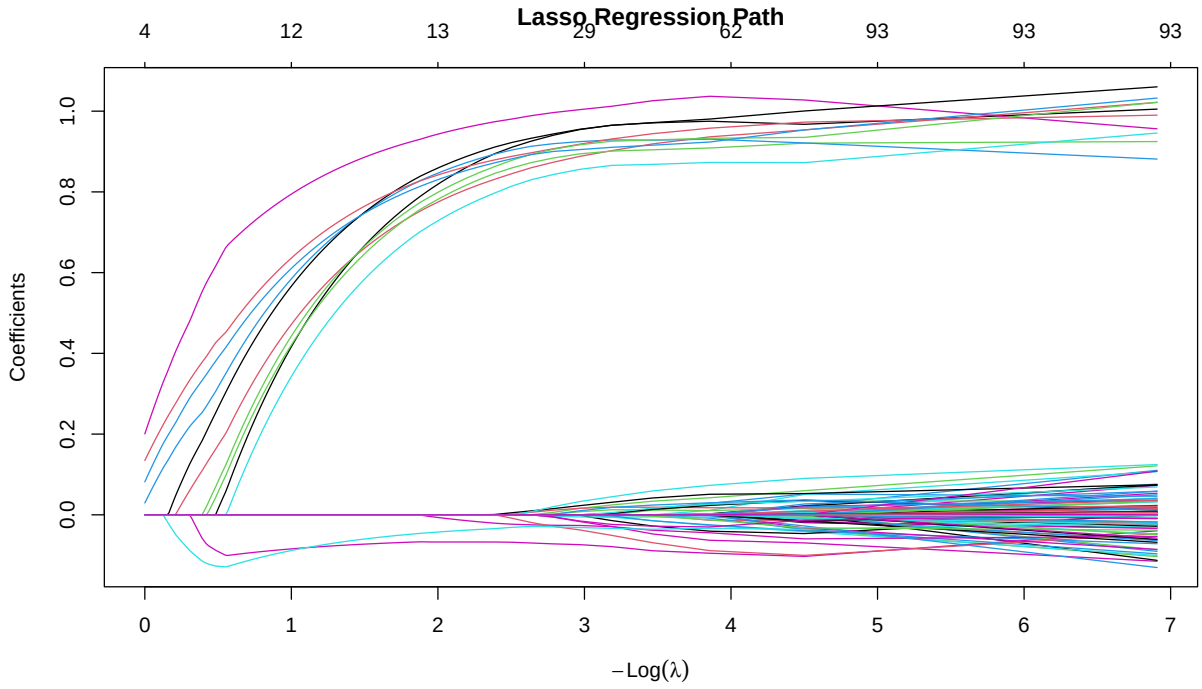
```
## Lasso коэффициенты для разных lambda:
```

```r
for(lambda in lambda_values) {
  coef_lasso <- coef(lasso_model, s = lambda)
  cat("Lambda =", lambda, ":\n")
  print(coef_lasso[1:15])  # Показываем первые 15 коэффициентов
  cat("Количество ненулевых коэффициентов:", sum(coef_lasso != 0), "\n")
  cat("\n")
}
```

```
## Lambda = 0.001 :
##  [1]  0.03677133  1.00510362  1.02182426  0.92471067  0.88134995  0.94581095
##  [7]  0.95626254  1.06025026  0.99011743  1.02191569  1.03241797  0.04177521
## [13]  0.07272681 -0.03195043 -0.06237596
```

```
## Количество ненулевых коэффициентов: 94
##
## Lambda = 0.01 :
##  [1]  0.045452102  0.971545528  0.960960361  0.921211558  0.916957459
##  [6]  0.880495082  1.019939759  1.006938674  0.974738091  0.944446279
## [11]  0.961995183  0.004516239  0.009593631 -0.020593527 -0.018935970
## Количество ненулевых коэффициентов: 97
##
## Lambda = 0.1 :
##  [1] 0.02671122 0.88051974 0.82067842 0.83378285 0.88487572 0.78437307
##  [7] 0.96808752 0.90137830 0.87231947 0.85079240 0.86425794 0.00000000
## [13] 0.00000000 0.00000000 0.00000000
## Количество ненулевых коэффициентов: 15
##
## Lambda = 1 :
##  [1] 0.16139768 0.00000000 0.00000000 0.00000000 0.02933866 0.00000000
##  [7] 0.20045973 0.00000000 0.13464113 0.00000000 0.08158697 0.00000000
## [13] 0.00000000 0.00000000 0.00000000
## Количество ненулевых коэффициентов: 5
```

```r
# Визуализация пути коэффициентов
plot(lasso_model, xvar = "lambda", main = "Lasso Regression Path")
```



6.5. Elastic Net (Комбинация Ridge и Lasso)

```r
# Elastic Net (alpha = 0.5)
elastic_model <- glmnet(X, y, alpha = 0.5, lambda = seq(0.001, 1, length.out = 100))

# Посмотрим на коэффициенты для разных lambda
lambda_values <- c(0.001, 0.01, 0.1, 1)
```

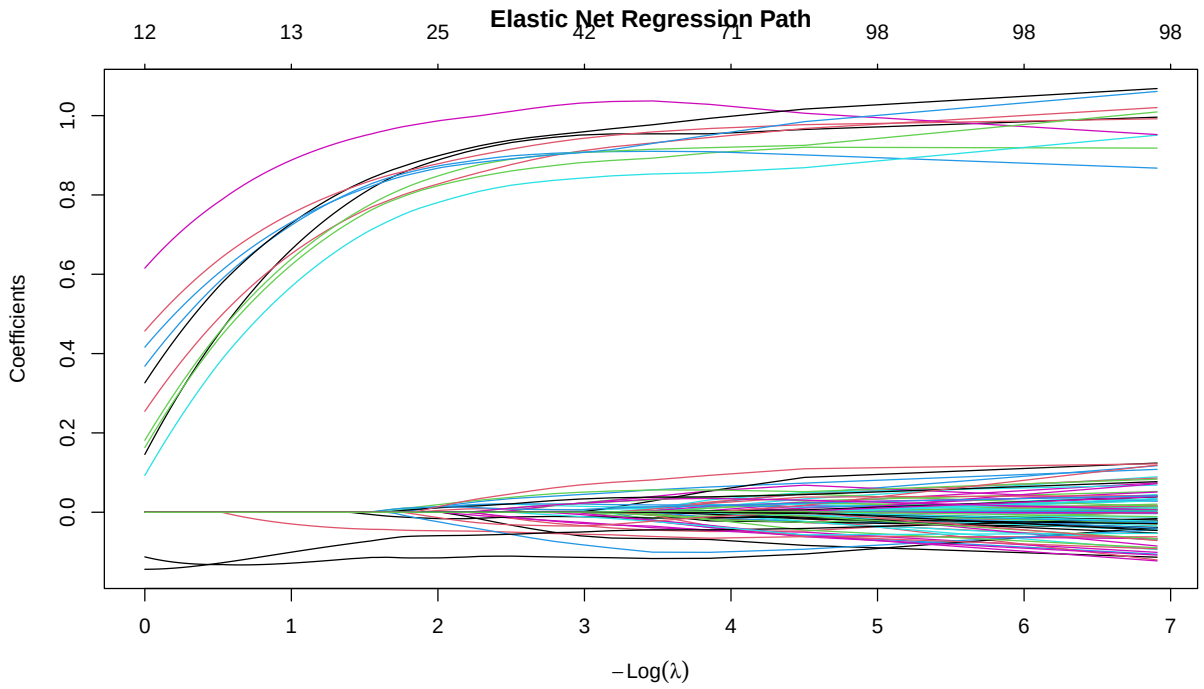
```r
cat("Elastic Net коэффициенты для разных lambda:\n")
```

```
## Elastic Net коэффициенты для разных lambda:
```

```r
for(lambda in lambda_values) {
  coef_elastic <- coef(elastic_model, s = lambda)
  cat("Lambda =", lambda, ":\n")
  print(coef_elastic[1:15])  # Показываем первые 15 коэффициентов
  cat("Количество ненулевых коэффициентов:", sum(coef_elastic != 0), "\n")
  cat("\n")
}
```

```
## Lambda = 0.001 :
##  [1]  0.03806029  0.99605945  1.02021838  0.91806717  0.86758385  0.95064848
##  [7]  0.95214656  1.06819375  0.99199239  1.00926536  1.06091316  0.04446267
## [13]  0.07394140 -0.02493887 -0.06690619
## Количество ненулевых коэффициентов: 99
##
## Lambda = 0.01 :
##  [1]  0.047645311  0.968409216  0.973224058  0.919984464  0.897240940
##  [6]  0.877577702  1.000156757  1.022091921  0.978800447  0.934155753
## [11]  0.992999305  0.004806775  0.028105701 -0.025738785 -0.030235285
## Количество ненулевых коэффициентов: 100
##
## Lambda = 0.1 :
##  [1] 0.02851543 0.91788104 0.85844768 0.84782849 0.88973917 0.81075494
##  [7] 1.00063590 0.92524582 0.90197818 0.87771045 0.88293693 0.00000000
## [13] 0.00000000 0.00000000 0.00000000
## Количество ненулевых коэффициентов: 31
##
## Lambda = 1 :
##  [1] 0.15126157 0.14579238 0.25444066 0.16296115 0.36823938 0.09266221
##  [7] 0.61529468 0.32622875 0.45685013 0.18110942 0.41618497 0.00000000
## [13] 0.00000000 0.00000000 0.00000000
## Количество ненулевых коэффициентов: 13
```

```r
# Визуализация пути коэффициентов
plot(elastic_model, xvar = "lambda", main = "Elastic Net Regression Path")
```



7. Кросс-валидация для Выбора Lambda

7.1. Ridge с Кросс-валидацией

```
# Ridge с кросс-валидацией
ridge_cv <- cv.glmnet(X, y, alpha = 0, nfolds = 10)

# Оптимальное lambda
lambda_min_ridge <- ridge_cv$lambda.min
lambda_1se_ridge <- ridge_cv$lambda.1se

cat("Ridge кросс-валидация:\n")
```

```
## Ridge кросс-валидация:
```

```
cat("Lambda min (минимум ошибки):", round(lambda_min_ridge, 4), "\n")
```

```
## Lambda min (минимум ошибки): 10.395
```

```
cat("Lambda 1SE (в пределах 1 стандартной ошибки):", round(lambda_1se_ridge, 4), "\n")
```

```
## Lambda 1SE (в пределах 1 стандартной ошибки): 31.7448
```

```
# Коэффициенты для оптимального lambda
coef_ridge_optimal <- coef(ridge_cv, s = "lambda.min")
cat("Коэффициенты для lambda.min:\n")
```

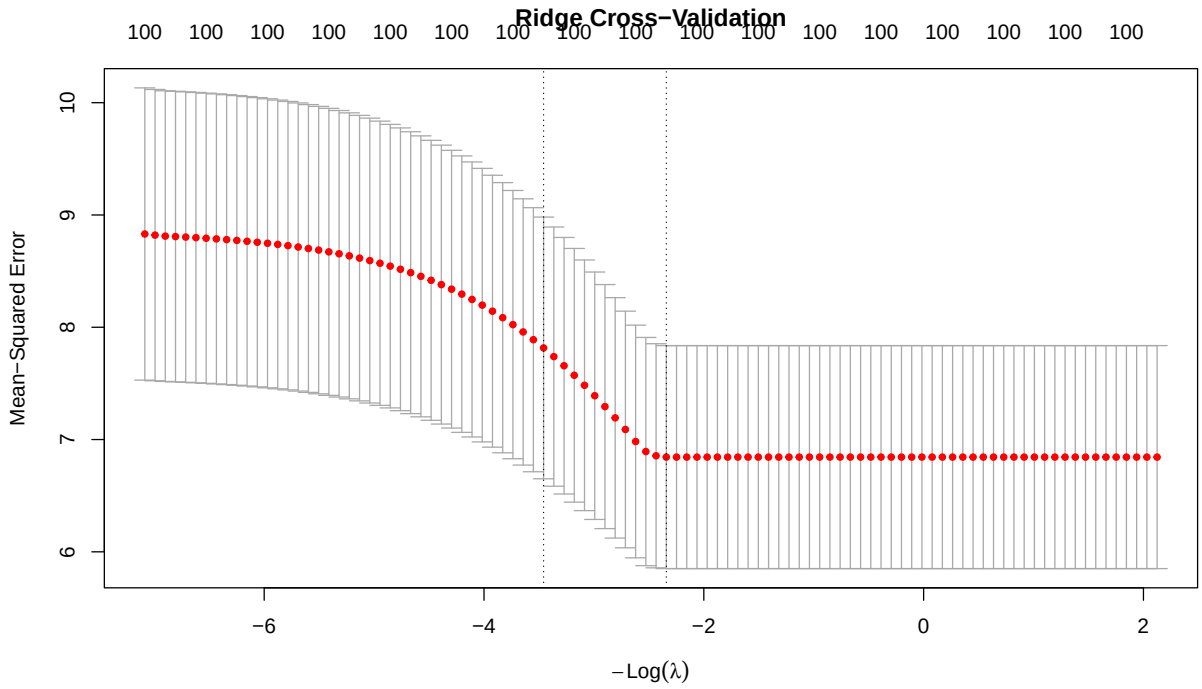
```
## Коэффициенты для lambda.min:
```

```
print(coef_ridge_optimal[1:15])
```

```
##  [1] 0.141080855 0.145552473 0.167375302 0.128555415 0.179297436 0.106432255
```

```
##  [7] 0.241422941 0.163067657 0.231154667 0.121052109 0.195616137 0.058316813
## [13] 0.008968489 0.071420708 0.041761881
```

```r
# График кросс-валидации
plot(ridge_cv, main = "Ridge Cross-Validation")
```



7.2. Lasso с Кросс-валидацией

```r
# Lasso с кросс-валидацией
lasso_cv <- cv.glmnet(X, y, alpha = 1, nfolds = 10)

# Оптимальное lambda
lambda_min_lasso <- lasso_cv$lambda.min
lambda_1se_lasso <- lasso_cv$lambda.1se

cat("Lasso кросс-валидация:\n")
```

```
## Lasso кросс-валидация:
```

```r
cat("Lambda min (минимум ошибки):", round(lambda_min_lasso, 4), "\n")
```

```
## Lambda min (минимум ошибки): 0.0264
```

```r
cat("Lambda 1SE (в пределах 1 стандартной ошибки):", round(lambda_1se_lasso, 4), "\n")
```

```
## Lambda 1SE (в пределах 1 стандартной ошибки): 0.0505
```

```r
# Коэффициенты для оптимального lambda
coef_lasso_optimal <- coef(lasso_cv, s = "lambda.min")
cat("Коэффициенты для lambda.min:\n")
```

```
## Коэффициенты для lambda.min:
```

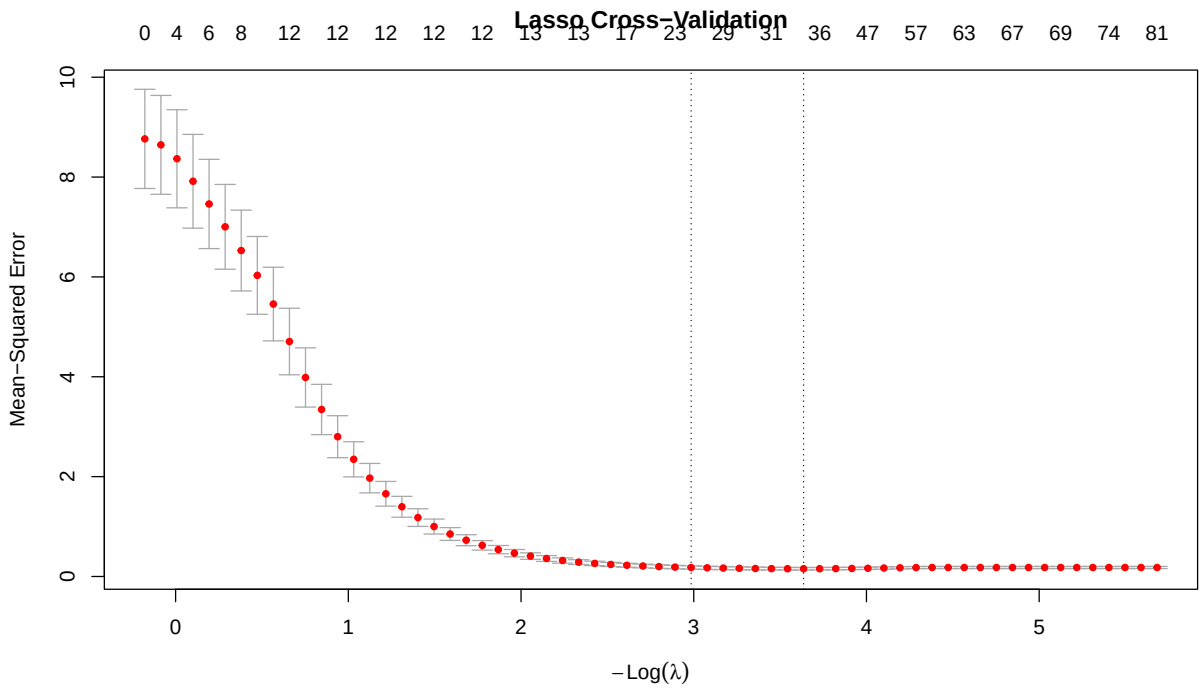
```r
print(coef_lasso_optimal[1:15])
```

```
##  [1] 0.03809335 0.97392173 0.92687702 0.90626516 0.92947750 0.86962873
##  [7] 1.03274379 0.97499637 0.95040843 0.93114371 0.91866286 0.00000000
## [13] 0.00000000 0.00000000 0.00000000
```

```r
cat("Количество ненулевых коэффициентов:", sum(coef_lasso_optimal != 0), "\n")
```

```
## Количество ненулевых коэффициентов: 35
```

```r
# График кросс-валидации
plot(lasso_cv, main = "Lasso Cross-Validation")
```



8. Сравнение Методов

8.1. Сравнение Производительности

```r
# Предсказания для всех методов
pred_ridge <- predict(ridge_cv, X, s = "lambda.min")
pred_lasso <- predict(lasso_cv, X, s = "lambda.min")

# Вычисляем MSE
mse_ridge <- mean((y - pred_ridge)^2)
mse_lasso <- mean((y - pred_lasso)^2)

# Сравнение с истинными коэффициентами
true_beta_with_intercept <- c(0, true_beta)  # Добавляем intercept

# Ridge коэффициенты
ridge_beta <- as.vector(coef_ridge_optimal)
```

```r
mse_beta_ridge <- mean((true_beta_with_intercept - ridge_beta)^2)

# Lasso коэффициенты
lasso_beta <- as.vector(coef_lasso_optimal)
mse_beta_lasso <- mean((true_beta_with_intercept - lasso_beta)^2)

# Таблица сравнения
comparison_table <- data.frame(
  Method = c("Ridge", "Lasso"),
  MSE_Prediction = c(mse_ridge, mse_lasso),
  MSE_Coefficients = c(mse_beta_ridge, mse_beta_lasso),
  Nonzero_Coefficients = c(sum(ridge_beta != 0), sum(lasso_beta != 0)),
  Lambda_Optimal = c(lambda_min_ridge, lambda_min_lasso)
)

cat("Сравнение методов:\n")
```

```
## Сравнение методов:
```

```r
print(comparison_table)
```

```
##   Method MSE_Prediction MSE_Coefficients Nonzero_Coefficients Lambda_Optimal
## 1  Ridge     4.08540642      0.071026661                  101    10.39498972
## 2  Lasso     0.05305382      0.000874672                   35     0.02635509
```

8.2. Визуализация Сравнения

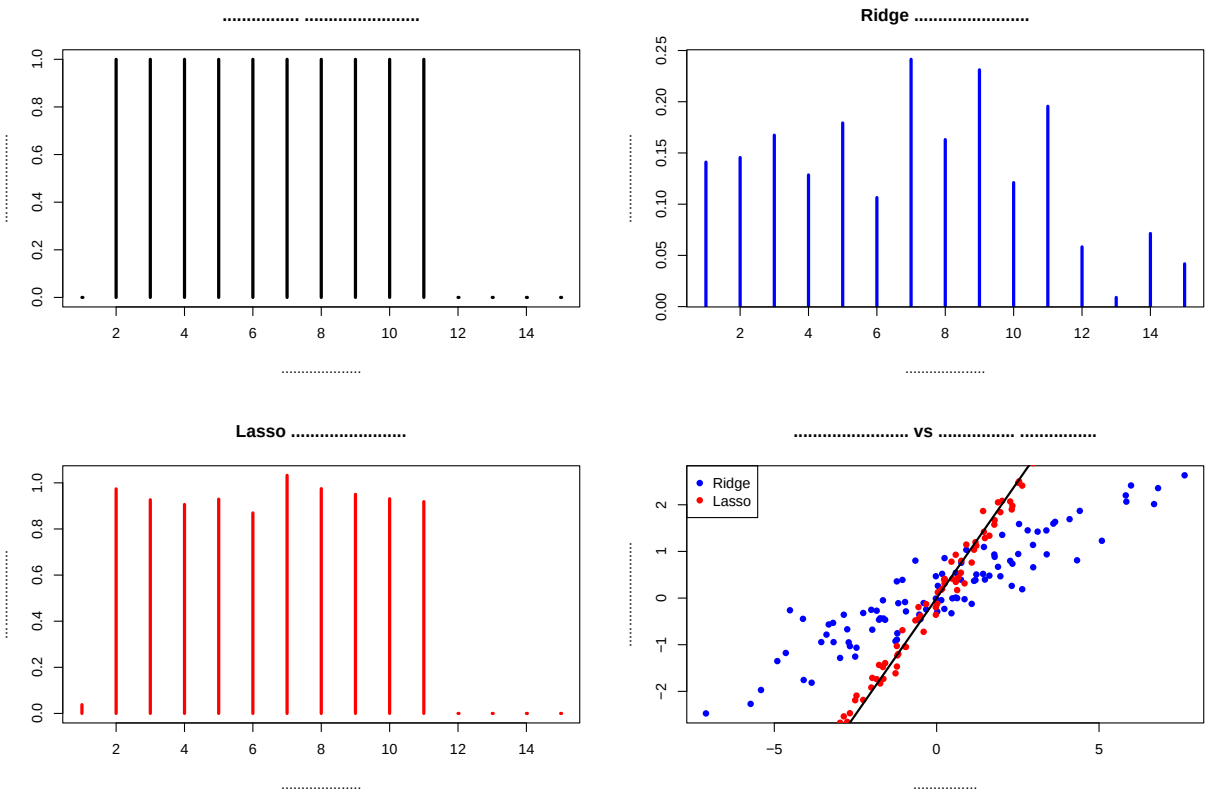
```r
# График сравнения коэффициентов
par(mfrow = c(2, 2))

# Истинные коэффициенты
plot(1:15, true_beta_with_intercept[1:15], type = "h", lwd = 3,
     main = "Истинные коэффициенты", xlab = "Переменная", ylab = "Коэффициент",
     col = "black")

# Ridge коэффициенты
plot(1:15, ridge_beta[1:15], type = "h", lwd = 3,
     main = "Ridge коэффициенты", xlab = "Переменная", ylab = "Коэффициент",
     col = "blue")

# Lasso коэффициенты
plot(1:15, lasso_beta[1:15], type = "h", lwd = 3,
     main = "Lasso коэффициенты", xlab = "Переменная", ylab = "Коэффициент",
     col = "red")

# Сравнение предсказаний
plot(y, pred_ridge, pch = 16, col = "blue", alpha = 0.6,
     main = "Предсказания vs Реальные значения", xlab = "Реальные", ylab = "Предсказанные")
points(y, pred_lasso, pch = 16, col = "red", alpha = 0.6)
abline(0, 1, lwd = 2, col = "black")
legend("topleft", legend = c("Ridge", "Lasso"), col = c("blue", "red"), pch = 16)
```




```
par(mfrow = c(1, 1))
```

9. Интерпретация Результатов

9.1. Что Показывают Результаты

9.1.1. Ridge Регрессия:

- Все переменные остаются в модели
- Коэффициенты сжимаются к нулю
- Хорошо справляется с мультиколлинеарностью
- Стабильные предсказания

9.1.2. Lasso Регрессия:

- Отбирает только важные переменные
- Создает разреженные модели
- Легче интерпретировать
- Хорошо для высокоразмерных данных

9.2. Выбор Между Ridge и Lasso

9.2.1. Используйте Ridge когда:

- Много коррелированных переменных
- Нужны стабильные предсказания
- Все переменные потенциально важны
- Есть мультиколлинеарность

9.2.2. Используйте Lasso когда:

- Нужен отбор признаков
- Высокоразмерные данные (p > n)
- Хотите интерпретируемую модель
- Подозреваете, что многие переменные неважны

9.2.3. Используйте Elastic Net когда:

- Хотите компромисс между Ridge и Lasso
- Есть и коррелированные, и неважные переменные
- Нужен баланс между стабильностью и разреженностью

10. Заключение

10.1. Ключевые Выводы

1. **Регуляризация решает проблему переобучения**
2. **Ridge сжимает коэффициенты, Lasso обнуляет их**
3. **Кросс-валидация помогает выбрать оптимальный lambda**
4. **Выбор метода зависит от задачи и данных**

10.2. Практические Рекомендации

1. **Начните с кросс-валидации** для выбора lambda
2. **Сравните Ridge и Lasso** на ваших данных
3. **Рассмотрите Elastic Net** как компромисс
4. **Интерпретируйте результаты** в контексте задачи

10.3. Дальнейшее Изучение

- **Групповая Lasso** (Group Lasso)
- **Адаптивная Lasso** (Adaptive Lasso)
- **SCAD** (Smoothly Clipped Absolute Deviation)
- **MCP** (Minimax Concave Penalty)

Этот анализ показывает, как Ridge и Lasso решают проблему переобучения в высокоразмерных данных, каждый со своими преимуществами и ограничениями.