МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И АНАЛИЗА ДАННЫХ

Отчет
о прохождении преддипломной практики

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

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

Минск, 2019

1 Задание на практику

- Провести аналитический обзор литературы методов статанализа данных при наличии классифицированных данных с искажениями.
- Реализовать альтернативные встречаемые в литературе методы статистического анализа данных при наличии классифицированных данных с искажениями.
- Провести сравнительный анализ реализованного в ходе курсового проекта метода с альтернативными.
- Обобщить все реализованные методы с линейной на полиномиальную регрессию.
- Подготовить отчет по преддипломной практике.

Содержание

ВВЕДЕНИЕ

Целью преддипломной практики было продолжение исследования и улучшение оценок, построенных в курсовом проекте. Темой курсового проекта было *"Статистическое оценивание параметров линейной регрессии с выбросами при наличии группирования наблюдений"*.

Оценки были построенны с помощью максимизирования функции правдоподобия:

$$l(\beta,\sigma^2,\nu_0,\dots,\nu_{k-1}) = \ln(\prod_{i=1}^{n} P(\mu_i=j)) = \tag{1}$$

$$= \sum_{i=1}^{n} \ln(P(\mu_i=j)), \tag{2}$$

где:

$$P(\mu_i=j)=P(y_i\in\nu_{\mu_i}), \tag{3}$$

y_i - значения функции регрессии, а $\nu_0,\dots,\nu_{k-1}$ - номера полуинтервалов, разбивающих множество значений функции регрессии:

$$(-\infty,a_1]\bigcup(a_1,a_2]\bigcup\cdots\bigcup(a_{k-1},+\infty)=\mathcal{R} \tag{4}$$

μ_i номер полуинтервала, в который он попал y_i.

$$\mu_i=j, \text{если } y_i \text{ отнесли к полуинтервалу } \nu_j. \tag{5}$$

Задача максимизирования решала с помощью решения нелинейной системы уравнений:

$$\frac{\delta l}{\delta\beta}=0, \tag{6}$$

где

$$\frac{\delta l}{\delta\beta}=\frac{\delta\sum_{i=1}^{n}\ln(P(\mu_i=j))}{\delta\beta}=\frac{\delta\sum_{i=1}^{n}\ln P(y_i\in\nu_{\mu_i})}{\delta\beta}=$$

$$=\frac{\delta\sum_{i=1}^{n}\ln(\frac{1}{2}(\mathrm{erf}(\frac{a_{\mu_i+1}-f(x_i,\beta)}{\sqrt{2}\sigma})-\mathrm{erf}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma})))}{\delta\beta}=$$

$$=\sum_{i=1}^{n}\Bigg((1-(\delta_{\mu_i0}+\delta_{\mu_ik-1}))\frac{(\mathrm{erf'}(\frac{a_{\mu_i+1}-f(x_i,\beta)}{\sqrt{2}\sigma})-\mathrm{erf'}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma}))}{(\mathrm{erf}(\frac{a_{\mu_i+1}-f(x_i,\beta)}{\sqrt{2}\sigma})-\mathrm{erf}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma}))}+$$

$$+(\delta_{\mu_i0}+\delta_{\mu_ik-1})\frac{\mathrm{erf'}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma})}{(1+\mathrm{erf}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma}))}\Bigg)(-1)\frac{\delta f(x_i,\beta)}{\delta\beta})= \tag{7}$$

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

$$+ (\delta_{\mu_i 0} + \delta_{\mu_i k-1}) \frac{\text{erf'}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma})}{(1+\text{erf}(\frac{a_{\mu_i}-f(x_i,\beta)}{\sqrt{2}\sigma}))} \Big).$$

2 Изучение материала

В ходе выполнения преддипломной практики были изучены следующие источники:

В источниках был встречен *метод наименьших квадратов по центрам интервалов.* Метод заключается в следующем: пусть имеется μ_i - номер полуинтервала, в который попало очередное наблюдение y_i. Ему соответствует полуинтервал ν_{μ_i} (см(4)), т.е. полуинтервал:

$$(a_{\nu_{\mu_i}}, a_{\nu_{\mu_i}+1}], \tag{8}$$

(считаем что $a_1 < y_i < a_{k-1}, i = \overline{1, n}$).

Найдем центральную точку этого интервала, т.е. точку

$$\check{y}_i = \frac{a_{\nu_{\mu_i}} + a_{\nu_{\mu_i}+1}}{2} \tag{9}$$

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

3 Реализация оценка

Описанные оценки были построены путем наследования от исходных оценок и переопределения соответствуюего метода fit().

```
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
```

4 Компьютерные эксперименты

4.1 Параметры модели и оценок

Параметры программы	
Переменная	значение
Размер выборки N	1000
Доля выбросов $\widetilde{\varepsilon}$	0.8
Параметры регрессии β	$(90, 4)$
Регрессоры x_i	$\sim U(-5, 5)$
ε_i	$\sim N(0, 16)$
η_i	$\sim N(100, 100)$
Величина K из пункта 2.3 курсового проекта	10

,

4.2 Сравнительный анализ построенной оценки с альтернативной

Если сравнить вариации оценок построенные на рис.2, можно увидеть, что оценки, построенные по методу, предложенному в курсовом проекте, показывают лучшие результаты

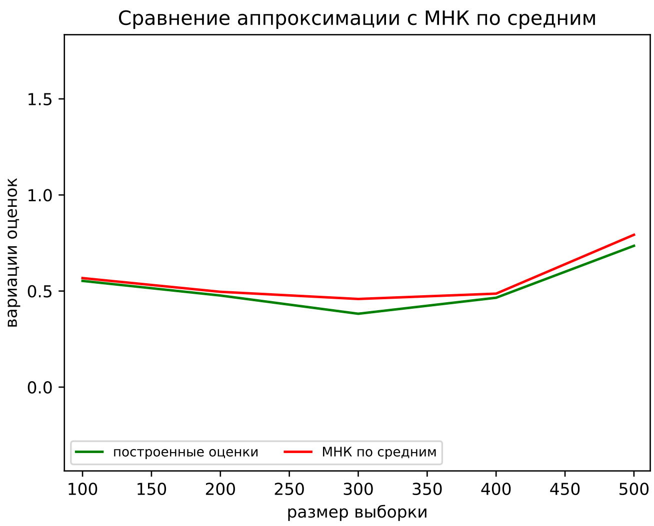

Рис. 1: Сравнение вариаций оценок

4.3 Дополнительные эксперименты

В ходе преддипломной практики были построены эксперименты с изменением величины K для метода k-близлежайших соседей, используемого в переклассификации. Параметры использовались такие же, как в ранее приведенной таблице.

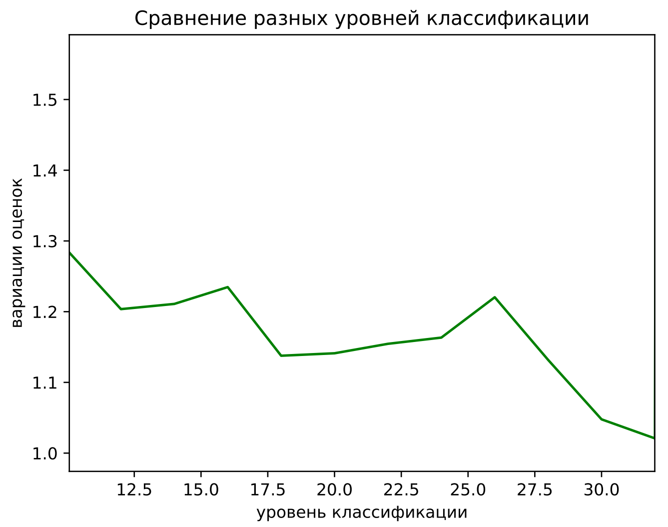

Рис. 2: Зависимость от K, упомянотого в пункте 2.3 курсового проекта

Были проведены эксперименты, где включалась и отключалась переклассификация и при этом на каждой итерации выборка увеличивалась.

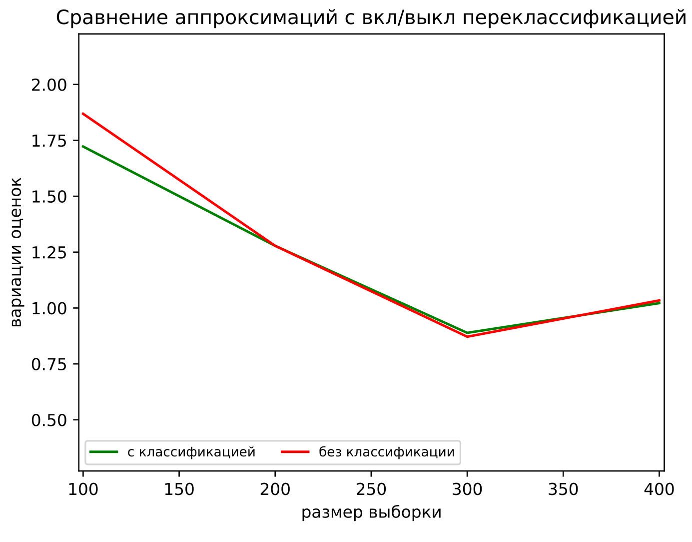

Рис. 3: Сравнение вариаций оценок когда используется и не используется переклассификация

4.4 Использование полиномиальной регрессии

Несложно заметить, что построенные в курсовом проекте оценки никак не зависят от регрессоров, они выступают лишь как параметры, поэтому можно моделировать полиномиальную регрессию и применить к ней описанный метод.

Были построены графики, схожие с рис.3. В итоге получились такие результаты:

Заключение

По проведенным экспериментам видно, что оценки показывают не хуже результаты, чем альтернативные оценки, поэтому их можно рассматривать к использованию. 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 правдопобия 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 c.

[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].

Приложение

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)
```