МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ (БАЗОВОЙ) ПРАКТИКЕ

студента 3 курса 312 группы
направления 01.03.02 — Прикладная математика и информатика

механико-математического факультета
Георгиева Ивана Владимировича

Место прохождения: механико-математический факультет
Сроки прохождения: с 29.06.2020 г. по 26.07.2020 г.
Оценка:

Руководитель практики от СГУ

к. ф.-м. н., доцент _____ О. А. Мыльцина

Руководитель практики от организации

д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2020

Тема практики:«Перевод фрагмента из книги "Machine Learning with R".»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью настоящей работы является перевод книги Абрахама Гхатака "Machine Learning with R знакомство с машинным обучением, языком программирования R и реализациями некоторых видов регрессий.

Отчет по практике состоит из введения, основной части, содержащей три раздела, и заключения. В первом разделе был переведён параграф 4.7. из книги Гхатака. Во втором - 4.8.В третьем - 4.9. Список использованных источников содержит 5 наименований на которые в тексте приведены ссылки.

1 Гребневая Регрессия

Ранее мы видели, что добавление многочленов более высокой степени к уравнению регрессии приводит к переобучению. Переобучение происходит, когда модель слишком хорошо подходит для обучающих данных и не обобщается на ранее неизветсные данные.

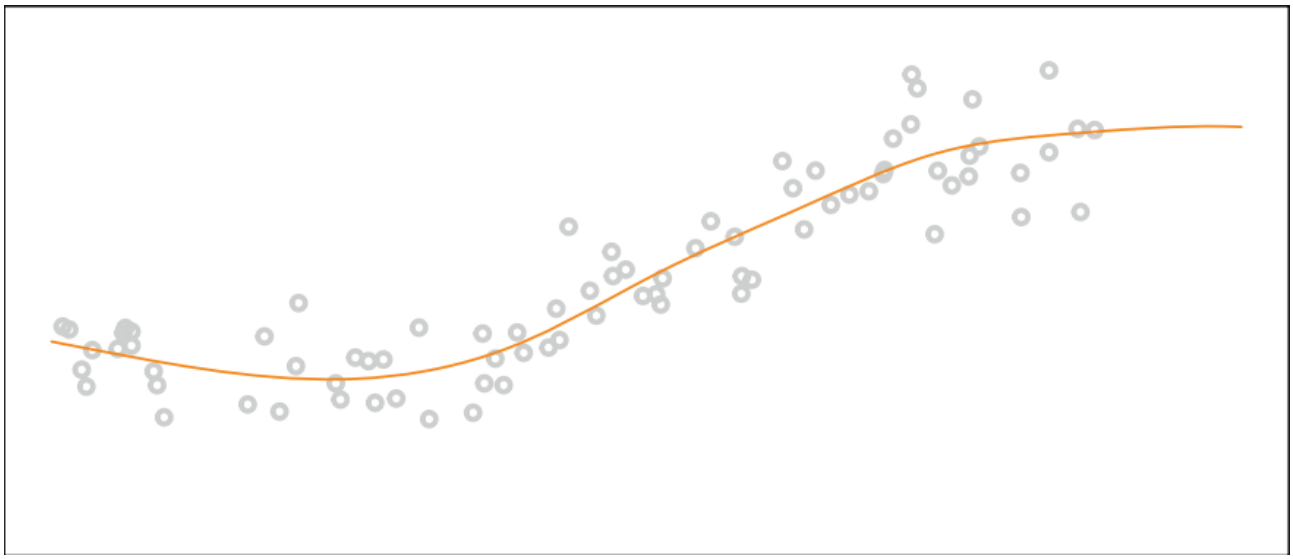

Рисунок 1 – Трудно переобучить из-за множества наблюдений

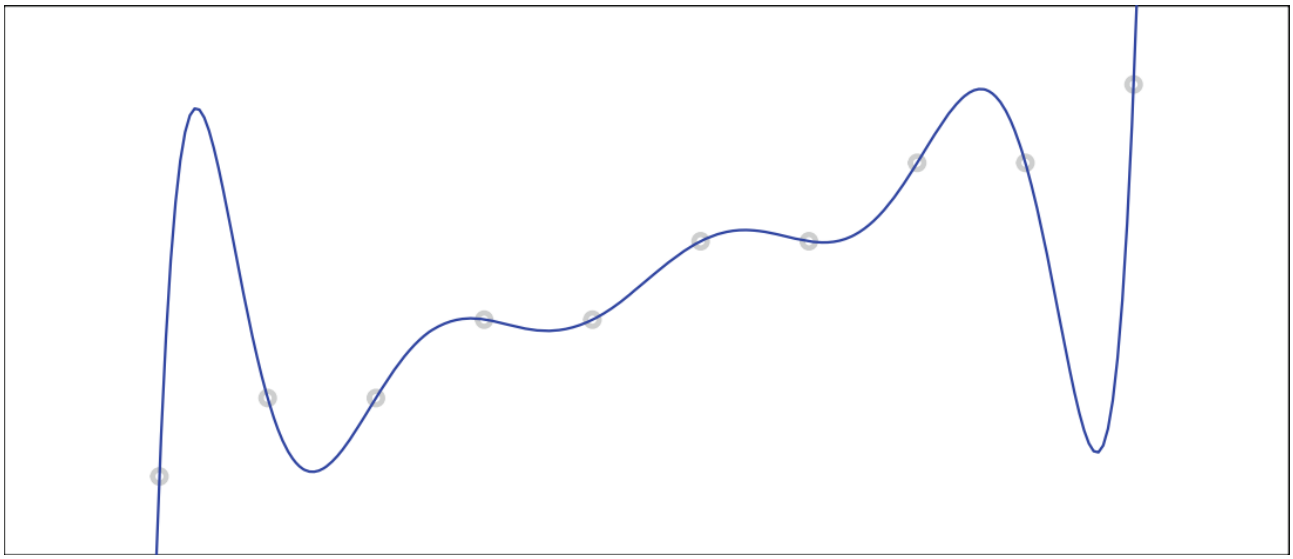

Рисунок 2 – Легко переобучить из-за крайне малого числа наблюдений

Обратимся к рисункам 1 и 2. Переобучение также может произойти, если в уравнении регрессии слишком много независимых переменных или, если наблюдений слишком мало. Переобучение также связано с очень большими оценочными параметрами (весами) $\hat{w}$.

Поэтому мы хотим найти баланс между

— Насколько хорошо наша модель соответствует данным (мерой соответствия)
— Величиной коэффициентов

Таким образом, общая стоимость модели представляет собой комбинацию меры соответствия и величиной коэффициентов. Мера соответствия представлена суммой квадратов разностей(RSS). Небольшая указывает на хорошее соответствие. Мера величины коэффициентов - это сумма абсолютных значений коэффициентов l_1 норма или сумма квадратов значений коэффициентов l_2 норм. Они представлены следующим образом:

$$\|w_0\| + \|w_1\| + \cdots + \|w_n\| = \sum_{j=0}^{n} \|w_j\| = \|w\|_1 \, (l_1 \text{ норма})$$

$$w_0^2 + w_1^2 + \cdots + w_n^2 = \sum_{j=0}^{n} w_j^2 = \|w\|_2^2 \, (l_2 \text{ норма})$$

В гребневой регрессии мы считаем l_2 норму как меру величины коэффициентов. Таким образом, общая стоимость

$$Total\ Cost = RSS(w) + \|w\|_2^2 \tag{1}$$

Наша цель в гребневой регрессии состоит в том, чтобы найти $\hat{w}$, чтобы минимизировать общие затраты в уравнении 1. Баланс между мерой соответствия и величиной коэффициентов достигается путем введения параметра настройки λ так, чтобы

$$Total\ Cost = RSS(w) + \lambda \|w\|_2^2 \tag{2}$$

Если $\lambda = 0 \rightarrow$ сводится к минимизации $RSS(w) \rightarrow \hat{w}^{LeastSquares(LS)}$

Если $\lambda = \infty \rightarrow$ общая стоимость равна ∞ когда $(\hat{w} \neq 0)$ и общая стоимость равна 0 когда $(\hat{w} = 0)$ Если λ между $\rightarrow \|\hat{w}\|_2^2 \leq \left\|\hat{w}^{(LS)}\right\|_2^2$

1.1 Вычисление градиента гребневой регрессии

Закрытая форма решения

$$RSS(w) = (y - \mathbf{H}w)^T(y - \mathbf{H}w)$$

Общая стоимость в случае гребневой регрессии составляет

$$\begin{aligned} Total\ Cost &= (y - \mathbf{H}w)^T(y - \mathbf{H}w) + \lambda \|w\|_2^2 \\ &= (y - \mathbf{H}w)^T(y - \mathbf{H}w) + \lambda w^T w \end{aligned} \tag{3}$$

Градиент уравнения 3

$$\begin{aligned} \Delta \left[RSS(w) + \lambda \|w\|_2^2 \right] &= \Delta(y - \mathbf{H}w)^T(y - \mathbf{H}w) + \lambda \|w\|_2^2 \\ \Delta Cost(w) &= -2\mathbf{H}^t(y - \mathbf{H}w) + \lambda(2w) \\ &= -2\mathbf{H}^t(y - \mathbf{H}w) + 2\lambda I w \end{aligned} \tag{4}$$

Приравнивая уравнение 4 к 0, получаем

$$\begin{aligned} \Delta Cost(w) &= 0 \\ -2\mathbf{H}^T(y - \mathbf{H}w) + 2\lambda I w &= 0 \\ -\mathbf{H}^T y + \mathbf{H}^T\mathbf{H}\hat{w} + \lambda I\hat{w} &= 0 \\ (\mathbf{H}^T\mathbf{H} + \lambda I)\hat{w} &= \mathbf{H}^T y \\ \hat{w}_{ridge} &= (\mathbf{H}^T\mathbf{H} + \lambda I)^{-1}\mathbf{H}^T y \end{aligned} \tag{5}$$

Если $\lambda = 0$:$\rightarrow$ $(\hat{w}_{ridge} = \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T y = \hat{w}^{LS}$

Если $\lambda = 0$:$\rightarrow$ $(\hat{w}_{ridge} = 0$

Тот факт, что мы заботимся о сложности модели, называется регуляризацией (см. предыдущие разделы). Сложность модели определяется параметром настройки λ, который снижает модель (за счет стоимости), если ее сложность увеличивается.

Градиентный спуск Поэлементный алгоритм градиентного спуска гребневой регрессии можно записать как

$$w_j^{(t+1)} \leftarrow w_j^{(t)} + \Delta Cost(w)$$
$$w_j^{(t+1)} \leftarrow w_j^{(t)} - 2\mathbf{H}^T(y - \mathbf{H}w) + 2\lambda w$$
$$w_j^{(t+1)} \leftarrow w_j^{(t)} - \eta \left[-2 \sum_{i=1}^{n} h_j(x_i)(y_i - \hat{y}_i(w^{(t)})) + 2\lambda w_j^{(t)} \right] \tag{6}$$
$$w_j^{(t+1)} \leftarrow w_j^{(t)}(1 - 2\eta\lambda) + 2\eta \sum_{i=1}^{n} h_j(x_i)(y_i - \hat{y}_i(w^{(t)}))$$

В уравнении 6 η - размер шага и выражение $(1 - 2\eta\lambda)$ всегда меньше или равно 1 пока $\eta > 0$, $\lambda > 0$. Выражение $2\eta \sum_{i=1}^{n} h_j(x_i)(y_i - \hat{y}_i(w^{(t)}))$ - коэффициент обновления из RSS

На рисунке 3 показано, что $w_j^{(t)}$ уменьшается в $(1 - 2\eta\lambda)$ на промежуточном этапе, а затем $w_j^{(t+1)}$ увеличивается в коэффициент обновления $2\eta \sum_{i=1}^{n} h_j(x_i)(y_i - \hat{y}_i(w^{(t)}))$

Обобщая,

— Для регрессии методом наименьших квадратов: $w_j^{(t+1)} \leftarrow w_j^{(t)} - \eta*$ (коэффициент обновления)

— Для гребневой регрессии: $w_j^{(t+1)} \leftarrow (1 - 2\eta\lambda)w_j^{(t)} - \eta*$ (коэффициент обновления)

Мы всегда можем лучше подобрать обучающую выборку со сложной моделью и настройкой спада веса $\lambda = 0$, чем мы могли бы с менее сложной моделью и положительным спадом веса. Это подводит нас к вопросу о том, как выбрать параметр настройки λ? Чтобы найти λ, мы используем k-кратную перекрестную проверку (см. предыдущие разделы).

Процесс включает подгонку $\hat{w}_\lambda$ к обучающему набору, тестирование производительности модели с $\hat{w}_\lambda$ на проверочном наборе для выбора λ^* и, наконец, оценку ошибки обобщения модели с $\hat{w}_{\lambda^*}$. Средняя ошибка вычисляется следующим образом

$$\text{Средняя ошибка } CV(\lambda) = \frac{1}{k} \sum_{k=1}^{k} error_k(\lambda) \tag{7}$$

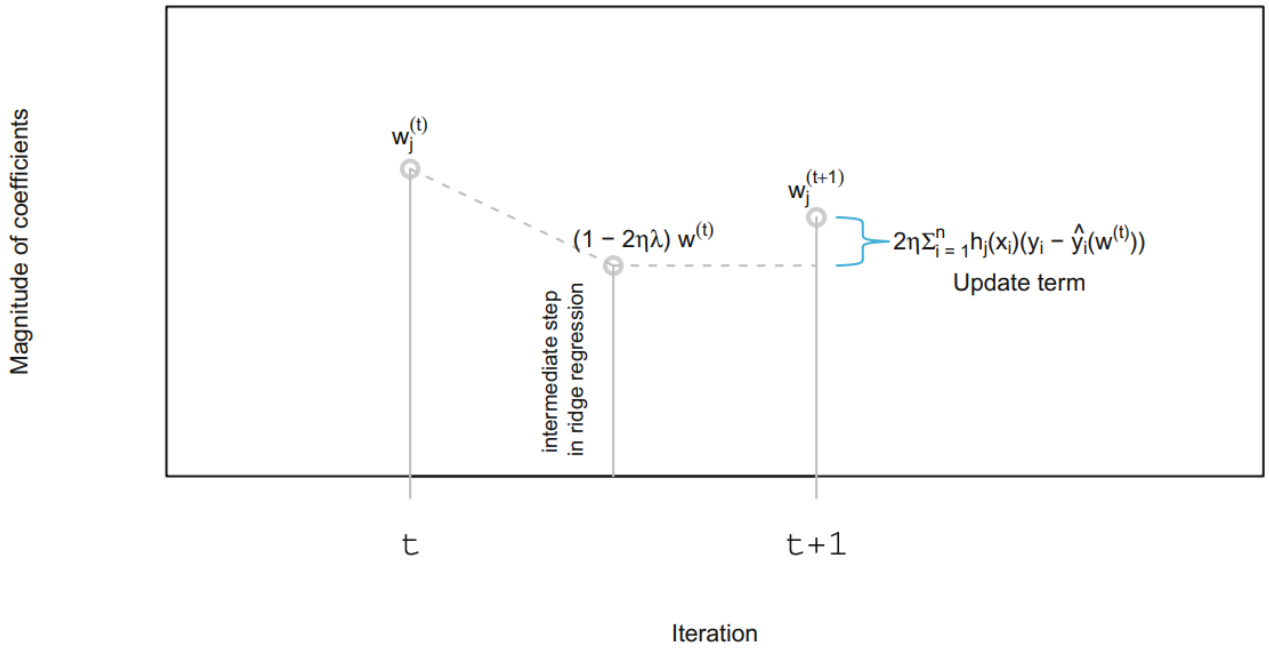

Рисунок 3 – Регуляризация весов при гребневой регрессии

1.2 Написание приложения градиентного спуска гребневой регрессии

Теперь мы напишем наше собственное приложение гребневой регрессии с нуля, и мы будем использовать *kc_house_data*, который использовался ранее. Давайте сначала воспользуемся функцией "lm"в R и извлечем коэффициенты, регрессируя *price* с *sqft_living* и *bedrooms*, из *house_train_data*.

```
lm(price ~ sqft_living + bedrooms, data=house_train_data)$coef
```

(Intercept)	sqft_living	bedrooms
97050.0942	305.2103	-57429.9302

Первая функция в нашем приложении градиентного спуска гребневой регрессии - это создание матрицы данных.

```
data_matrix <- function(data, features, output){
  scaled_feature_matrix <-
    data.frame(scale(data[features]),
      row.names=NULL)
  length <- nrow(data)
  scaled_features <- as.matrix(cbind('Intercept' =
      rep(1, length),
    scaled_feature_matrix[, features]))
  output <- as.matrix(scale(data[output]))
```

```
  return(list(scaled_features, output))
}
```

Функция predict_output предсказывает целевые значения.

```
predict_output<-function(feature_matrix, weights){
  predictions=(feature_matrix) %*% (weights)
  return(predictions)
}
```

Штраф l_2 получил свое название, потому что он заставляет веса иметь меньшую l_2 норму, чем в противном случае. Следующая функция принимает начальные веса и матрицу признаков и предсказывает целевые значения с помощью функции predict_output. errors находятся по разнице в предсказанных и фактических значениях. Производная или gradient вычисляется с использованием матрицы признаков и errors ($\Delta cost(w) = 2\mathbf{H}^T(y - \mathbf{H}w) + 2\lambda w$). Веса обновляются путем вычитания произведения gradient и step_size (см. уравнение 6. Норма градиентов рассчитывается (см. уравнение выше), чтобы проверить, меньше ли она tolerance в цикле «while».

```
ridge_regression_gradient_descent =
      function(feature_matrix, output,
    initial_weights, step_size, l2_penalty, tolerance) {
    converged = FALSE
    weights = matrix(initial_weights)
    j = 0
    while (!converged) {
        predictions = predict_output(feature_matrix, weights)
        errors = predictions - output
        # Loop over each feature weight
        for (i in 1:length(weights)) {
            if (i == 1) {
                gradient =
      2 * (feature_matrix[, i] %*% errors)
                gradient_norm = sqrt(sum(gradient^2))
                weights[1, 1] =
      weights[1, 1] - (step_size * gradient)
            } else {
                gradient =
      2 * (feature_matrix[, i] %*% errors) +
                  2 * (l2_penalty * weights[i, 1])
                gradient_norm = sqrt(sum(gradient^2))
                if (gradient_norm < tolerance) {
                    converged = TRUE
                }
```

```
                weights[i, 1] =
      weights[i, 1] - (step_size * gradient)
            }
        }
        j = j + 1
    }
    print(paste0("Gradient descent converged at iteration:",
  j - 1))
    return(weights)
}
```

Мы присвоим два разных значения параметру настройки $\lambda - 0$ и ∞ и исследуем, как параметры (веса) корректируются. Мы рассмотрим модель с двумя функциями, то есть sqft_living и bedrooms.

```
features <- c("sqft_living", "bedrooms")
target <- "price"
train_data <- house_train_data[, c(features, target)]
feature_matrix = data_matrix(train_data,
    features,
    target)[[1]]
output_matrix = data_matrix(train_data,
    features,
    target)[[2]]
initial_weights = c(0, 0, 0)
step_size = 1e-6
tolerance = 1e-8
weights_with_0_penalty =
    ridge_regression_gradient_descent(feature_matrix,
        output_matrix,
        initial_weights,
        step_size,
        l2_penalty = 0,
        tolerance)
```

[1] "Gradient descent converged at iteration: 3002"

```
print(weights_with_0_penalty)
```

$$[, 1]$$
$$[1,] - 5.560047e - 17$$
$$[2,] 7.885384e - 01$$
$$[3,] - 1.489809e - 01$$

Коэффициенты с $\lambda = 0$ такие же, как для OLS регрессии. Давайте проверим, получим ли мы те же результаты, используя функцию «lm» в R.

```
train_data_features <-
  data.frame(scale(house_train_data[, features]))
train_data_output <-
  data.frame(scale(house_train_data[, target]))
train_data <-
  cbind(train_data_features, train_data_output)
colnames(train_data) <-
  c("sqft_living", "bedrooms", "price")
(OLS_coef <-
  lm(price ~ sqft_living + bedrooms, data =
    train_data)$coef)
```

(Intercept)	sqft_living	bedrooms
2.600634e-17	7.885384e-01	-1.489809e-01

Действительно получим! R имеет функцию «glmnet» в пакете glmnet. Характеристики и результат принадлежат классу matrix. Аргумент «alpha» равен 0 для гребневой регрессии гребня и 1 для регрессии лассо. Допуск, определенный в нашем алгоритме, - это «порог», который по умолчанию равен 1e-07. Есть много других параметров, которые можно изучить, набрав help(glmnet). Давайте проверим наши результаты с помощью функции «glmnet» в R (Таблица 1).

```
library(glmnet)
ridge_model_lambda_0 <- glmnet(x = feature_matrix,
  y = output_matrix,
  alpha = 0,
  lambda = 0,
```

Таблица 1 – Сравнение коэффициентов гребневой регрессии с $\lambda = 0$ для нашего приложения, модели регрессии OLS и glmnet

	Application	OLS	glmnet
(Intercept)	0.00000	0.00000	0.00000
sqft_living	0.78854	0.78854	0.78853
bedrooms	-0.14898	-0.14898	-0.14898

Таблица 2 – Сравнение коэффициентов гребневой регрессии с $\lambda = 1e + 5$ для нашего приложения и glmnet

	Application	glmnet
(Intercept)	0.00000	0e+00
sqft_living	0.06679	1e-05
bedrooms	0.02696	0e+00

```
  thresh=1e-8)
glm_ridge_0_coef <- coef(ridge_model_lambda_0)[, 1]
```

Действительно, наше приложение хорошо сравнивает! Теперь давайте попробуем наше приложение с $\lambda = 1e + 05$

```
initial_weights = c(0, 0, 0)
step_size = 1e-6
tolerance = 1e-8
weights_with_high_penalty =
    ridge_regression_gradient_descent(feature_matrix,
        output_matrix,
        initial_weights,
        step_size,
        l2_penalty = 1e+05,
        tolerance)
```

Эта модель имеет очень низкую предвзятость. Давайте проверим наши результаты с помощью функции glmnet в R (таблица 2).

```
ridge_model_lambda_1e5 <- glmnet(x = feature_matrix,
  y = output_matrix,
  alpha = 0,
  lambda = 1e+5,
  thresh=1e-8)
glm_ridge_1e5_coef <- coef(ridge_model_lambda_1e5)[, 1]
```

Действительно, мы получаем одинаковые результаты, т.е. все коэффициенты очень близки к 0! Представим себе уравнения гребневой регрессии для двух значений λ на рис. 4. Модель гребневой регрессии с градиентным спуском переходит в нулевую модель с отсечением 0 (высокое смещение), когда $\lambda = \infty$, и модель регрессии LS, когда $\lambda = 0$. Наша следующая цель - найти наилучшее значение λ; и мы будем использовать перекрестную проверку, как описано в предыдущих разделах.

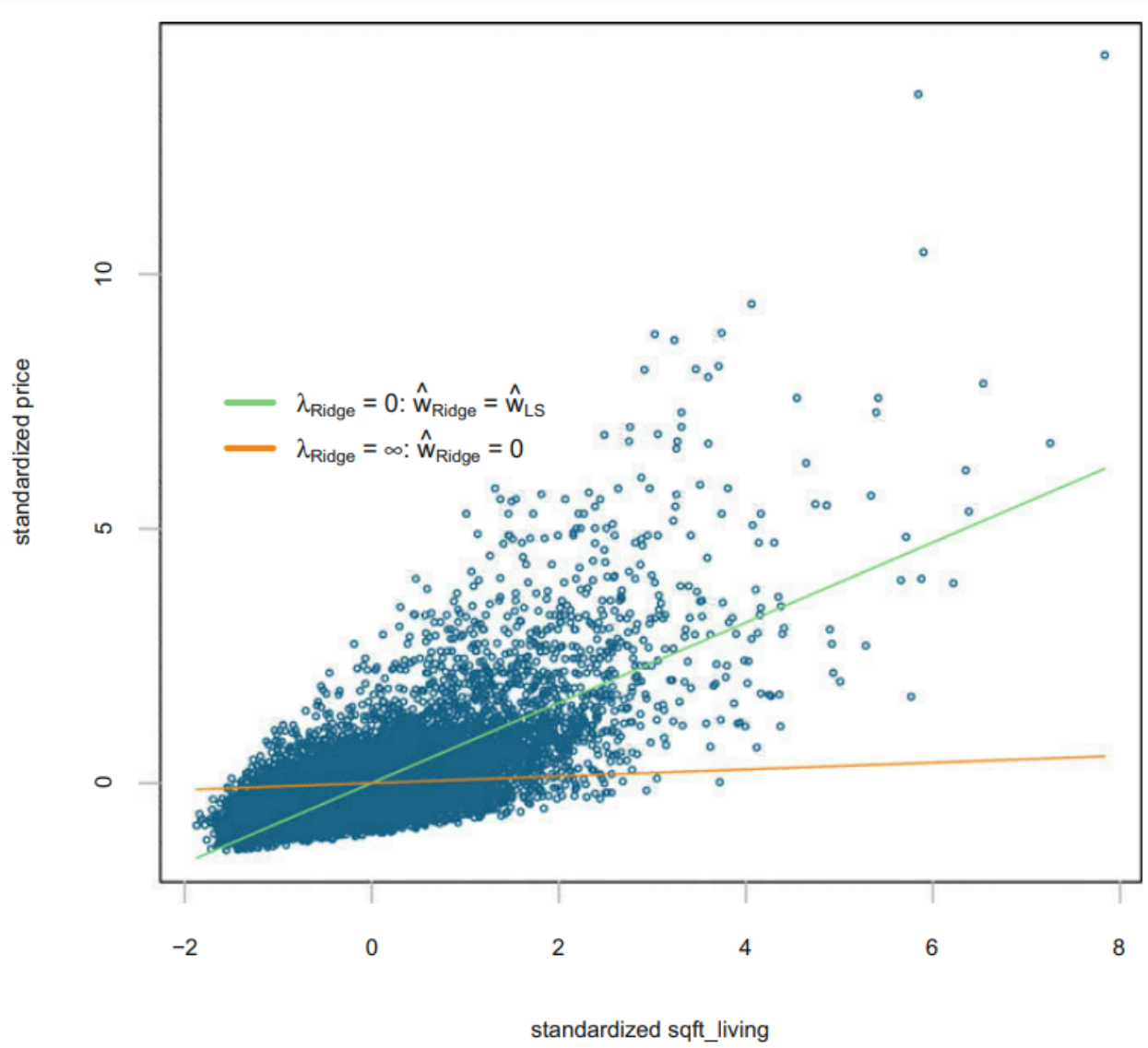

Рисунок 4 – Гребневая регрессия с $\lambda = 0$ и $\lambda = 1e + 05$

2 Оценка производительности

Мы говорили о стоимости модели в разных разделах. По сути, стоимость модели - это мера ошибок модели, то есть того, насколько точно модель может предсказать фактическое значение. То есть, если модель не допускает ошибок прогнозирования, стоимость, приписываемая модели, равна нулю, и, следовательно, ее мера точности составляет 100%. Ранее мы видели, что стоимость регрессионной модели измеряется функцией потерь, определяемой как

$$\begin{gathered} Loss(y, f_{\hat{w}}(x)) = (y - f_{\hat{w}}(x))^2 - \text{ошибка в квадрате} \\ Loss(y, f_{\hat{w}}(x)) = |y - f_{\hat{w}}(x)| - \text{модуль ошибки} \end{gathered} \tag{8}$$

О мере точности модели можно судить по стоимости модели на данных обучения и невидимых данных. Мы видели в предыдущих разделах, что при обучении модели модель пытается соответствовать обучающим данным. Следовательно, с увеличением сложности модели ошибка обучения уменьшается. Очень низкая ошибка обучения не является хорошим показателем точности модели, если обучающие данные не включают «вселенную», то есть «видимые» и «невидимые» данные. Следовательно, эффективность модели можно оценить только по степени ее точности с «невидимыми» данными. **Ошибка обобщения** - это ошибка модели из «вселенной» «невидимых» данных. **Истинная ошибка** - это ошибка из доступного подмножества, из «вселенной» «невидимых» данных. Обсуждаемые здесь ошибки модели зависят от сложности и других алгоритмических характеристик модели. Но в самих данных могут быть ошибки. В следующем разделе мы попытаемся выяснить, как мы можем решить эту проблему.

2.1 Еще раз об источниках ошибок

Мы обсудили, как ошибки могут возникать из данных обучения модели и из невидимых данных. Однако существуют также три других источника ошибок в модели, как описано в одном из предыдущих разделов - шум, смещение и дисперсия. **Шум** всегда проникает в данные, потому что мы не можем точно записать то же самое, скажем, «человеческие эмоции», «проблемы во взаимоотношениях», «глобальные проблемы» и т. Д. На основании имеющих-

ся данных в наши модели также проникают **смещения** и **отклонения**. Если мы обучим модель на основе обучающих данных, в которых нет шума(noise), и оценим ту же модель на невидимых данных, которые содержат шум, мы получим модель с очень низкой точностью. Следовательно, как описано в одном из предыдущих разделов, ошибка прогнозирования в модели состоит из трех компонентов: шума, смещения и дисперсии, которые можно записать как

$$\begin{aligned} PredictionError &= \varsigma^2 + MSE\left[f_{\hat{w}}(x)\right] \\ &= \varsigma^2 + \left[bias(f_{\hat{w}}(x))\right]^2 + var(f_{\hat{w}}(x)) \end{aligned} \tag{9}$$

Подходящая модель выбирается путем выбора параметра настройки λ, который управляет сложностью модели. Это делается с помощью перекрестной проверки. Выбрав подходящую модель, оцениваем ошибку обобщения. Давайте рассмотрим house_train_data и house_test_data, которые мы разделили из набора данных kc_house. Наборы данных для обучения и тестирования имеют 10 807 и 10 806 наблюдения соответственно.

```
train_features =
  as.matrix(data.frame(scale(house_train_data[,
features])))
train_output =
  as.matrix(data.frame(scale(house_train_data[,
target])))
test_features =
  as.matrix(data.frame(scale(house_test_data[,
features])))
test_output =
  as.matrix(data.frame(scale(house_test_data[, target])))
```

Мы обучаем модель регрессии гребня с $\lambda = 0$, используя наши обучающие данные, т. е. реплицируем OLS регрессию и используем нашу модель для прогнозирования невидимых тестовых данных с $\lambda = 10$.

```
ridge_model_train <- glmnet(x = train_features,
  y = train_output,
  alpha=0,
  lambda = 0)
ridge_predict = predict(ridge_model_train,
  newx = test_features,
  s = 10,
  exact = TRUE)
```

```
MSE_lambda_10 = mean((ridge_predict - test_output)^2)
```

[1] "MSE_lambda_10 = 0.903155"

Среднеквадратичная ошибка для этой модели составляет 0.903155. R имеет встроенную функцию перекрестной проверки cv.glmnet(). По умолчанию функция выполняет 10-кратную перекрестную проверку, но это можно изменить с помощью аргумента «nfolds». Мы выполняем 10-кратную перекрестную проверку данных обучения и находим значение λ, для которого ошибка перекрестной проверки наименьшая.

```
cv_ridge = cv.glmnet(x = train_features,
  y = train_output,
  alpha=0)
ideal_lambda=min(cv_ridge$lambda)
```

[1] "ideal_lambda = 0.076901"

```
ridge_predict = predict(ridge_model_train,
  s=ideal_lambda,
  newx = test_features,
  exact = TRUE)
MSE_ideal_lambda =
  mean((ridge_predict - test_output)^2)
```

[1] "MSE with ideal_lambda = 0.495869"

Среднеквадратичная ошибка при $\lambda = 0.076901$ составляет 0.495869, т. е. MSE имеет упала на 45%. Выясним коэффициенты этой модели

```
ridge_train = glmnet(x =
  train_features, y = train_output, alpha = 0)
ridge_ideal_lambda_coef <-
    predict(ridge_train, type = "coefficients",
  s = ideal_lambda)
ridge_ideal_lambda_coef[, 1]
```

(Intercept)	sqft_living	bedrooms
-1.397232e-16	6.998124e-01	-8.978171e-02

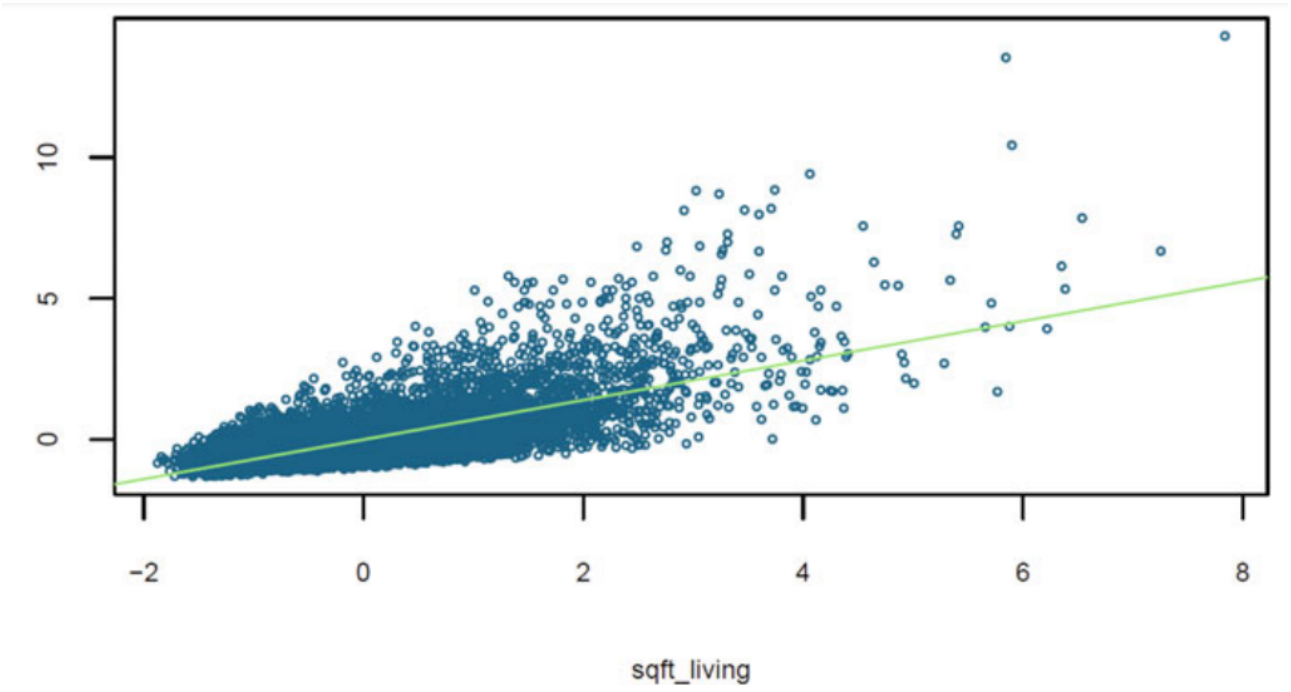

Рисунок 5 – Гребневая регрессия с $\lambda = 0.076901$ полученное с перекрестной проверкой

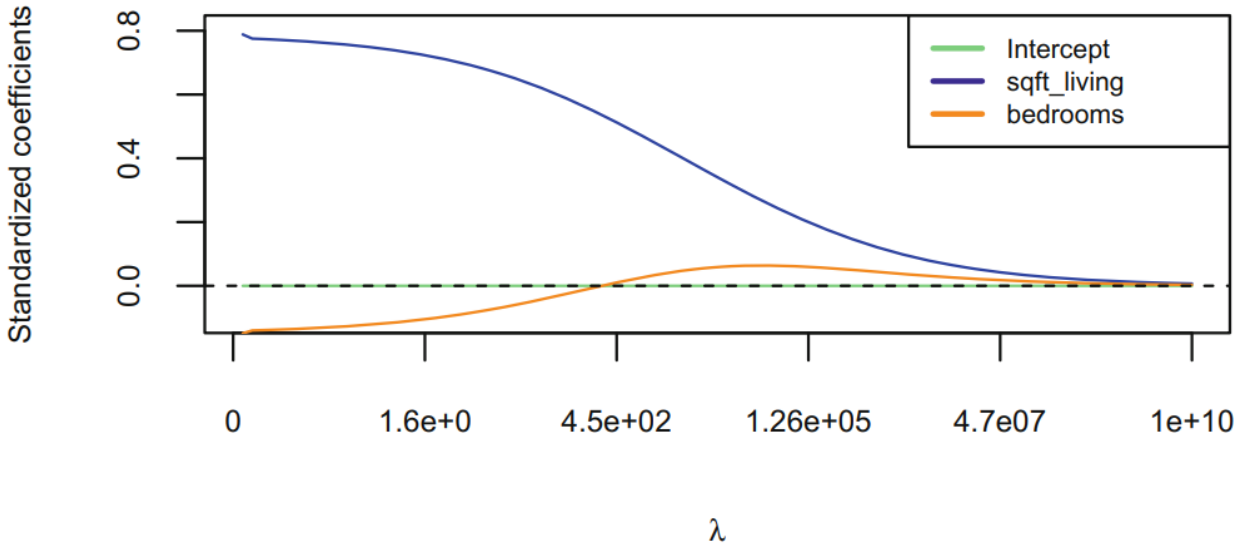

Рисунок 6 – Стандартизированные коэффициенты регрессии Риджа при изменении λ. Коэффициенты не падают резко до 0 при $\lambda \to \infty$

Давайте визуализировать это уравнение гребневой регрессии с помощью предиктора sqft_living. На рис. 5 представлено уравнение регуляризованной гребневой регрессии.

Представим себе, как меняются стандартизированные коэффициенты при разных значениях λ. На рис. 6 видно, что коэффициенты меняются постепенно от $\lambda = 0$ до $\lambda \to \infty$. Он никогда резко не уменьшается до 0.

2.2 Компромисс смещения – дисперсии в гребневой регрессии

Большое значение λ приведет к высокому смещению (простая модель) и, следовательно, к низкой дисперсии ($\hat{w} = 0$, для $\lambda = \infty$). Небольшое значение λ приведет к низкому смещению и, как следствие, большой дисперсии.

3 Лассо-регрессия

Если у нас есть «широкий» набор функций (скажем, $1e + 10$), гребневая регуляризация может создать вычислительные проблемы, поскольку выбраны все функции. Алгоритм лассо отбрасывает менее важные / избыточные характеристики, переводя их коэффициенты в ноль. Это позволяет нам интерпретировать функции, а также сокращает время вычислений. Лассо (регуляризованная регрессия l_1) использует норму l_1 в качестве штрафа, вместо нормы l_2 в гребневой регрессии. Прежде чем мы перейдем к целевой функции лассо, давайте вернемся к алгоритму гребневой регрессии. Гребневая регрессия выбирает параметры β с минимальной RSS при условии, что норма l_2 параметров $\beta_1^2 + \beta_2^2 + \cdots + \beta_n^2 \leq t$, ограничение гребня. На рисунке 7 показаны изолинии RSS и ограничения параметров для гребневой регрессии и регрессии лассо. Контуры RSS представляют собой эллипсы, центрированные по методу наименьших квадратов (точка красного цвета). Ограничение построено для различных значений параметров β. В случае гребневой регрессии $\beta_1^2 + \beta_2^2 \leq t$, ограничение гребня. Ограничение принимает форму круга для двух параметров и становится сферой с большим количеством параметров. Первая точка соприкосновения контура RSS с окружностью - это точка, описывающая параметры гребня β_1 и β_2. Значения β в большинстве случаев оказываются ненулевыми. Если t мало, параметры будут маленькими, а если велико, будет стремиться к решению по методу наименьших квадратов.

В случае регрессии лассо параметры β выбираются такими, что $|\beta_1| + |\beta_2| \leq t$, ограничение лассо, для минимального RSS. Как показано на правом графике рис. 7 для регрессии лассо с двумя параметрами, ограничение имеет форму ромба с четырьмя углами; для более чем двух элементов контур становится ромбовидным с множеством углов. Если контур RSS касается угла, он заставляет одну из β стать нулевой. Мы можем переписать общую стоимость в формуле 3, для регрессии лассо как

$$Total\ Cost = RSS(w) + \lambda \left\| w \right\|_1 \tag{10}$$

Если $\lambda = 0 \rightarrow \hat{w}^{lasso} = \hat{w}^{LeastSquares}$ Если $\lambda = \infty \rightarrow \hat{w}^{lasso} = 0$ Если λ между $\rightarrow 0 \leq \hat{w}^{lasso} \leq \hat{w}^{LeastSquares}$

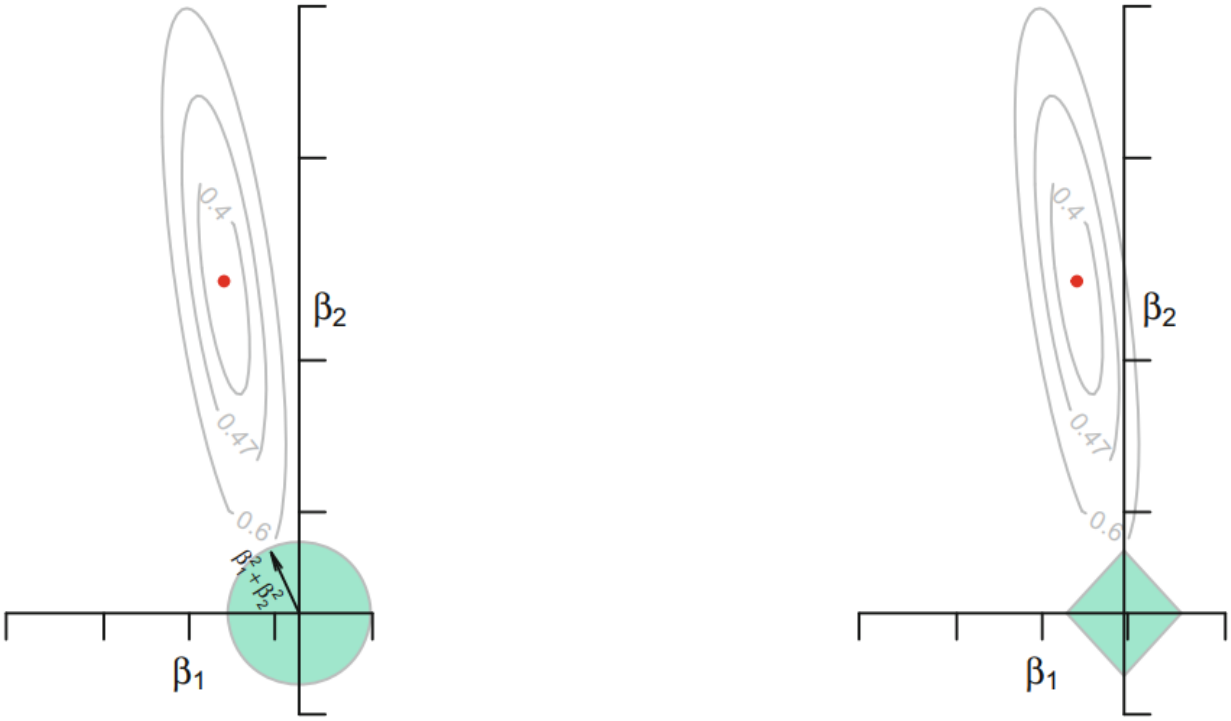

Рисунок 7 – Левый график показывает контуры RSS гребневой регрессии и ограничение на коэффициенты гребня $\beta_1^2 + \beta_2^2 \leq t$ (ограничение гребня). Правый график показывает RSS-контуры регрессии лассо и ограничение на коэффициенты лассо $|\beta_1| + |\beta_2| \leq t$ (ограничение лассо)

Лассо выбирает параметры β с минимальным RSS, при условии, что l_1 норма параметров $(|\beta_1| + |\beta_2| + \cdots + |\beta_n|) \leq tolerance$. Ранее мы видели, что оптимальное решение задачи минимизации лассо находится в начале координат, и поэтому мы не можем вычислить градиент. Поэтому решением является выпуклый алгоритм оптимизации, называемый Координатным спуском. Этот алгоритм пытается минимизировать

$$\begin{gathered} f(w) = f(w_0, w_1, \cdots, w_n) \\ find \underset{min}{f}(w) \end{gathered} \tag{11}$$

Алгоритм координатного спуска можно описать следующим образом:

$$\begin{gathered} \text{Initialize } \hat{w} \\ \text{while not converged, pick a coordinate j} \\ \hat{w}_j \leftarrow \underset{min}{f}(\hat{w}_0.\hat{w}_1, \cdots, w, \hat{w}_{j+1}, \cdots, \hat{w}_n) \end{gathered} \tag{12}$$

Если мы выберем следующую координату случайным образом, это станет стохастическим координатным спуском. При координатном спуске нам не нужно выбирать размер шага.

3.1 Координатный спуск для регрессии наименьших квадратов

Ниже приведен алгоритм спуска координат для LS, по одной координате за раз.

$$RSS(w) = \sum_{i=1}^{n}(y_i - \sum_{j=0}^{n} w_j h_j(x_i))^2$$

Зафиксируем все координаты w_{-j} и возьмем частную производную по w_j

$$\frac{\partial}{\partial w_j} = -2\sum_{i=1}^{n} h_j(x_i)(y_i - \sum_{j=0}^{n} w_j h_j(x_i))^2$$

$$= -2\sum_{i=1}^{n} h_j(x_i)(y_i - \sum_{k\neq j} w_k h_k(x_i) - w_j h_j(x_i))$$

$$= -2\sum_{i=1}^{m} h_j(x_i)(y_i - \sum_{k\neq j} w_k h_k(x_i)) + 2w_j \sum_{i=1}^{n} h_j(x_i)^2$$

Пояснения:

(i) по определению нормированных функций, $\sum_{i=1}^{n} h_j(x_i)^2 = 1$

(ii) мы будем обозначать$(\sum_{i=1}^{n} h_j(x_i)(y_i - \sum_{k\neq j} w_k h_k(x_i)) = \rho_j$

$$= -2\rho + 2w_j$$

приравняв частичные производные к 0, получим

$$w_j = \rho_j \tag{13}$$

3.2 Координатный спуск для регрессии лассо

Ниже приведен псевдокод спуска координат для регрессии лассо, по одной координате за раз.

$$
\begin{gathered}
\text{Initialize } \hat{w} \\
\text{while not converged} \\
\text{for j in } 0, 1, 2, \cdots, n \\
\text{compute: } \rho_j = \sum_{i=1}^{n} h_j(x_i)(y_i ? \hat{y}_i(\hat{w}_j)) \\
\text{set: } w_j = \begin{cases} \rho_j + \frac{\lambda}{2} \text{ if } \rho_j < -\frac{\lambda}{2} \\ 0 \text{ if } \rho_j \text{ in } \left[-\frac{\lambda}{2}, \frac{\lambda}{2}\right] \\ \rho_j - \frac{\lambda}{2} \text{ if } \rho_j > \frac{\lambda}{2} \end{cases}
\end{gathered} \tag{14}
$$

3.3 Написание приложения для координатного спуска регрессии лассо

Теперь мы готовы написать алгоритм регрессии лассо. Функция get_features подбирает данные на основе выбранных функций.

```
get_features <- function(data, features){
  selected_features <- matrix(nrow = nrow(data), ncol = length(features))
  for (i in 1: length(features)){
    selected_features[,i] <- as.numeric(data[[features[[i]]]])
  }
  selected_features
}
```

Функция add_constant_term добавляет столбец единиц к матрице признаков.

```
add_constant_term <- function(data_features){
  length <- nrow(data_features)
  combined <- cbind(rep(1, length), data_features)
  combined
}
```

Функция construct_matrix принимает имена функций и выходные данные из data.frame. Затем он выбирает значения функций и выходные значения, вызывая функцию get_features, и возвращает их в виде списка.

```
construct_features_matrix <- function(features, outputs, data) {
  features <- as.list(features)
  data_features <- get_features(data, features)
  output_data <- get_features(data, outputs)
  features_data <- add_constant_term(data_features)
```

```
  list(features_data, output_data)
}
```

Функция normalize_features вычисляет норму для каждой функции и нормализует feature_matrix.

```
normalize_features <- function(feature_Matrix) {
  norms <- as.numeric(vector(length = ncol(feature_Matrix)))
  normalized_features <- matrix(nrow = nrow(feature_Matrix),
    ncol = ncol(feature_Matrix))
  for (i in 1:ncol(feature_Matrix)) {
    v <- feature_Matrix[, i]
    norms[i] <- sqrt(sum(v^2))
    normalized_features[, i] <- feature_Matrix[, i]/norms[i]
  }
  list(normalized_features, norms)
}
```

Функция predict_output предсказывает выходные значения.

```
predict_output <- function(feature_matrix, weights){
  predictions<-feature_matrix[[1]]%*%weights
  predictions
}
```

В уравнении 13 было показано, что $\rho_j = \sum_{i=1}^{n} h_j(x_i)(y_i - \sum_{k \neq j} w_k h_k(x_i))$. Для понимания последующего кода, в котором мы вычисляем ρ, давайте рассмотрим набор данных с 3 функциями, и мы хотим найти ρ_2. Это можно записать как

$$\begin{aligned} \rho_2 &= h_2(x_2)\left[(y_1 + y_2 + y_3) - (\hat{w}_1 h_1(x_1) + \hat{w}_3 h_3(x_3))\right] \\ &= h_2(x_2)\left[(y_1 + y_2 + y_3) - (\hat{y}_1 + \hat{y}_3)\right] \\ &= h_2(x_2)\left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) + (y_3 - \hat{y}_3) + \hat{y}_2\right] \\ &= h_2(x_2)\left[(target - predicted) + \hat{w}_2 h_2(x_2)\right] \end{aligned} \tag{15}$$

```
get_ro <- function(feature_matrix, output, weights, i) {
  prediction = predict_output(feature_matrix, weights)
  feature_i = feature_matrix[[1]][, i]
  ro_i = sum(feature_i * (output - prediction + weights[i] * feature_i))
  return(ro_i)
}
```

Теперь мы реализуем функцию координатного спуска, которая минимизирует функцию стоимости для одного признака i. Функция возвращает

новый вес для признака i.

```
lasso_coordinate_descent_step <- function(i, data_matrix, weights,
  l1_penalty) {
  normalized_features <- normalize_features(feature_Matrix = data_matrix
    [[1]])
  rho <- get_ro(normalized_features, data_matrix[[2]], weights,
    i)
  if (i == 1) {
    new_weight_i <- rho
  } else if (rho < -l1_penalty/2) {
    new_weight_i <- rho + l1_penalty/2
  } else if (rho > l1_penalty/2) {
    new_weight_i <- rho - l1_penalty/2
  } else {
    new_weight_i <- 0
  }
  return(new_weight_i)
}
```

Теперь, когда у нас есть функция, которая оптимизирует функцию стоимости по одной координате, давайте реализуем циклический спуск координат, где мы оптимизируем координаты от 1 до n по порядку и повторяем.

```
lasso_cyclical_coordinate_descent <- function(data_matrix, initial_weights,
  l1_penalty, tolerance) {
  normalized_features <- normalize_features(feature_Matrix = data_matrix
    [[1]])
  norms <- normalized_features[[2]]
  converged <- FALSE
  weights <- initial_weights
  new_weights <- vector(length = length(weights))
  while (!converged) {
    old_weights <- weights
    for (i in 1:length(weights)) {
      weights[i] <- lasso_coordinate_descent_step(i = i,
        data_matrix = data_matrix, weights = weights,
        l1_penalty = l1_penalty)
    }
    delta <- vector()
    for (i in 1:length(weights)) {
      delta[i] <- old_weights[i] - weights[i]
    }
    if (max(abs(delta)) < tolerance) {
      converged <- TRUE
    }
  }
```

```
  weights/norms
}
```

3.4 Реализация координатного спуска

Мы будем использовать полный набор данных KC house, состоящий из 21613 наблюдений, и разделим его на тренировочные и тестовые данные, состоящие из 19451 и 2162 наблюдений соответственно. Мы будем использовать две функции, а именно «sqft_living» и «bedrooms», и выполним нашу функцию lasso_cyclical_coordinate_descent со штрафом $l_1 = 0$, что аналогично регрессии LS.

```
features = c("sqft_living", "bedrooms")
target = "price"
data_matrix <- construct_features_matrix(features = features, outputs =
    target,
  data = house_train_data)
weights <- lasso_cyclical_coordinate_descent(data_matrix = data_matrix,
  initial_weights = c(0, 0, 0), l1_penalty = 0, tolerance = 1e-07)
weights
```

[1] 97050.0942 305.2103 -57429.9302

Сравним наши результаты с функциями glm и lm в R.

```
x = model.matrix(price ~ sqft_living + bedrooms, data = house_train_data)[,
  -1]
y = house_train_data$price
lasso_model = glmnet(x, y, alpha = 1, lambda = 0, thresh = 1e-07)
predict(lasso_model, type = "coefficients", s = 0, exact = T)[1:3]
```

[1] 97052.1826 305.1896 -57417.7511

```
lm(price ~ sqft_living + bedrooms, data = house_train_data)$coef
```

(Intercept)	sqft_living	bedrooms
97050.0942	305.2103	-57429.9302

Действительно, наш алгоритм лассо очень хорошо сравнивается и дает те же результаты!

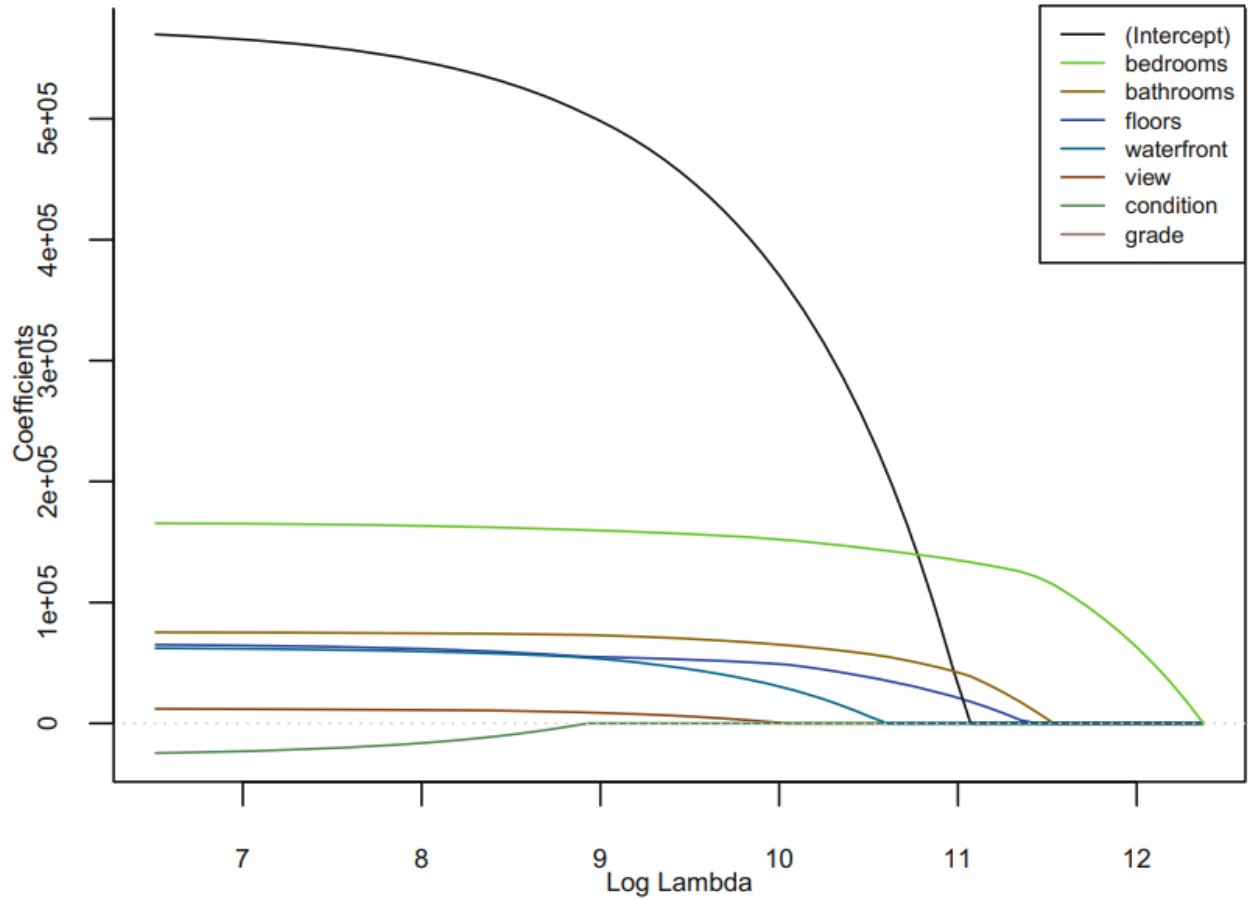

Рисунок 8 – Коэффициенты регрессии Лассо при изменении λ. Коэффициенты резко падают до 0 при $\lambda \to \infty$

В лассо, когда $\lambda = 0$, решение лассо является методом наименьших квадратов, а когда λ становится достаточно большим, лассо дает нулевую модель, в которой все оценки коэффициентов равны нулю. На рис. 8 мы построили 7 коэффициентов лассо при изменении λ от 0 до $1e + 10$. Можно заметить, что между двумя крайностями подгонки LS и нулевой модели, в зависимости от значения λ, лассо создает модели с любым количеством переменных, и большинство из них отклоняются по мере увеличения значения λ.

3.5 Компромисс отклонения отклонения в регрессии лассо

Большое значение λ приведет к высокому смещению (простая модель) и, следовательно, к низкой дисперсии($\hat{w} = 0$, для $\lambda = \infty$). Небольшое значение λ приведет к низкому смещению и, как следствие, большой дисперсии. Мы подробно обсудили регрессию; Теперь мы перейдем к следующей области

обучения с учителем - классификации.

ЗАКЛЮЧЕНИЕ

В ходе данной работы был успешно переведён отрывок из книги [1]. Для этого были использованы англо-русские словари [2], [3], [4], [5].

Мы познакомились с современными методами перевода технической литературы.

Была проведена работа с системой Google как в целях поиска информации, так и во время работы с переводом.

Было проведено знакомство с множеством различных англо-русских словарей.

Мы познакомились с методами реализации на языке R следующих понятий:

— Гребневая регрессия
— Градиентный спуск
— Регрессия лассо
— Координатный спуск

С помощью платформы Kaggle была проведена попытка проверить работоспособность кода, приведённого в книге в разделах 4.7 - 4.9. Kaggle выдал ошибку компиляции, а значит, код оказался нерабочим.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Ghatak, A. Machine Learning with R / A. Ghatak. - Singapore: Springer Nature, 2017. - 210 pp.

2 Борковский, А.Б. Англо-русский словарь по программированию и информатике / Борковский, А.Б., - СССР: Русский язык, 1987. - 336 стр.

3 Борковский, А.Б., Зайчик, Б.И. Словарь по программированию (английский, русский, немецкий, французский) / Зайчик, Б.И., - СССР: Русский язык, 1991. - 286 стр.

4 Александров, П.С., Ловатер, А.Дж. Англо-русский словарь математических терминов / Александров, П.С., Ловатер, А.Дж., - Россия: Мир, 2001. - 416 стр.

5 Мюллер, В.К.Самый полный англо-русский русско-английский словарь с современной транскрипцией: около 500 000 слов / Мюллер, В.К., - Россия: АСТ, 2016. - 800 стр.