Поиск калибровочных границ методами машинного обучения

В. Г. Мосин

Аннотация

Предложен метод, позволяющий по данной дискретной случайной величине вычислять границы отклонений, превышение которых дает аномальное значение. Проведена серия компьютерных экспериментов, благодаря которым сформулированы и указаны границы применения метода. Рассмотрены примеры на синтетических данных.

Содержание

1 Введение **2**
- 1.1 Теоретическая часть . . . 2
- 1.2 Постановки задачи . . . 3
  - 1.2.1 Предмет исследования . . . 3
  - 1.2.2 Методика исследования . . . 3
  - 1.2.3 Цель исследования . . . 3
- 1.3 Технологии . . . 3
  - 1.3.1 Стандартные библиотеки . . . 3
  - 1.3.2 Авторские функции . . . 4

2 Генератор синтетических данных **4**
- 2.1 Константы и параметры генерации . . . 4
- 2.2 Случайный шум . . . 4
- 2.3 Аномальные отклонения и их позиции . . . 4
- 2.4 Внедрение аномалий . . . 5
- 2.5 Примеры . . . 6

3 Алгоритм **7**
- 3.1 Установка стартовых значений цикла и генерация данных 7
- 3.2 Вычисление отклонений . . . 7
- 3.3 Агрегация промежуточных результатов . . . 7
- 3.4 Сортировка отклонений . . . 8

3.5 Поиск локтевой точки 8
3.6 Процент обнаруженных аномалий 8
3.7 Внутренний цикл по процентам заданных аномалий 9
3.8 Внешний цикл по выраженности заданных аномалий . . . 9

4 Эксперименты **9**

5 Калибровочные границы **11**

6 Выводы и рекомендации **12**

7 Литература **13**

1 Введение

Метод калибровки в анализе качества промышленных изделий представляет собой процесс сравнения измеряемых значений с известными стандартами или эталонами для определения точности и достоверности измерений. Этот метод позволяет проверять соответствие изделий установленным стандартам качества и выявлять любые отклонения от них. Отклонение измеряемых значений от эталонного значения, превышающее некоторую заранее известную величину, означает аномалию, которую следует локализовать и устранить. Метод калибровки хорошо работает в промышленном производстве, когда характеристики изделий — размеры, допуски и т. д. — известны заранее. Однако есть целый ряд ситуаций, когда они неизвестны. Например, при изучении биржевых показателей у ценной бумаги нет эталонной цены: сегодня ее цена такова, а завтра она взлетит, или, наоборот, рухнет. И отклонение цены (выражаясь бухгалтерским языком, волатильность ценной бумаги), которое можно считать нормальным, тоже неизвестно: в одни периоды цены стабильны, а в другие — подвержены высоким колебаниям.

1.1 Теоретическая часть

Любой измеряемый показатель является дискретной случайной величиной, подверженной небольшому нормальному шуму. Мы будем называть калибровочным значением измеряемого показателя его среднее значение за период измерений (в другой период это может быть другое значение), а калибровочной границей мы будем называть величину отклонения случайной величины от калибровочного значения, превышение которой означает аномалию. Если известны как калибровочное значение, так и калибровочная граница, то определение нормы или аномалии —

это детская задача: измеряем величину, вычисляем разность между полученным значением и эталонным, и если по абсолютной величине она меньше калибровочной границы, то это норма, а если больше — то аномалия. Нас будет интересовать ситуация, когда калибровочная граница неизвестна.

1.2 Постановки задачи

1.2.1 Предмет исследования

Предметом нашего исследования будет новый алгоритм поиска калибровочных значений, основанный на вычислении так называемой локтевой точки, которая позволяет отделить величины с большими отклонениями от величин с малыми отклонениями от нормы. Функция поиска локтевой точки так же является авторской, она построена на принципе линеаризации целевой переменной (см. [5]) и успешно применяется нами для решения различных задач.

1.2.2 Методика исследования

Мы генерируем множественные синтетические данные, про которые нам заранее известны все характеристики их анормальности: выраженность аномальных значений и их доля в общем числе наблюдений. Прогоняя синтетические данные через наш алгоритм, мы получаем аномалии, выявленные алгоритмом вслепую, и сравниваем результат с априорно известными нам аномалиями. Условия, при которых результаты близки или совпадают, указывают на условия, в которых наш алгоритм может быть применим.

1.2.3 Цель исследования

Реализовать алгоритм поиска калибровочной границы и указать условия его применения.

1.3 Технологии

1.3.1 Стандартные библиотеки

Для выполнения вычислений и анализа данных мы пользуемся средой `Jupyter Notebook`, которая предоставляет удобные средства для работы с языком программирования Python и его главными библиотеками: `NumPy`, `Pandas`, `sklearn` и `matplotlib`. Благодаря этим инструментам, мы можем эффективно работать с данными, выполнять исследования и визуализировать результаты (см. [1], [2]).

1.3.2 Авторские функции

Мы используем функцию поиска локтевой точки `fold_point`, которая принимает в качестве аргументов два одномерных массива A и B одинаковой длины, причем, массив A выступает в качестве горизонтальной переменной некоторой заданной таблично функции, а массив B — в качестве вертикальной. Функция исполняет алгоритм линеаризации (см. [5]) функции B с разбиением на два участка и построением регрессионной сплит-модели, оптимизируя разбиение по максимуму коэффициента детерминации. Функция `fold_point(A, B)` возвращает номер `fold_num` массива `A`, в котором достигается максимум, а точка с координатами `(A(fold_num), B(fold_num))` — это и есть искомая точка максимума.

2 Генератор синтетических данных

2.1 Константы и параметры генерации

Задаем необходимые константы:

1. `N` — объем выборки,
2. `noise_mean` — среднее значение естественных отклонений, возникающих при измерении показателя,
3. `noise_std` — среднее квадратичное отклонение естественных отклонений,

и параметры генерации синтетических аномалий:

1. `out_coef` — коэффициент, при помощи которого мы будем варьировать выраженность аномалий,
2. `out_pro` — процент аномальных значений в выборке объема `N`.

2.2 Случайный шум

Для генерации синтетического показателя `y`, имитирующего результаты измерений какого-либо реального показателя, воспользуемся методом `normal` из модуля `random` библиотеки `numpy`, который генерирует набор чисел, распределенных по нормальному закону:

```
y = np.random.normal(noise_mean, noise_std, N)
```

2.3 Аномальные отклонения и их позиции

Мы будем варьировать две величины: выраженность аномалии, то есть то, насколько аномальное значение отстоит от нормальных значений,

и процент аномалий в общей выборке. Выраженность аномалии получается при помощи коэффициента, на который умножается естественный шум: если коэффициент нулевой, то сдвига не происходит вообще, если двукратный — то выраженность аномалии весьма слаба, если десятикратный — выраженность высокая. Для этого вводим переменную `out_shift`:

```
out_shift = noise_std*out_coef
```

Далее, выполняем случайную выборку из набора номеров показателя `y` так, чтобы объем этой выборки составлял нужный нам процент `k` от объема `N`. Для этого нужно выбрать

```
k = N*(out_pro/100)
```

элементов, но проблема в том, что процент аномалий `out_pro` не обязан быть целым, а число `N` не обязано делиться на 100. Другими словами, выражение в правой части относится к типу `float`, что противоречит здравому смыслу: мы не можем создавать множество, в котором присутствует дробное число элементов. Поэтому мы переводим полученное значение k в формат `integer`:

```
k = int(k)
```

Теперь из списка номеров `list(np.arange(N))` извлекаем случайным образом `k` номеров при помощи метода `sample` библиотеки `random`:

```
out_num = random.sample(list(np.arange(N)), k)
```

Остается на позиции с номерами из `out_num` записать величины, смещенные на `out_shift`.

2.4 Внедрение аномалий

Проводим цикл по всем номерам из `out_num` и выполняем на каждом шаге смещение на `out_shift`:

```
for i in out_num:
    y[i] = y[i] + out_shift
```

Теперь в массиве y присутствуют аномальные значения. Их число составляет `out_pro` процентов от общего объема, и они смещены относительно своих нормальных положений на `out_shift` (см. рис. 1 и 2).

2.5 Примеры

На первом из рисунков в серии рис. 1 мы использовали

```
out_pro  = 7
out_coef = 0
```

Понятно, что при этом `out_shift = 0`, и аномалий, которые мы помечали желтыми точками, нет вообще. На втором рисунке мы задали

```
out_pro  = 7
out_coef = 3
```

Теперь отклонение аномалий `out_shift` в три раза больше нормального шума `nise_std` и, тем не менее, аномалии выражены настолько слабо, что возникает вопрос: а аномалии ли это? На третьем рисунке значения параметров таковы:

```
out_pro  = 7
out_coef = 8
```

то есть, при восьмикратном увеличении отклонений аномалии становятся выраженными.

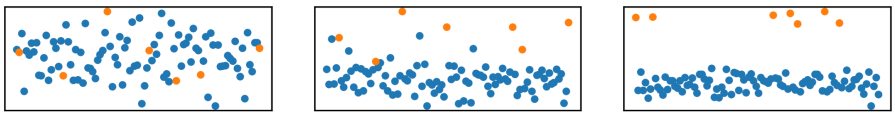

Рис. 1: Примеры распределения аномалий по выраженности

В рисунках серии рис. 2. мы, наоборот, зафиксировали выраженность аномалий и изменяли их процент. На первом рисунке

```
out_pro  = 1
out_coef = 8
```

Поскольку для иллюстрации идеи мы использовали 100 точек, понятно, что мы получили 99 нормальных точек и одну аномальную. На втором рисунке мы увеличили процент аномалий, оставив ту же выраженность

```
out_pro  = 10
out_coef = 8
```

а на третьем — довели процент аномальных точек до 40

```
out_pro  = 40
out_coef = 8
```

в результате чего возник еще один вопрос: являются ли эти точки аномальными, если их так много, и не являются ли они типичными, но относящимися к другому типу?

Мы еще вернемся к этим вопросам в разделе 4.

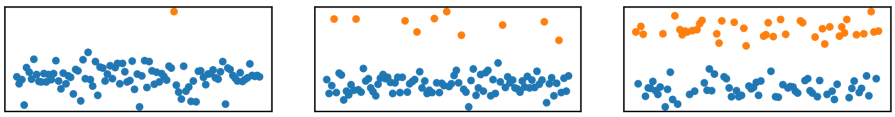

Рис. 2: Примеры распределения аномалий по объему

3 Алгоритм

Мы будем варьировать два параметра: `out_coef`, который определяет степень выраженности аномалии, и `out_pro`, который задает процент аномальных значений, причем коэффициент `out_coef` мы будем изменять в цикле в пределах от 1 до 9, а процент `out_pro` — от 1% до 99% (см. рис. 3 в следующем разделе).

3.1 Установка стартовых значений цикла и генерация данных

Устанавливаем значения констант:

```
N = 1000
noise_mean = 0
noise_std= 0.1
```

и присваиваем параметрам `out_pro` и `out_coef` стартовые значения:

```
out_pro  = 1
out_coef = 1
```

Затем исполняем шаги 2.2–2.4. В результате получаем случайную величину `y` объема `N`, которая содержит 1% выбросов, выраженных с малым коэффициентом выраженности.

3.2 Вычисление отклонений

Пользуясь методами `mean` и `abs`, вычисляем массив отклонений случайной величины `y` от ее среднего значения:

```
fluctation = abs(y.mean() - y)
```

3.3 Агрегация промежуточных результатов

Нам понадобится промежуточная система для хранения и обработки вычисляемых величин. Для этого, пользуясь методом `DataFrame` библиотеки `pandas`, мы создаем пустой датафрейм, в который будем записывать необходимые для дальнейших действий сведения

```
df = pd.DataFrame()
```

Помещаем в него случайную величину y и массив `fluctuation`:

```
df['y'] = y
df['fluctation'] = fluctuation
```

3.4 Сортировка отклонений

Пользуясь методом `sort_values` библиотеки `pandas`, упорядочиваем объекты датафрейма `df` по убыванию величины отклонения и помещаем результат в датафрейм `df_sorted`:

```
df_sorted = df.sort_values(by='fluctation', ascending=False)
```

3.5 Поиск локтевой точки

На предыдущем шаге мы получили массив упорядоченных по убыванию отклонений, элементы которого нумеруются от 0 до `N-1`. Следовательно, мы имеем таблично заданную функцию, к которой можно применить алгоритм поиска локтевой точки. Пользуясь методом `arange` библиотеки `numpy`, формируем массив горизонтальной переменной:

```
A = np.arange(N)
```

Пользуясь методом `to_numpy()` библиотеки `pandas`, переводим в массив столбец дата-фрейма, к котором записаны абсолютные величины вертикальной переменной:

```
B = df_sorted['fluctation'].to_numpy()
```

Затем применяем к массивам `A` и `B` функцию `fold_point`, в результате чего получаем номер `fold_num`, в котором наблюдается локтевая точка массива отклонений:

```
fold_num = fold_point(A, B)
```

3.6 Процент обнаруженных аномалий

Все элементы, которые в массиве B находятся левее номера `fold_num`, являются аномалиями, которые обнаружил алгоритм. Поэтому процент обнаруженных аномалий вычисляется как следующая величина:

```
 out_pro_detected = 100*fold_num/N
```

и у нас возникает возможность сравнить процент заданных аномалий `out_pro` с процентом обнаруженных `out_pro_detected` (см. рис. 3).

3.7 Внутренний цикл по процентам заданных аномалий

Возвращаемся к шагу 3.1, увеличиваем значение `out_pro` на 1 и повторяем шаги 3.2–3.6.

3.8 Внешний цикл по выраженности заданных аномалий

Возвращаемся к шагу 3.1, увеличиваем значение `out_coef` на 1 и повторяем шаги 3.2–3.6.

4 Эксперименты

Исполняя в предыдущем разделе двухуровневый вложенный цикл, мы, по сути, провели серию из `100*10=1000` компьютерных экспериментов. Их цель — сравнить, насколько выявленные аномалии соотносятся с аномалиями, заложенными нами в синтетические данные, а самое главное — установить, при каких условиях это соотношение можно считать приемлемым. Визуализация этих экспериментов представлена на рис. 3.

Мы фиксировали уровень выраженности аномалий `out_coef` (на рисунке это значение, отображенное в легенде) и постепенно увеличивали процент заложенных в синтетические данные аномальных значений. Это равномерное увеличение визуализировано в виде синей линии, которая является биссектрисой первого координатного угла: по горизонтали изменяется процент заданных аномалий, по вертикали — тот же процент. Желтая линия показывает процент обнаруженных аномалий.

Первый рисунок в серии рис. 3 иллюстрирует ситуацию с низкой выраженностью заданных аномалий. В такой ситуации алгоритм не в состоянии отличить нормальные объекты от аномальных, и каким бы ни был процент заданных аномалий, он всегда возвращает один и тот же (с точностью до случайных колебаний) ответ: примерно 15%.

Здесь уместно вспомнить один из вопросов, сформулированных нами на этапе генерации синтетических аномалий: если сгенерированное нами аномальное значение ничем не отличается от нормальных значений (см. раздел 2, первую и вторую позиции рис. 1), то является ли такое значение аномальным? С точно таким же вопросом сталкивается наш алгоритм, и результаты, которые он возвращает в этой ситуации (желтая линия) никак не согласуются с действительностью (синяя линия).

Вместе с тем, с постепенным увеличением коэффициента выраженности аномальных значений, постепенно изменяются и ответы нашего

алгоритма. Рассмотрим ситуацию, проиллюстрированную восьмой (или девятой) позицией рис. 3.

Здесь заложенные в синтетические данные аномалии выражены весьма ярко, с коэффициентом `out_coef = 8`, то есть, отличаются от нормальных значений почти на порядок. Мы видим, что при увеличении процента заложенных аномалий от 1 и выше алгоритм возвращает точные проценты обнаруженных им аномалий, но, начиная примерно с 20%, начинает вести себя неуверенно, теряется, а после 80% вообще возвращает точную противоположность реальному проценту.

Вспомним второй философский вопрос, который мы сформулировали в разделе 2, на этапе генерации синтетических данных: если аномальных значений слишком много, то не являются ли они новой нормой, относящейся к другому типу (см. раздел 2, третью позицию рис. 2)? Именно с этим вопросом сталкивается наш алгоритм примерно после 20% и отвечает на него, как может. В конце концов, после 80% он вообще начинает считать аномалией то, что мы считали нормой, а нормой — заложенные нами в синтетические данные аномалии, и поэтому возвращает противоположные результаты.

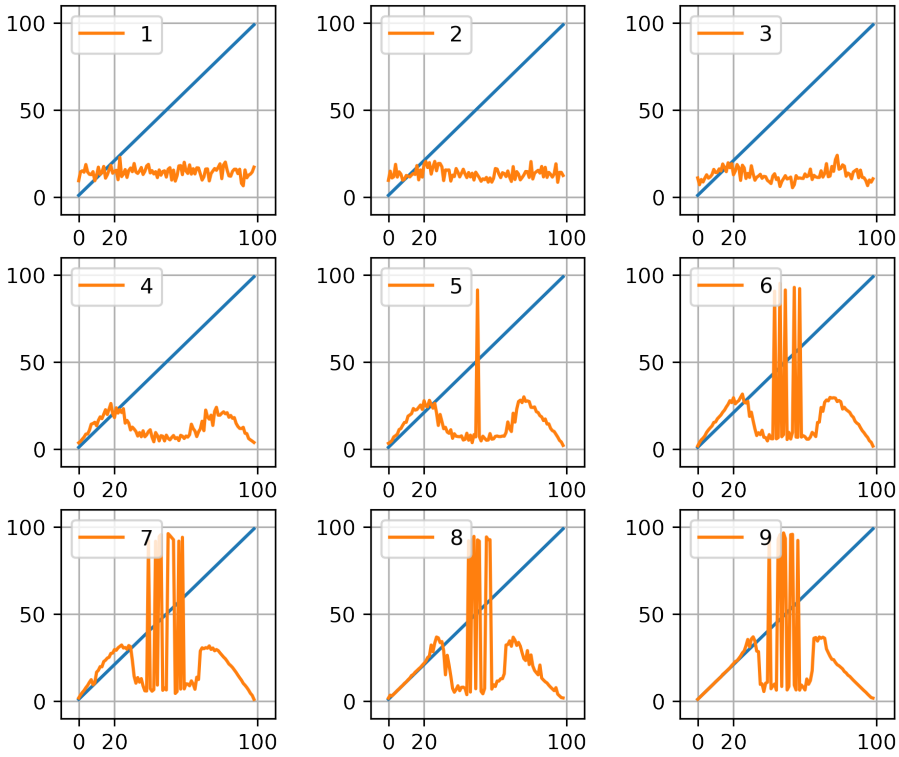

Рис. 3: Синяя — заданный, зеленая — обнаруженный процент

Предварительно можно сказать, что наш алгоритм дает приемлемые

результаты при выполнении следующих условий:

1. аномальные значения отличаются от нормальных значений на порядок и выше,
2. процент аномальных значений не превышает 20%.

5 Калибровочные границы

После того, как в результате серии компьютерных экспериментов мы получили предварительные границы применимости нашего алгоритма, мы можем перейти собственно к процедуре вычисления калибровочных границ. Теперь это сделать очень легко: достаточно после вычисления номера `fold_num` локтевой точки, которое мы провели на шаге 3.5 нашего алгоритма, принять в качестве допустимой границы отклонения величину `bound`:

```
bound = df_sorted['fluctation'].to_numpy()[fold_num]
```

Например, если мы задаем следующие константы

```
 N = 1000
noise_mean = 0
noise_std = 0.1
```

и следующие параметры генерации синтетических данных

```
out_pro  = 15
out_coef  = 6
```

то коэффициент выраженности аномалии недостаточен для приемлемых результатов (см. раздел 4, шестую позицию рис. 3). Действительно, в этом случае алгоритм возвращает в качестве прогнозируемого процента аномалий `out_pro_detected=21.6%`, но это слишком сильно расходится с величиной параметра `out_pro =15%`, которую мы задали при генерации (см. первую позицию рис. 4). При этом возвращаемое алгоритмом значение допустимой границы отклонений, превышение которой означает аномалию, равно `bound=0.2264`, но ему нельзя доверять. Алгоритм отправил в аномальную зону слишком большой процент объектов, поэтому реальная калибровочная граница выше той, которую вернул алгоритм.

С другой стороны, если при тех же, что и выше, значениях констант мы выставим следующие параметры генерации

```
out_pro  = 15
out_coef  = 9
```

то, согласно рассуждениям предыдущего раздела, алгоритм должен вернуть приемлемый результат (см. раздел 4, девятую позицию рис. 3). И действительно, возвращаемый процент аномалий на этот раз не просто близок к заданному проценту аномалий, они буквально совпадают `out_pro_detected=15%` (см. вторую позицию рис. 4) . Поэтому значение калибровочной границы, равное `bound=0.431`, вполне соответствует действительности.

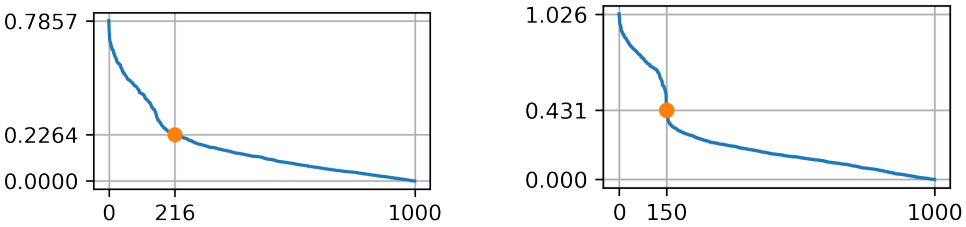

Рис. 4: Примеры калибровочных границ

В качестве комментария к иллюстрациям рис. 4 добавим, что здесь мы визуализировали упорядоченные по убыванию отклонения из серии `df_sorted['fluctation']` (синяя линия) и локтевую точку, которая была найдена при помощи функции `fold_point` (желтая точка).

6 Выводы и рекомендации

Предложенный нами алгоритм не только детектирует аномальные значения, но и самостоятельно устанавливает калибровочные границы, то есть величины отклонений, превышение которых означает аномалию. Это бывает очень важно, особенно в ситуациях, когда и сама измеряемая величина, и ее вариация изменяются с течением времени, например, при измерении биржевых показателей.

Вместе с тем, следует отметить, что наш алгоритм имеет ряд существенных ограничений. Прежде всего, это отмеченные нами выше ограничения на выраженность аномалий и их долю в общем числе наблюдений (см. раздел 4). Однако гораздо более существенным является логическое ограничение, состоящее в том, что для корректного применения нашего алгоритма необходимо заранее знать хотя бы приблизительно то, насколько выражены аномалии в наборе данных и каков их процент.

Если, исходя из каких-либо внешних по отношению к данным соображений, мы заранее знаем, что выраженность аномалий составляет примерно порядок или выше, а их объем не превышает 20% от объема выборки, мы можем быть уверены, что алгоритм вернет приемлемый результат. Для такого знания нужны какие-то (в разных ситуациях —

разные) представления о предметной области, из которой получены данные, что противоречит «чистому» анализу, который ведется вслепую, без апелляции к предметной области. Но практический анализ данных без знания предметной области, видимо, не возможен.

7 Литература

1. Хейдт М. Изучаем Pandas / М. Хейдт; — Москва: ДМК Пресс, 2018. — 438 с.
2. Бурков А. Машинное обучение без лишних слов / А. Бурков; — СПб: Питер, 2020. — 192 с.
3. Вьюгин, В. В. Математические основы теории машинного обучения и прогнозирования / В. В. Вьюгин; — М.: МЦИМО. — 2013. — 387 с.
4. Бринк Х. Машинное обучение / Х. Бринк, Дж. Ричардс, М. Феверолф — СПб.: Питер, 2017. — 336 с.
5. Мосин, В. Г. Линеаризация целевой функции в регрессионных задачах методом сингулярных разложений / В. Г. Мосин // Математическое моделирование : Тезисы II Международной конференции, Москва, 21–22 июля 2021 года. – Москва: Издательство Перо, 2021. – С. 66–67.