

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И АНАЛИЗА ДАННЫХ

Отчет
о прохождении преддипломной практики

студента 4 курса
Румянцева Андрея Кирилловича,
специальность "прикладная математика"

Руководитель практики:
зав. кафедрой ММАД,
канд. физ.-мат. наук, доцент
Бодягин Игорь Александрович

Минск, 2019

Содержание

ВВЕДЕНИЕ	2
1 Статистическое оценивание параметров линейной регрессии с выбросами при наличии группирования наблюдений	3
1.1 Альтернативные оценки параметров модели	7
1.2 Полиномиальная регрессия	8
2 Компьютерные эксперименты	10
2.1 Сравнительный анализ построенной оценки с альтернативной .	10
2.1.1 Эксперимент с изменением объема выборки	10
2.1.2 Эксперимент с полиномиальной регрессией	11
2.2 Эксперименты с изменением уровня переклассификации выборки	12
2.3 Сравнение вариаций с оценками без переклассификации	13
Заключение	15
Список Литературы	16
Приложение	17

ВВЕДЕНИЕ

В математической статистике широко используется регрессионная модель. Существует несколько подходов для оценки параметров регрессии, но далеко не все устойчивы к возникновению аномальных наблюдений, то есть таких наблюдений, которые не подчиняются общей модели. В реальной жизни аномальные наблюдения возникают постоянно. Такие наблюдения могут возникать по разным причинам: из-за ошибки измерения, из-за необычной природы входных данных. По этой причине большинство методов просто неприменимо. В прошлом веке в работах Хьюбера была заложена теория робастного оценивания.

Такие случаи, когда зависимые переменные наблюдаются с выбросами или с пропусками, хорошо исследованы [3]. Более сложный случай, когда вместо содержащих выбросы значений зависимой переменной наблюдаются номера классов(интервалов), в которые попадают эти наблюдения [11]. Темой курсового проекта было *"Статистическое оценивание параметров линейной регрессии с выбросами при наличии группирования наблюдений"*.

Целью преддипломной практики было продолжение исследования и улучшение оценок, построенных в курсовом проекте.

1 Статистическое оценивание параметров линейной регрессии с выбросами при наличии группирования наблюдений

В ходе курсового проекта рассматривается модель линейной регрессии:

$$y_i = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_n \end{pmatrix} \times \begin{pmatrix} 1 \\ x_{i1} \\ \dots \\ x_{in} \end{pmatrix}^T + \varepsilon_i, \quad (1)$$

$$y_i = f(x_i, \beta) + \varepsilon_i, \quad (2)$$

$$f(x_i, \beta) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_n x_{in}, \quad (3)$$

Здесь y_i – зависимая переменная, $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ – вектор регрессоров, $\{\beta_k, k = \overline{0, n}\}$ – коэффициенты линейной регрессии, а ε_i – случайная ошибка i -го эксперимента, распределение которой подчиняется нормальному закону с нулевым математическим ожиданием и дисперсией σ^2 , N -объем выборки. Каждый y_i принадлежит нормальному распределению:

$$y_i = f(x_i, \beta) + \varepsilon_i \sim \mathcal{N}(f(x_i, \beta), \sigma^2). \quad (4)$$

Предполагается, что выборка содержит выбросы, описываемые следующими соотношениями.

$$y_i^{\tilde{\varepsilon}} = (\xi_i) y_i + (1 - \xi_i) \eta_i, \quad (5)$$

где ξ_i принимает значение, равное 1, с вероятностью $1 - \tilde{\varepsilon}$ и значение, равное 0, с вероятностью $\tilde{\varepsilon}$:

$$\begin{cases} P\{\xi_i = 0\} = \tilde{\varepsilon}, \\ P\{\xi_i = 1\} = 1 - \tilde{\varepsilon}, \end{cases}, \quad (6)$$

η_i -случайная величина из некоторого вообще говоря неизвестного распределения.

Параметр ξ_i имеет следующий содержательный смысл: если $\xi_i = 0$, то вместо истинного значения мы наблюдаем выброс, если $\xi_i = 1$, то наблюдается истинное значение. Переменную $\tilde{\varepsilon}$ будем называть долей аномальных наблюдений. Величины ξ_i, x_i и η_i являются независимыми.

Пусть множество значений функции регрессии, т.е. множество \mathbb{R} , разбито на k непересекающихся полуинтервалов:

$$\mathbb{R} = (-\infty, a_1] \cup (a_1, a_2] \cup \dots \cup (a_{k-1}, +\infty). \quad (7)$$

Полученные полуинтервалы будем обозначать: ν_0, \dots, ν_{k-1} .

Предполагается, что каждый раз вместо истинного значения зависимой переменной y_i наблюдается только номер интервала, в который это наблюдение попало. Тогда для каждого y_i будем наблюдать лишь номер полуинтервала μ_i , в который он попал.

$$\mu_i = j, \text{ если } y_i \in \nu_j. \quad (8)$$

В таком случае принято говорить, что имеет место группирование наблюдений, а сами наблюдения называются группированными [3].

В курсовом проекте решается задача статистического оценивания параметров модели $\{\beta_k, k = \overline{0, n}\}$ по известным группированным наблюдениям с аномалиями.

Для этого была построим функцию правдоподобия:

$$l(\beta, \sigma^2, \mu_1, \dots, \mu_N) = \sum_{i=1}^N \ln(P(y_i \in \nu_{\mu_i})) = \quad (9)$$

$$= \sum_{i=1}^N \ln \begin{cases} \frac{1}{2}(\operatorname{erf}(\frac{a_{j+1}-f(x_i, \beta)}{\sqrt{2}\sigma}) - \operatorname{erf}(\frac{a_j-f(x_i, \beta)}{\sqrt{2}\sigma})), & i = \overline{1, k-2} \\ \frac{1}{2}(1 + \operatorname{erf}(\frac{a_1-f(x_i, \beta)}{\sqrt{2}\sigma})), & i = 0 \\ \frac{1}{2}(1 + \operatorname{erf}(\frac{a_{k-1}-f(x_i, \beta)}{\sqrt{2}\sigma})), & i = k-1 \end{cases}, \quad (10)$$

где:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \quad (11)$$

Для максимизирования функции правдоподобия решим систему уравнений:

$$\frac{\delta l}{\delta \beta} = 0, \quad (12)$$

где:

$$\begin{aligned}
\frac{\delta l}{\delta \beta} &= \frac{\delta \sum_{i=1}^N \ln P(y_i \in \nu_{\mu_i})}{\delta \beta} = \\
&= \frac{\delta \sum_{i=1}^N \ln \left(\frac{1}{2} \left(\operatorname{erf} \left(\frac{a_{\mu_i+1} - f(x_i, \beta)}{\sqrt{2}\sigma} \right) - \operatorname{erf} \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right) \right) \right)}{\delta \beta} = \\
&= \sum_{i=1}^N \left((1 - (\delta_{\mu_i 0} + \delta_{\mu_i k-1})) \frac{(\operatorname{erf}' \left(\frac{a_{\mu_i+1} - f(x_i, \beta)}{\sqrt{2}\sigma} \right) - \operatorname{erf}' \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right))}{(\operatorname{erf} \left(\frac{a_{\mu_i+1} - f(x_i, \beta)}{\sqrt{2}\sigma} \right) - \operatorname{erf} \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right))} + \right. \\
&\quad \left. + (\delta_{\mu_i 0} + \delta_{\mu_i k-1}) \frac{\operatorname{erf}' \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right)}{(1 + \operatorname{erf} \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right))} \right) (-1) \frac{\delta f(x_i, \beta)}{\delta \beta} = \quad (13) \\
&= - \sum_{i=1}^N \begin{pmatrix} 1 \\ x_{i1} \\ \dots \\ x_{in} \end{pmatrix} \times \left((1 - (\delta_{\mu_i 0} + \delta_{\mu_i k-1})) \frac{(\operatorname{erf}' \left(\frac{a_{\mu_i+1} - f(x_i, \beta)}{\sqrt{2}\sigma} \right) - \operatorname{erf}' \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right))}{(\operatorname{erf} \left(\frac{a_{\mu_i+1} - f(x_i, \beta)}{\sqrt{2}\sigma} \right) - \operatorname{erf} \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right))} + \right. \\
&\quad \left. + (\delta_{\mu_i 0} + \delta_{\mu_i k-1}) \frac{\operatorname{erf}' \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right)}{(1 + \operatorname{erf} \left(\frac{a_{\mu_i} - f(x_i, \beta)}{\sqrt{2}\sigma} \right))} \right).
\end{aligned}$$

δ_{ij} - символ Кронекера.

Уравнение (12) решается методом секущих.

Для уменьшения влияния выбросов будем использовать переклассификацию выборки методом K ближайших соседей. Идея заключается в том, что аномальные наблюдения с большей вероятностью попадают не в те интервалы, в которые попадают истинные наблюдения. При этом переклассификация может помочь отнести аномальные наблюдения к истинным классам и улучшить качество оценивания.

На первом этапе для каждого вектора x_i имели класс μ_i : т.е. пару (x_i, μ_i) . Далее выполним переклассификацию выборки. Для этого построим новую выборку такого же объема N . Пройдемся по каждому элементу (x_i, μ_i) выборки и для этого наблюдения построим новое:

$$(x_i, \check{\mu}_i), \quad (14)$$

где $\check{\mu}_i$ получен по методу K -соседей.

$$\check{\mu}_i = \arg \max_j \sum_{k \in V_i, k \neq i} \delta_{\mu_k j}, \quad (15)$$

где V_i множество индексов l первых K векторов x_l , отсортированных по возрастанию расстояния до вектора x_i .

После переклассификации выборки, применим к ней функцию правдоподобия из уравнений (9), только теперь с использованием новых классов $\check{\mu}_i$ вместо μ_i . Аналогично максимизируем ее и найдем новую оценку параметров $\hat{\beta}$.

1.1 Альтернативные оценки параметров модели

Рассмотрим альтернативный метод оценивания параметров модели регрессии, основанный на замене группированных наблюдений серединами соответствующих интервалов. Такой метод встречается в литературе, например в [9].

Метод заключается в следующем: пусть имеется μ_i - номер полуинтервала, в который попало очередное наблюдение y_i . Ему соответствует полуинтервал ν_{μ_i} (из (8)), т.е. полуинтервал:

$$y_i \in (a_{\nu_{\mu_i}}, a_{\nu_{\mu_i}+1}], i = \overline{1, N} \quad (16)$$

(считаем что $a_1 < y_i < a_{k-1}, i = \overline{1, N}$, т.е $1 \leq \mu_i \leq k - 2$).

Найдем центральную точку этого интервала, т.е. точку

$$\check{y}_i = \frac{a_{\nu_{\mu_i}} + a_{\nu_{\mu_i}+1}}{2}. \quad (17)$$

Построим для всех значений функции регрессии y_i значения \check{y}_i . Будем использовать в качестве значений функции регрессии полученные значения \check{y}_i , а в качестве регрессоров x_i и построим МНК оценки параметров β .

Теперь имеет три вида оценок: оценки максимального правдоподобия, оценки максимального правдоподобия с переклассификацией, МНК по серединам интервалов.

1.2 Полиномиальная регрессия

Введем теперь модель полиномиальной регрессии.

$$\begin{aligned}
 y_i &= \beta_0 + \beta_1 x_{i1}^1 + \beta_2 x_{i2}^2 + \cdots + \beta_n x_{in}^n + \varepsilon_i, i = \overline{1, N}, \\
 y_i &= \sum_{l=1}^n x_{il}^{l-1} + \varepsilon_i, i = \overline{1, N}, \\
 y_i &= f(x_i, \beta) + \varepsilon_i, \\
 f(x_i, \beta) &= \beta_0 + \beta_1 x_{i1}^1 + \beta_2 x_{i2}^2 + \cdots + \beta_n x_{in}^n
 \end{aligned} \tag{18}$$

В случае полиномиальной регрессии также справедливо:

$$y_i = f(x_i, \beta) + \varepsilon_i \sim \mathcal{N}(f(x_i, \beta), \sigma^2). \tag{19}$$

Поскольку оценки строились путём максимизирования функции:

$$\begin{aligned}
 l(\beta, \sigma^2, \nu_0, \dots, \nu_{k-1}) &= \sum_{i=1}^N \ln(P(y_i \in \nu_{\mu_i})) = \\
 &= \sum_{i=1}^N \ln \begin{cases} \frac{1}{2}(\operatorname{erf}(\frac{a_{j+1}-f(x_i, \beta)}{\sqrt{2}\sigma}) - \operatorname{erf}(\frac{a_j-f(x_i, \beta)}{\sqrt{2}\sigma})), & i = \overline{1, k-2} \\ \frac{1}{2}(1 + \operatorname{erf}(\frac{a_1-f(x_i, \beta)}{\sqrt{2}\sigma})), & i = 0 \\ \frac{1}{2}(1 + \operatorname{erf}(\frac{a_{k-1}-f(x_i, \beta)}{\sqrt{2}\sigma})), & i = k-1 \end{cases}, \tag{20}
 \end{aligned} \tag{21}$$

а функция правдоподобия максимизировалась путём решения системы уравнений:

$$\frac{\delta l}{\delta \beta} = 0, \tag{22}$$

которая примет вид:

$$\begin{aligned}
 \frac{\delta l}{\delta \beta} &= \frac{\delta \sum_{i=1}^N \ln P(y_i \in \nu_{\mu_i})}{\delta \beta} = \\
 &= \frac{\delta \sum_{i=1}^N \ln(\frac{1}{2}(\operatorname{erf}(\frac{a_{\mu_i+1}-f(x_i, \beta)}{\sqrt{2}\sigma}) - \operatorname{erf}(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma})))}{\delta \beta} = \\
 &= \sum_{i=1}^N \left((1 - (\delta_{\mu_i 0} + \delta_{\mu_i k-1})) \frac{(\operatorname{erf}'(\frac{a_{\mu_i+1}-f(x_i, \beta)}{\sqrt{2}\sigma}) - \operatorname{erf}'(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma})))}{(\operatorname{erf}(\frac{a_{\mu_i+1}-f(x_i, \beta)}{\sqrt{2}\sigma}) - \operatorname{erf}(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma}))} + \right. \\
 &\quad \left. + (\delta_{\mu_i 0} + \delta_{\mu_i k-1}) \frac{\operatorname{erf}'(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma})}{(1 + \operatorname{erf}(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma}))} \right) (-1) \frac{\delta f(x_i, \beta)}{\delta \beta} = \tag{23}
 \end{aligned}$$

$$\begin{aligned}
= - \sum_{i=1}^N \begin{pmatrix} 1 \\ x_{i1}^1 \\ \dots \\ x_{in}^n \end{pmatrix} \times \left((1 - (\delta_{\mu_i 0} + \delta_{\mu_i k-1})) \frac{(\operatorname{erf}'(\frac{a_{\mu_i+1}-f(x_i, \beta)}{\sqrt{2}\sigma}) - \operatorname{erf}'(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma}))}{(\operatorname{erf}(\frac{a_{\mu_i+1}-f(x_i, \beta)}{\sqrt{2}\sigma}) - \operatorname{erf}(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma}))} + \right. \\
\left. + (\delta_{\mu_i 0} + \delta_{\mu_i k-1}) \frac{\operatorname{erf}'(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma})}{(1 + \operatorname{erf}(\frac{a_{\mu_i}-f(x_i, \beta)}{\sqrt{2}\sigma}))} \right),
\end{aligned}$$

то построенные оценки также применимы и для полиномиальной регрессии.

Также как и в случае линейной регрессии считаем, что выборка содержит выбросы, т.е., аналогично:

$$y_i^{\tilde{\varepsilon}} = (\xi_i) y_i + (1 - \xi_i) \eta_i, \quad (24)$$

здесь y_i задаются формулой (18).

2 Компьютерные эксперименты

2.1 Сравнительный анализ построенной оценки с альтернативной

2.1.1 Эксперимент с изменением объема выборки

В следующем эксперименте был произведен сравнительный анализ вариаций ОМП-оценок с МНК оценками в зависимости от объема выборки.

Объем выборки N изменялся от $N_1 = 100$ до $N_2 = 500$, при этом выборка дополнялась, а не генерировалась новая. Использовалась модель линейной регрессии. Доля выбросов была постоянна и равнялась $\tilde{\varepsilon} = 0.08$. Параметры регрессии были постоянными и равнялись $\beta = (90, 4)^T$. Регрессоры x_i были из равномерного распределения $U(-5, 5)$, ошибки экспериментов $\varepsilon_i \sim \mathcal{N}(0, 16)$.

Таблица 1: Параметры модели и оценок

Параметры программы	
Переменная	значение
Размер выборки N	от 100 до 500
Доля выбросов $\tilde{\varepsilon}$	0.08
Параметры регрессии β	$(90, 4)$
Регрессоры x_i	$\sim U(-5, 5)$
ε_i	$\sim \mathcal{N}(0, 16)$
η_i	$\sim \mathcal{N}(100, 100)$
Величина K	10

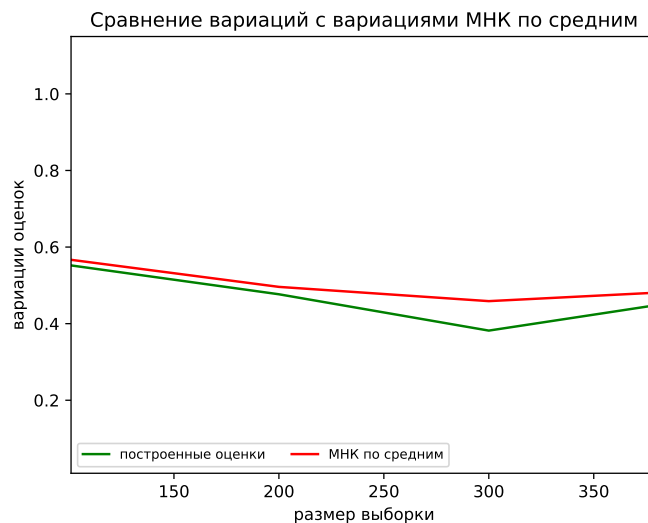


Рис. 1: Сравнение вариаций оценок

При сравнении графиков вариаций (рис.1) можно сделать вывод, что ОМП дают лучший результат,

2.1.2 Эксперимент с полиномиальной регрессией

Был проведен эксперимент с полиномиальной регрессией. Использовались те же параметры модели (таблица 3), объем выборки N изменялся от 100 до 1000:

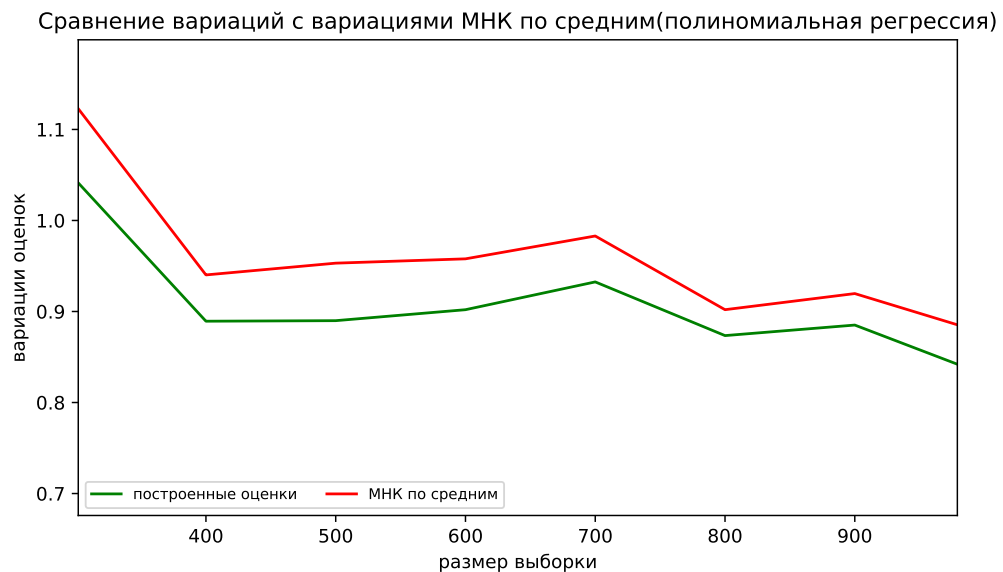


Рис. 2: Вариации оценок в случае полиномиальной регрессии

Оба метода имели схожее поведение при изменении объема выборки, но построенные оценки максимального правдоподобия стабильно показывали лучший результат.

2.2 Эксперименты с изменением уровня переклассификации выборки

В ходе преддипломной практики были построены эксперименты с изменением величины K для метода K -ближайших соседей, используемого в переклассификации.

Объем выборки N был постоянным: $N = 500$. Использовалась модель линейной регрессии. Доля выбросов была постоянна и равнялась $\tilde{\varepsilon} = 0.08$. Параметры регрессии были постоянными и равнялись $\beta = (90, 4)^T$. Регрессоры x_i были из равномерного распределения $U(-5, 5)$, ошибки экспериментов $\varepsilon_i \sim \mathcal{N}(0, 16)$. Величина K менялась от 10 до 40.

Таблица 2: Параметры модели и оценок экспериментов с переклассификацией выборки

Параметры программы	
Переменная	значение
Размер выборки N	500
Доля выбросов $\tilde{\varepsilon}$	0.08
Параметры регрессии β	$(90, 4)$
Регрессоры x_i	$\sim U(-5, 5)$
ε_i	$\sim \mathcal{N}(0, 16)$
η_i	$\sim \mathcal{N}(100, 100)$
Величина K	от 10 до 40

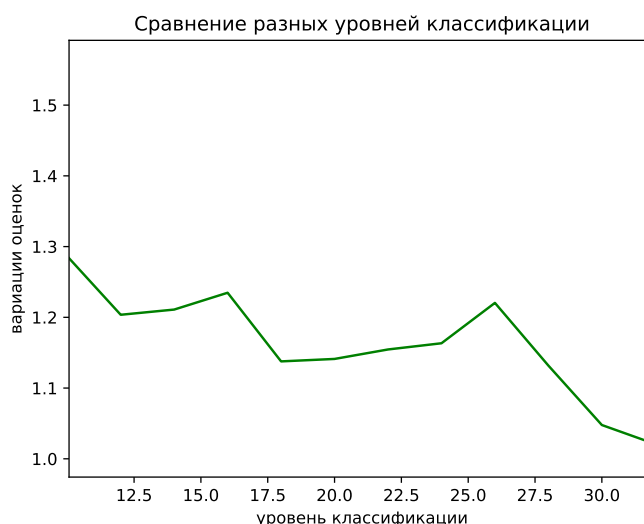


Рис. 3: Зависимость вариаций от K — числа соседей, используемого в переклассификации выборки

В результате получилось, что при увеличении константы K точность оценки параметров растёт.

2.3 Сравнение вариаций с оценками без переклассификации

Были проведены эксперименты для сравнения эмпирической вариации оценок максимального правдоподобия, когда использовалась вышеописанная переклассификация и когда не использовалась. При этом на каждой итерации выборка увеличивалась.

Объем выборки N изменялся от $N_1 = 100$ до $N_2 = 400$, при этом выборка дополнялась, а не генерировалась новая. Использовалась модель линейной регрессии. Доля выбросов была постоянна и равнялась $\tilde{\varepsilon} = 0.08$. Параметры регрессии были постоянными и равнялись $\beta = (90, 4)^T$. Регрессоры x_i были из равномерного распределения $U(-5, 5)$, ошибки экспериментов $\varepsilon_i \sim \mathcal{N}(0, 16)$. В методе, где использовалась переклассификация, величина K выбиралась: $K = 10$.

Таблица 3: Параметры модели и оценок экспериментов

Параметры программы	
Переменная	значение
Размер выборки N	от 100 до 400
Доля выбросов $\tilde{\varepsilon}$	0.08
Параметры регрессии β	$(90, 4)$
Регрессоры x_i	$\sim U(-5, 5)$
ε_i	$\sim \mathcal{N}(0, 16)$
η_i	$\sim \mathcal{N}(100, 100)$
В методе, с переклассификацией величина K	10

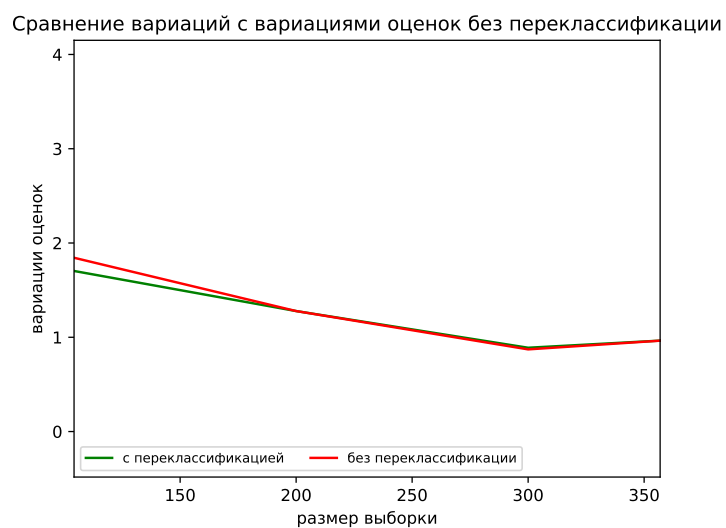


Рис. 4: Сравнение вариаций оценок когда используется и не используется переклассификация

Заключение

В ходе преддипломной практики был проведен аналитический обзор литературы методов статистического анализа данных при наличии классифицированных наблюдений с искажениями. В результате был реализован альтернативный метод - *метод наименьших квадратов по центрам интервалов*.

Был проведен сравнительный анализ альтернативного метода с оценками максимального правдоподобия. Оценки максимального правдоподобия с переклассификацией выборки показали наилучшие результаты.

Над оценками максимального правдоподобия с переклассификацией выборки были осуществлены эксперименты, в которых изменялась константа K для метода K —соседей (см. п. 2.2). Выяснилось, что увеличение константы K повышает точность аппроксимации.

Реализованные методы максимального правдоподобия с переклассификацией и МНК по серединам интервалов были обобщены на случай полиномиальной регрессии.

По проведенным экспериментам видно, что ОМП с переклассификацией показывают не хуже результаты, чем альтернативные оценки. Можно добиться более точных результатов аппроксимации, если хорошо подобрать параметры оценок.

Список литературы

- [1] Хьюбер Дж П. *Робастность в статистике: пер. с англ.* – М.: Мир, 1984.- 304 с.
- [2] Харин Ю.С., Зуев Н.М., Жук Е.Е. *Теория вероятностей, математическая и прикладная статистика: учебник* – Минск: БГУ, 2011.-463 с.
- [3] Е. С Агеева, чл.-корр. НАН Беларуси Ю.С. Харин *Состоятельность оценки максимального правдоподобия параметров множественной регрессии по классифицированным наблюдениям*
- [4] John Fox, Sanford Weisberg *Robust Regression* – October 8, 2013
- [5] А.В. Омельченко *Робастное оценивание параметров полиномиальной регрессии второго порядка* – Харьковский национальный университет радиоэлектроники, Украина, 2009
- [6] Özlem Gürünlü Alma *Comparison of Robust Regression Methods in Linear Regression* – Int. J. Contemp. Math. Sciences, Vol. 6, 2011, no. 9, 409 - 421 с.
- [7] Sergei Winitzki *A handy approximation for the error function and its inverse.*
- [8] Мандрик П.А., Репников В.И., Фалейчик Б.В., *Численные методы* [Электронный ресурс].
- [9] Paolo Giordani *Linear regression analysis for interval-valued data based on the Lasso technique* – Department of Statistical Sciences Sapienza University of Rome
- [10] Masahiro Inuiguchi, Tetsuzo Tanino, *interval linear regression methods based on minkowski difference a bridge between traditional and interval linear regression models.* – KYBERNETIKA, volume 42, 2006 , number 4, pages 423 - 440
- [11] Nelson, W., Hahn, G.J. *Technometrics.* volume 14, 1972, pages 247–269.

Приложение

Метод наименьших квадратов по центрам интервалов

```
class ApproximationGEMModelNaive(ApproximationGEMModelRedesigned):
    def fit(self):
        self.classify()

        def ex_generator(mu_data):
            for i in range(0, self.endogen.size):
                if mu_data[i] is None:
                    continue
                a_mu_i_plus_1 = mu_data[i] * Defines.INTERVAL_LENGTH
                a_mu_i = mu_data[i] * Defines.INTERVAL_LENGTH - Defines.INTERVAL_LENGTH
                yield (a_mu_i_plus_1 + a_mu_i) / 2

        naive_ex_data_positive = np.fromiter(ex_generator(self._np_freq_positive), float)
        naive_ex_data_negative = np.fromiter(ex_generator(self._np_freq_negative), float)

        naive_ex_data_full = np.append(naive_ex_data_positive, naive_ex_data_negative)

        z, resid, rank, sigma = np.linalg.lstsq(self.exogen, naive_ex_data_full, rcond=None)
        return z
```

Моделирование полиномиальной регрессии:

```
def modulate_polynomial_regression(regression_sample_quintity, regression_outlier_percentage):
    regression_parameters = ACCURATE_RESULT
    _x_points = np.zeros(shape=[regression_sample_quintity, len(regression_parameters)])
    _y_points = np.zeros(shape=regression_sample_quintity)

    def np_random_polynomial(size):
        _res = np.zeros(size)
        for i in range(0, size):
            _res[i] = random.uniform(-5, 5) ** (i + 1)

        return _res

    for i in range(0, regression_sample_quintity):
        _x_points[i] = np.append(np.ones(1), np_random_polynomial(len(ACCURATE_RESULT) - 1))
        if random.random() > regression_outlier_percentage / 100:
            _y_points[i] = (_x_points[i] * ACCURATE_RESULT) + np.random.normal(0, 4)
        else:
            _y_points[i] = np.random.normal(100.0, 15.0, size=1)

    return _x_points, _y_points
```

Моделирование линейной регрессии:

```
def modulateRegression(regression_sample_quintity, regression_outlier_percentage):
    regression_parameters = ACCURATE_RESULT
    _x_points = np.zeros(shape=[regression_sample_quintity, len(regression_parameters)])
    _y_points = np.zeros(shape=regression_sample_quintity)

    for i in range(0, regression_sample_quintity):
        if random.random() > regression_outlier_percentage / 100:
            _x_points[i] = np.append(np.ones(1), np.random.uniform(-5, 5, size=len(regression_parameters) - 1))
            _y_points[i] = (_x_points[i] * regression_parameters) + np.random.normal(0, 4)
        else:
            _x_points[i] = np.append(np.ones(1), np.random.uniform(-5, 5, size=len(regression_parameters) - 1))
            _y_points[i] = np.random.normal(100.0, 15.0, size=1)

    return _x_points, _y_points
```

Метод наименьших квадратов по центрам интервалов:

```
def fit_data_naive_classic():
    sample_sizes = []
    all_results_classic = []
    all_results_naive = []
    for sample_size in range(SAMPLE_SIZE_MIN, SAMPLE_SIZE_MAX+1, SAMPLE_SIZE_STEP):
```

```

successful_fit = False
while not successful_fit:
    x_points, y_points = modulateRegression(sample_size, OUTLIER_PERCENTAGE)
    approx_model = groupingEstimates.GEM(x_points, y_points)
    approx_model_naive = groupingEstimatesNaive.GEM_N(x_points, y_points)
    try:
        result = approx_model.fit()
        print("GEM {}".format(result))
        result_naive = approx_model_naive.fit()
        print("GEM_N {}".format(result_naive))

        successful_fit = True

        all_results_classic.append(result)
        all_results_naive.append(result_naive)
        sample_sizes.append(sample_size)
    except KeyboardInterrupt:
        print("stopping...")
        np.save(NP_DATA_PATH + "gem_res_classic", all_results_classic)
        np.save(NP_DATA_PATH + "gem_res_naive", all_results_naive)
        np.save(NP_DATA_PATH + "gem_sizes", sample_sizes)
        quit()
    except Exception as e:
        print(e)
np.save(NP_DATA_PATH + "gem_res_classic", all_results_classic)
np.save(NP_DATA_PATH + "gem_res_naive", all_results_naive)
np.save(NP_DATA_PATH + "gem_sizes", sample_sizes)

```

График с разным объемом выборки:

```

def plot_with_different_sample_size():
    sample_sizes = []
    all_results_with_classification = []
    all_results_without_classification = []

    x_points = None
    y_points = None

    for sample_size in range(SAMPLE_SIZE_MIN, SAMPLE_SIZE_MAX+1, SAMPLE_SIZE_STEP):
        successful_fit = False
        while not successful_fit:
            x_points_t, y_points_t = modulateRegression(sample_size, OUTLIER_PERCENTAGE)

            if x_points is None or y_points is None:
                x_points = x_points_t
                y_points = y_points_t
            else:
                x_points = np.append(x_points, x_points_t, axis=0)
                y_points = np.append(y_points, y_points_t, axis=0)

            approx_model = groupingEstimates.GEM(x_points, y_points)
            try:
                result = approx_model.fit()
                print("GEM {}".format(result))
                result_without = approx_model.fit_without_reclassification()
                print("GEM_without {}".format(result_without))

                successful_fit = True

                all_results_with_classification.append(result)
                all_results_without_classification.append(result_without)
                sample_sizes.append(sample_size)
            except KeyboardInterrupt:
                print("stopping...")
                np.save(NP_DATA_PATH + "gem_res_with", all_results_with_classification)
                np.save(NP_DATA_PATH + "gem_res_without", all_results_without_classification)
                np.save(NP_DATA_PATH + "gem_sizes_with_without", sample_sizes)
                quit()
            except Exception as e:
                print(e)
        np.save(NP_DATA_PATH + "gem_res_with", all_results_with_classification)
        np.save(NP_DATA_PATH + "gem_res_without", all_results_without_classification)
        np.save(NP_DATA_PATH + "gem_sizes_with_without", sample_sizes)

```

График с разным уровнем переклассификации:

```
def plot_with_different_reclassification_level():
    reclassification_levels = []
    all_results_with_classification = []
    recl_level_min = 10
    recl_level_max = 40

    x_points, y_points = modulateRegression(500, OUTLIER_PERCENTAGE)

    for recl_level in range(recl_level_min, recl_level_max + 1, 2):
        GroupingEstimatesDefines.RECLASSIFICATION_LEVEL = recl_level

        successful_fit = False
        while not successful_fit:
            approx_model = groupingEstimates.GEM(x_points, y_points)
            try:
                result = approx_model.fit()
                print("GEM {}".format(result))

                successful_fit = True

                all_results_with_classification.append(result)
                reclassification_levels.append(recl_level)
            except KeyboardInterrupt:
                print("stopping...")
                np.save(NP_DATA_PATH + "gem_with_dif_level_results", all_results_with_classification)
                np.save(NP_DATA_PATH + "gem_with_dif_level_levels", reclassification_levels)
                quit()
            except Exception as e:
                print(e)
        np.save(NP_DATA_PATH + "gem_with_dif_level_results", all_results_with_classification)
        np.save(NP_DATA_PATH + "gem_with_dif_level_levels", reclassification_levels)
```