Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Пермский государственный аграрно-технологический университет

имени академика Д.Н. Прянишникова»

Кафедра информационных технологий и программной инженерии

**ЛАБОРАТОРНАЯ РАБОТА № 1**

Линейная регрессия.

|  |  |
| --- | --- |
| Выполнил: | группа ПИНб-4  П.С. Плотников |
| Проверил: | доцент каф. ИТиПИ,  Т.А. Казаченко |

Пермь-2024 г.

# Градиентный спуск

**Формулировка задачи**

1) Запрограммируйте в среде RStudio метод градиентного спуска для заданной функции. Функция:

2) Графически отобразите линии уровня выбранной функции.

3) Выведите значение экстремума и число итераций, за который он достигнут.

4) Сравните полученное значение экстремума с экстремумом для данной функции в Excel.

**Программный код**

library("ggplot2")

# Определение функции

f <- function(x1, x2) {

85 \* x1^2 +168 \* x1 \* x2 + 85 \* x2^2 + 29 \* x1 - 51 \* x2 + 83

}

# Создание сетки значений x1 и x2

x1 <- seq(-50, 10, length.out = 100)

x2 <- seq(-10, 50, length.out = 100)

grid <- expand.grid(x1 = x1, x2 = x2)

# Вычислние значений z

grid$z <- with(grid, f(x1, x2))

# Построение графика линий уровня

ggplot(grid, aes(x = x1, y = x2, z = z)) +

geom\_contour(aes(color = after\_stat(level)), bins = 20) +

scale\_color\_viridis\_c() +

labs(title = "Линии уровня функции",

x = expression(x[1]),

y = expression(x[2]),

color = "Уровень") +

theme\_minimal()

# Градиент

gradient <- function(x1, x2) {

# Производная по x1

df\_x1 <- 170 \* x1 + 168 \* x2 + 29

# Производная по x2

df\_x2 <- 168 \* x1 + 170 \* x2 - 51

c(df\_x1, df\_x2)

}

# Поиск точки экстремума

gradient\_descent <- function(start, rate = 0.001, tol = 1e-6, max\_steps = 6000) {

# Начальные значения

x1 <- start[1]

x2 <- start[2]

iter <- 0

repeat {

# Счетчик проделанных шагов

iter <- iter + 1

# Получение градиента

grad <- gradient(x1, x2)

# Получение следующих точек

x1\_new <- x1 - rate \* grad[1]

x2\_new <- x2 - rate \* grad[2]

# Проверка условия сходимости

if (sqrt((x1\_new - x1)^2 + (x2\_new - x2)^2) < tol || iter >= max\_steps) {

break

}

# Присваивание новых значений

x1 <- x1\_new

x2 <- x2\_new

}

# Список с точкой экстремума, его значением и кол-вом шагов

list(

minimum\_x1 = round(x1, 2),

minimum\_x2 = round(x2, 2),

extremum = round(f(x1, x2), 2),

steps = iter

)

}

# Стартовая точка

start\_point <- c(0, 0)

result <- gradient\_descent(start\_point)

result

**Линии уровня**

В ходе выполнения написанного кода, были получены линии уровня, показанные на рисунке 1.

Описание: Изображение выглядит как текст, линия, диаграмма, рукописный текст

Автоматически созданное описание

Рисунок 1 – Линии уровня функции

**Вычисленный экстремум, количество итераций**

Для заданной функции был получен экстремум в точке , значение функции в этой точке: . Количество шагов: 5468. Большое количество шагов обусловлено тем, что функция имеет «овражный» вид и градиентный метод на ней работает плохо. Вывод результатов в консоли показан на рисунке 2.

Описание: Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 2 – Точка экстремума, значение экстремума и количество итераций

**Сравнение с Excel**

На рисунке 3 показано значение экстремума, полученное в Excel. Для получения значений была использована функция «Поиск решений».

Описание: Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 3 – Результат поиска экстремума в Excel

Сравнивая результаты работы программы с результатами в Excel, можно сделать вывод, что написанный код программы работает корректно.

# Линейная регрессия

**Ответы на вопросы**

1. Ответьте на вопросы: **Таблица 1**

|  |  |  |
| --- | --- | --- |
| **№** | **Вопрос** | **Ответ** |
|  | Как называется модель, изучаемая в лабораторной работе 1? | Линейная регрессионная модель |
|  | К какой категории методов ML она относится? | Обучение с учителем |
|  | Что может прогнозировать алгоритм, обученный данной моделью? | Вещественное значение, получаемое на основе выборки *действительное число* |
|  | В файле исходных данных ***alligators.txt***: |  |
| a) | Что является признаком, что ответом? | Признак: Length; Ответ: Weight |
| b) | Сколько ответов? | 24 / *число ответов равно числу объектов* |
|  | Запишите функцию ***Q(w0, w1, X)*** ошибки алгоритма для данных в файле ***alligators.txt***. |  |
|  | Запишите градиент этой функции, т.е. ***∇ Q(w0, w1, X).*** | . |
|  | Выпишите **в развернутом виде** от руки итерационную формулу для метода градиентного спуска **в парной регрессии.** | Изображение выглядит как текст, рукописный текст, линия, Шрифт  Автоматически созданное описание |

**Формулировка задачи об аллигаторах**

**Цель:** Для данных alligators.txt постройте регрессионную модель, отражающую зависимость веса аллигатора от его длины:

1) Загрузите данные из файла alligators.txt.

2) Сравните свое решение задачи с решением с посредством стандартного пакета для парной регрессии методом градиентного спуска.

3) Выведите диаграмму рассеяния в RStudio. В этих же осях выведите результат обученной модели.

**Программный код**

# Градиентный спуск для квадратичной ошибки

gradient\_descent\_mse <- function(x, y, rate = 0.0001, tol = 1e-6, max\_steps = 1000000) {

# Начальные параметры

# Смещение

w0 <- 0

# Наклон

w1 <- 0

# Количество наблюдений

n <- length(y)

iter <- 0

repeat {

# Счетчик итераций

iter <- iter + 1

# Вычисление предсказанных значений

y\_pred <- w0 + w1 \* x

# Вычисление градиентов

grad\_w0 <- -2 / n \* sum(y - y\_pred)

grad\_w1 <- -2 / n \* sum((y - y\_pred) \* x)

# Обновление параметров

w0\_new <- w0 - rate \* grad\_w0

w1\_new <- w1 - rate \* grad\_w1

# Проверка условия сходимости

if (sqrt((w0\_new - w0)^2 + (w1\_new - w1)^2) < tol || iter >= max\_steps) {

break

}

# Присвоение новых значений

w0 <- w0\_new

w1 <- w1\_new

}

# Результаты

list(

intercept = w0,

slope = w1,

steps = iter

)

}

# Данные

alligators <- read.table("./alligators.txt", header = TRUE)

x <- alligators$Length

y <- alligators$Weight

# Решение методом наименьших квадратов с помощью lm

model <- lm(Weight ~ Length, data = alligators)

lm\_intercept <- coef(model)[1]

lm\_slope <- coef(model)[2]

# Решение методом градиентного спуска

gd\_result <- gradient\_descent\_mse(x, y)

# Построение графика

plot(x, y, pch = 1, col = "black",

xlab = "Длина", ylab = "Вес",

main = "Зависимость веса аллигатора от его длины")

# Добавление линии регрессии из модели lm

abline(lm\_intercept, lm\_slope, col = "sienna2", lwd = 2, lty = 1)

# Добавление линии регрессии из градиентного спуска

abline(gd\_result$intercept, gd\_result$slope, col = "green", lwd = 2, lty = 2)

# Легенда

legend("topleft", legend = c("lm", "Градиентный спуск"),

col = c("sienna2", "green"), lty = c(1, 2), lwd = 2)

# Функция для вычисления среднеквадротичной ошибки

compute\_mse <- function(x, y, intercept, slope) {

y\_pred <- intercept + slope \* x

mean((y - y\_pred)^2)

}

# Вычисление среднеквадротичной ошибки для обеих моделей

lm\_mse <- compute\_mse(x, y, lm\_intercept, lm\_slope)

gd\_mse <- compute\_mse(x, y, gd\_result$intercept, gd\_result$slope)

# Вывод результатов

cat("Результаты линейной регрессии:\n")

cat("Метод наименьших квадратов (lm):\n")

cat(sprintf(" Смещение (intercept): %.4f\n", lm\_intercept))

cat(sprintf(" Наклон (slope): %.4f\n", lm\_slope))

cat(sprintf(" Среднеквадратичная ошибка (MSE): %.4f\n", lm\_mse))

cat("\nМетод градиентного спуска:\n")

cat(sprintf(" Смещение (intercept): %.4f\n", gd\_result$intercept))

cat(sprintf(" Наклон (slope): %.4f\n", gd\_result$slope))

cat(sprintf(" Среднеквадратичная ошибка (MSE): %.4f\n", gd\_mse))

cat(sprintf(" Количество шагов до сходимости: %d\n", gd\_result$steps))

**Результаты вычислений**

На рисунке 1 представлен вывод результатов в консоли.

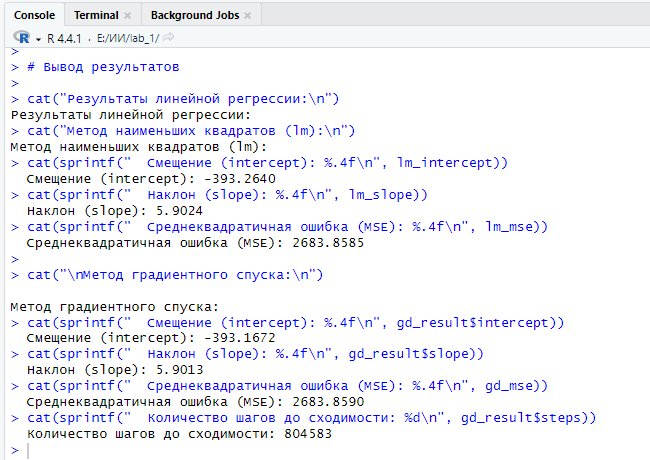


Рисунок 1 – Результаты обучения модели

На рисунке 2 представлен график полученной зависимости.

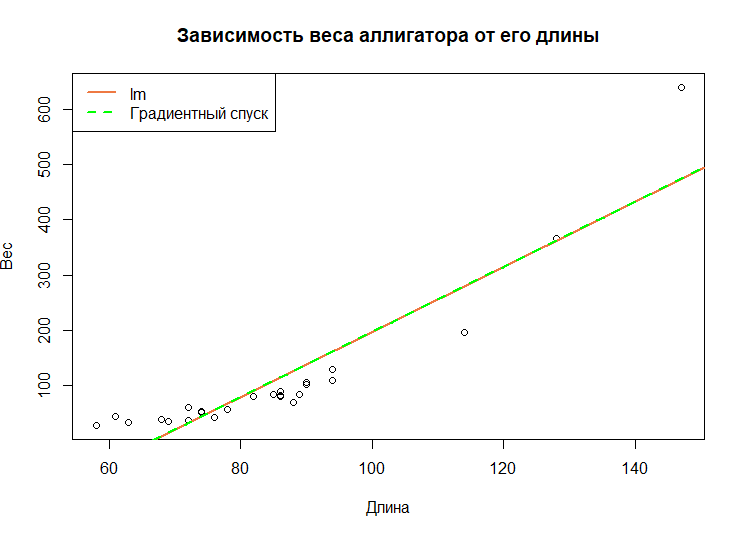


Рисунок 2 – Диаграмма рассеяния и графики модели парной регрессии

**Вывод**

По результатам обучения модели были получены данные, отраженные в таблице 1.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** |  |  | **Среднеквадратичная ошибка** |
| **Библиотека lm** | -393.2640 | 5.9024 | 2683.8585 |
| **Градиентный спуск** | -393.1672 | 5.9013 | 2683.8590 |

Согласно полученным данным, можно сделать вывод, что написанный ранее метод градиентного спуска дает результаты максимально схожие с встроенной библиотекой lm и имеет небольшую погрешность, которую можно компенсировать при помощи более тонкой настройке параметров градиентного спуска.

Однако стоит обратить внимание на то, что метод градиентного спуска для данной задачи выполняет много циклов, что говорит либо о некачественно настроенных параметрах модели, либо о нежелательном использовании данного метода при таком обучении модели.

**Формулировка задачи о возрасте исследуемых отложений**

**Цель**: Для данных cygage.txt, построив регрессию, выражающую зависимость возраста исследуемых отложений от глубины залегания, используя веса наблюдений:

1) Загрузите данные из файла cygage.txt.

2) Сравните свое решение задачи с решением с посредством стандартного пакета для парной регрессии методом градиентного спуска.

3) Выведите диаграмму рассеяния в RStudio. В этих же осях выведите результат обученной модели.

**Программный код**

# Загрузка данных

cygage <- read.table("./cygage.txt", header = TRUE)

# Извлечение переменных

x <- cygage$Depth

y <- cygage$calAge

w <- cygage$Weight

# Взвешенная линейная регрессия с использованием lm

lm\_model <- lm(y ~ x, weights = w)

lm\_intercept <- coef(lm\_model)[1]

lm\_slope <- coef(lm\_model)[2]

# Вычисление MSE для lm

compute\_mse <- function(x, y, w, intercept, slope) {

y\_pred <- intercept + slope \* x

mean(w \* (y - y\_pred)^2)

}

lm\_mse <- compute\_mse(x, y, w, lm\_intercept, lm\_slope)

# Градиентный спуск для взвешенной регрессии

weighted\_gradient\_descent <- function(x, y, w, start, rate = 0.000001, tol = 1e-4, max\_steps = 10000000) {

intercept <- start[1]

slope <- start[2]

iter <- 0

repeat {

iter <- iter + 1

# Градиенты по смещению и наклону

y\_pred <- intercept + slope \* x

grad\_intercept <- -2 \* sum(w \* (y - y\_pred))

grad\_slope <- -2 \* sum(w \* (y - y\_pred) \* x)

# Обновление параметров

intercept\_new <- intercept - rate \* grad\_intercept

slope\_new <- slope - rate \* grad\_slope

# Проверка условия сходимости

if (sqrt((intercept\_new - intercept)^2 + (slope\_new - slope)^2) < tol || iter >= max\_steps) {

break

}

intercept <- intercept\_new

slope <- slope\_new

}

# Возврат результата

list(

intercept = intercept,

slope = slope,

steps = iter

)

}

# Начальная точка и выполнение градиентного спуска

start\_point <- c(0, 0)

gd\_result <- weighted\_gradient\_descent(x, y, w, start\_point)

# Вычисление MSE для градиентного спуска

gd\_mse <- compute\_mse(x, y, w, gd\_result$intercept, gd\_result$slope)

# Построение графика

plot(x, y, pch = 1, col = "black", main = "Регрессия возраста отложений", xlab = "Глубина", ylab = "Возраст")

abline(lm\_intercept, lm\_slope, col = "blue", lwd = 2, lty = 2)

abline(gd\_result$intercept, gd\_result$slope, col = "red", lwd = 2, lty = 3)

legend("topleft", legend = c("Метод lm", "Градиентный спуск"), col = c("blue", "red"), lty = c(1, 2), lwd = 2)

# Вывод результатов

cat("Результаты взвешенной регрессии:\n")

cat("Метод наименьших квадратов (lm):\n")

cat(sprintf(" Смещение (intercept): %.4f\n", lm\_intercept))

cat(sprintf(" Наклон (slope): %.4f\n", lm\_slope))

cat(sprintf(" Среднеквадратичная ошибка (MSE): %.4f\n", lm\_mse))

cat("\nМетод градиентного спуска:\n")

cat(sprintf(" Смещение (intercept): %.4f\n", gd\_result$intercept))

cat(sprintf(" Наклон (slope): %.4f\n", gd\_result$slope))

cat(sprintf(" Среднеквадратичная ошибка (MSE): %.4f\n", gd\_mse))

cat(sprintf(" Количество шагов до сходимости: %d\n", gd\_result$steps))

**Результаты вычислений**

Для cygage.txt были получены результаты, показанные на рисунке 3.

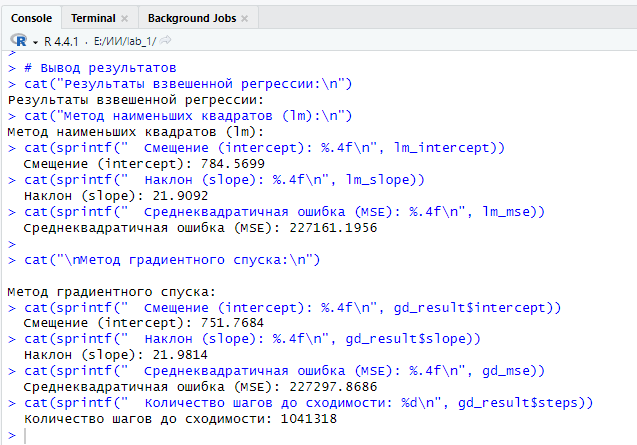


Рисунок 3 – Полученные результаты для cygage.txt

На рисунке 4 показан результат обучения модели.

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 4 – Диаграмма рассеяния и графики модели парной регрессии для cygage.txt

**Вывод**

В ходе обучения модели были получены данные, показанные в таблице 2.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** |  |  | **Среднеквадратичная ошибка** |
| **Библиотека lm** | 784.5699 | 21.9092 | 227161.1956 |
| **Градиентный спуск** | 751.7684 | 21.9814 | 227297.8686 |

Как и в первой задаче, метод градиентного спуска достаточно хорошо находит экстремум, однако при использовании весов его точность заметно снижается. Более того, количество итераций, требуемое для получения ответа, растет вплоть до нескольких порядков.