

Санкт-Петербургский государственный университет

ТАТЬЯНЕНКО Алексей Дмитриевич

Выпускная квалификационная работа

Байесовский подход к обнаружению разладки

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование
информационных систем»*

Основная образовательная программа *СВ.5162.2021 «Технологии программирования»*

Научный руководитель:
доцент кафедры системного программирования, к.ф.-м.н., В. И. Гориховский

Рецензент:
старший инженер, ООО «Техкомпания Хуавэй», к.ф.-м.н., М. С. Куля

Санкт-Петербург
2025

Saint Petersburg State University

Alexey Tatyanyenko

Bachelor's Thesis

Bayesian approach to change point detection

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5162.2021 "Programming Technologies"*

Scientific supervisor:
docent, Sc.C, V.I. Gorikhovskii

Reviewer:
engineer-researcher at "Huawei Technologies Co.", Sc.C, M.S. Kulya

Saint Petersburg
2025

Оглавление

Введение	5
1. Обзор байесовского подхода к обнаружению разладки и его реализаций	7
1.1. Задача обнаружения разладки	7
1.2. Байесовский подход к обнаружению разладки	9
1.3. PySATL-CPD	12
2. Постановка задачи	14
3. Адаптация байесовского опдхода к обнаружению разладки к прикладному использованию	15
3.1. Рекуррентная оценка длины пробега	15
3.2. Функция выживаемости	17
3.3. Предсказательная модель	17
3.4. Адаптация алгоритма к онлайн-задаче	19
4. Предсказательные модели	21
4.1. Гауссовская предсказательная модель	22
4.2. Экспоненциальная предсказательная модель	23
4.3. Адаптивная предсказательная модель	24
5. Реализация и интеграция алгоритма с эвристиками	27
5.1. Реализация и интеграция онлайн-алгоритма	27
5.2. Предсказательные модели	29
6. Настройка гиперпараметров и численное исследование онлайн-алгоритма	30
6.1. Параметры эксперимента и генерация синтетических данных	30
6.2. Исследование гауссовской предсказательной модели . . .	31
6.3. Исследование адаптивной предсказательной модели . . .	33
6.4. Анализ задержек локализации	35

6.5. Результаты на реальных данных	39
Заключение	42
Список литературы	43

Введение

Анализ временных рядов является быстро развивающейся областью, в которой возникают разнообразные важные задачи, в частности, задача обнаружения разладки. Разладка (change point) — один из видов аномалий во временных рядах, смена распределения данных с одного на некоторое другое (пример приведён на рис. 1). Задача её обнаружения возникает в различных контекстах: анализ зашумлённых бенчмарков [7], анализ поведения туристов [15], физические исследования [13] и т.д. Существуют различные алгоритмы обнаружения разладки, которые классифицируются [3] по необходимости разметки данных, задержке при обнаружении разладки, степени зависимости от параметров и т.д. Алгоритмы имеют свои достоинства и недостатки, а также границы применимости, и из них затруднительно выбрать какой-либо наилучший, что показывают результаты исследований [17]. Для их реализации и исследования разрабатывается PySATL-CPD.¹

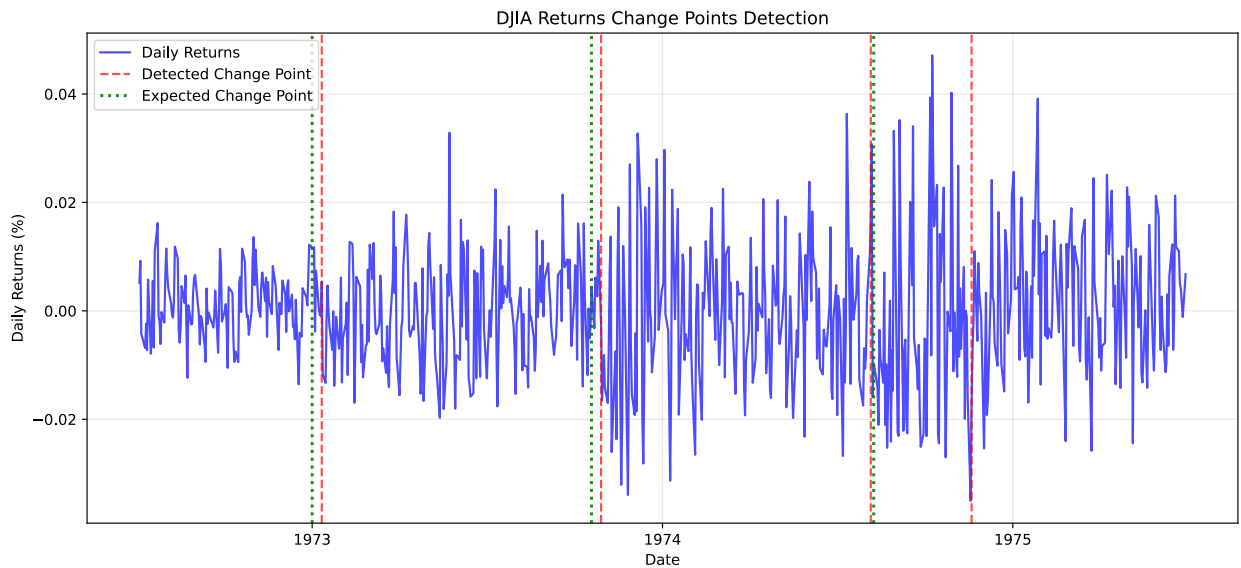


Рис. 1. Разладки в наборе данных 1972-75 DOW JONES RETURNS

Один из подходов к решению задачи обнаружения разладки — байесовский, основанный на пошаговой оценке количества шагов, прошедших с последней разладки, и последующем принятии решения о появлении новой разладки. Идея была предложена в препринте [1] и, хотя

¹<https://github.com/PySATL/pysatl-cpd> — дата обращения 23.05.2025

и не была доведена до конца, оказалась популярной² и послужила основой для разнообразных модификаций [2, 8, 11] исходного алгоритма. Во-первых, такой принцип работы потенциально способен обеспечивать высокую точность. Во-вторых, математически метод позволяет сколь угодно долго принимать новые значения и анализировать их на предмет наличия разладки. При этом в исходной формулировке алгоритма отсутствует явно выделенный критерий наличия разладки. Кроме того, большая ответственность перекладывается на пользователя, который должен подобрать подходящую предсказательную модель с точностью до значения априорных параметров. Наконец, теоретически подходящий для работы с постоянно поступающими новыми данными (онлайн-задача обнаружения разладки) алгоритм на практике сталкивается с линейным ростом затрачиваемых времени и памяти, что делает использование на больших данных затруднительным.

В работе представлена адаптация алгоритма под онлайн-задачу. Далее рассматриваются гауссовская, экспоненциальная и эвристическая адаптивная предсказательные модели, позволяющие оценивать правдоподобие получаемых наблюдений. Особое внимание уделяется моделям со свойством сопряжённости априорного/апостериорного распределений в силу их вычислительной простоты. При реализации были выделены в качестве интерфейсов основные компоненты алгоритма, что обеспечивает гибкую конфигурируемость. Наконец, проводится численное исследование предсказательных моделей, позволяющее получить представление о поведении алгоритма на различных данных и при различной параметризации. На основе этого делаются выводы о допустимых эвристиках (в том числе для борьбы с ростом сложности каждого шага алгоритма). Также для подтверждения работоспособности алгоритма на реальных данных воспроизводятся результаты из исходной работы [1].

²На 24.05.2025 более 1000 цитирований согласно GOOGLE SCHOLAR: <https://tinyurl.com/bocpd-citations>

1. Обзор байесовского подхода к обнаружению разладки и его реализаций

В данном разделе кратко излагаются результаты предыдущих работ [18, 19], посвящённых формулировке, реализации и исследованию байесовского алгоритма обнаружения разладки. Рассматриваются существующие реализации и аналогичные инструменты. Также описываются возможности и особенности совместно разрабатываемого PySATL-CPD, перечисляются уже реализованные алгоритмы.

1.1. Задача обнаружения разладки

Разладка (change point) — один из видов аномалий во временных рядах, точка смены одного распределения данных на другое (при этом могут меняться параметры распределения или даже семейство). Задача её обнаружения:

$$\begin{aligned} &\{y_i\}, i = 1, \dots, n \\ &H_0: y_i \sim F_0, i = 1, \dots, n \\ &H_a: \exists \tau : 1 \leq \tau < n, y_i \sim \begin{cases} F_1, & i > \tau, \\ F_0 & \text{иначе.} \end{cases} \end{aligned}$$

Тогда τ — точка разладки.

Поскольку установить точное местоположение разладки зачастую затруднительно, для задачи **корректной локализации** рассмотрим

найденные разладки с отклонением не более k от размеченной:

$$\{x_i\}, i = 1, \dots, n$$

$$H_0: x_i \sim F_0, i = 1, \dots, n$$

$$H_1: 1 \leq \tau_1 < n \Rightarrow \exists \tau \in [\tau_1 - k; \tau_1 + k], x_i \sim \begin{cases} F_1, i > \tau \\ F_0, \text{ иначе} \end{cases}$$

τ_1 — корректно распознанная с точностью k разладка.

Это статистическая постановка задач, позволяющая проводить статистическое исследование на синтетических данных. С алгоритмической точки зрения, однако, стоит отметить несколько важных нюансов. Во-первых, если говорить об онлайн-задаче, то в ней алгоритму никогда не известны все данные, поскольку всегда продолжают поступать новые наблюдения, в отличие от оффлайн-задачи, где известны сразу все данные. Во-вторых, в данных возможно наличие множества разладок. В оффлайн-задаче нужно по всему набору данных определить, сколько разладок было, и все их локализовать, а в онлайн-задаче требуется как можно быстрее обнаружить новую разладку и, если она случилась, локализовать её.

В прошлых семестрах [18] работа фокусировалась на статистической постановке задачи и исследовании результатов на синтетических данных. Однако для онлайн-задачи требовалось провести адаптацию алгоритма, добавив в него дополнительные эвристики. Это, в частности, поспособствовало исследованию на реальных данных, где зачастую было более одной разладки.

Задача обнаружения разладки встречается в различных областях: анализ зашумлённых бенчмарков [7], анализ поведения туристов [15], физика [13], финансы, робототехника и др. [3] При этом контекст её возникновения можно условно разделить на две группы случаев:

- анализ накопленных данных с целью получения некоторой новой информации (т.н. «data mining»);
- детектирование изменений состояния в процессе работы (напри-

мер, анализ температуры и частоты работы процессора при бенчмаркинге).

Также есть смысл рассматривать обнаружение разладки как две различные задачи: определения её наличия (детектирование разладки) и определение её местоположения (локализация разладки). В силу статистической природы задачи во втором случае может потребоваться рассмотрение не точного соответствия, а некоторого допустимого отклонения от истинной точки при локализации.

Существуют различные подходы к решению задачи обнаружения разладки, классифицируемые [3] по необходимости разметки данных, задержке при обнаружении разладки, степени зависимости от параметров и т.д. В разных случаях лучшие результаты могут демонстрировать разные алгоритмы [17]. Например, из существующих подходов можно назвать графовый [10], классификационный [14], основанный на отношении плотностей [6], основанный на определении подпространств [5], ядерный [16], байесовский [1] и другие.

1.2. Байесовский подход к обнаружению разладки

В предыдущих работах были сформулированы различные варианты задачи обнаружения разладки (**определение её наличия, корректная локализация**) и рассмотрены различные случаи её возникновения. Также был предложен байесовский алгоритм обнаружения разладки на выборке фиксированного размера, основной идеей которого является оценка количества шагов с последней разладки (для чего используется термин «длина пробега»). Он является развитием предложенного в статье [1] алгоритма. Сильной стороной алгоритма является его потенциально высокая точность, поскольку он нацелен на анализ не только возникновения разладки, но и её конкретного местоположения.

В базовой версии алгоритма в качестве предсказательной модели было решено использовать гауссовскую функцию правдоподобия, основанную на нормальном-обратном-гамма сопряжённом распределении. Преимущество использования сопряжённых априорных/апостери-

орных распределений — возможность обновления параметров в явном виде без таких вычислительно трудоёмких действий, как интегрирование и решение оптимизационных задач. Пороги детектора были оценены для нормального распределения в ходе исследования и далее зафиксированы. В качестве априорной вероятности появления разладки по прошествии времени была использована постоянная функция правдоподобия, соответствующая экспоненциальному случаю.

Предварительные исследования показали [18], что байесовский подход позволяет обнаруживать разладку в ситуациях как перехода из нормального состояния в другое (разлаженное), так и возвращения из разлаженного состояния в нормальное. Такой режим использования алгоритма назовём штатным. Однако гауссовская предсказательная модель может иногда обнаруживать разладку и в прочих случаях (нештатный режим). Например, при переходе с бета-распределения на экспоненциальное. Но есть ситуации, в которых гауссовская предсказательная модель даёт нестабильные результаты (из экспоненциального в Вейбулл). *Поэтому подход требует расширения множества предсказательных моделей, при котором будут покрыты все основные случаи.*

Существующие реализации на RYTHON³⁴⁵⁶⁷⁸ байесовского алгоритма являются, скорее, исследовательскими прототипами, поскольку они:

- не входят в состав более крупных и универсальных инструментов;
- не предоставляют (как и исходная статья, где был предложен алгоритм) явного критерия определения разладки, а лишь позволяют вычислять вероятность этого;
- не используют преимущества алгоритма для онлайн-режима работы и не решают сопутствующие проблемы роста на каждом шаге

³<https://github.com/naspli/BOCD> — дата обращения 22.05.2025

⁴https://github.com/hildensia/bayesian_changepoint_detection — дата обращения 22.05.2025

⁵<https://github.com/dtolpin/bocd> — дата обращения 22.05.2025

⁶<https://github.com/gwgundersen/bocd> — дата обращения 22.05.2025

⁷<https://github.com/hannawallach/changepoint-detection> — дата обращения 22.05.2025

⁸<https://github.com/y-bar/bocd> — дата обращения 22.05.2025

требуемых времени и памяти;

- зачастую не имеют документации, unit-тестов, подходящей лицензии и других важных для практического использования вещей.

Ещё одним популярным инструментом является RBEAST⁹, являющийся PYTHON-обёрткой над реализацией на R. Он предоставляет широкую функциональность для анализа, но не поддерживает работу в онлайн-режиме. Также стоит отметить проект¹⁰, предоставляющий PYTHON-обёртку над реализацией на RUST. Такое решение может иметь преимущество по производительности, но, как минимум, в нём не предусмотрена возможность выбора не константной функции выживаемости, а добавление новых предсказательных моделей требует расширения списков реализаций вместо имплементации интерфейса. Кроме того, оба инструмента приводят к проблеме двух языков, что может затруднить самостоятельное использование и расширение функциональности. *Таким образом, реализация предложенного алгоритма пока не была доведена до стабильного инструмента, поэтому было решено его разработать.*

При этом стоит отметить, что сам по себе байесовский подход в том числе для решения задачи обнаружения разладки активно используется в библиотеке PyMC¹¹. Однако сам алгоритм там выбран другой, и он требует априорного предположения о данных до и после разладки с точностью до конкретных значений параметров.

Ключевая идея байесовского алгоритма получила развитие в различных его модификациях. В статьях [11] и [2] рассматриваются более тяжеловесные, но менее зависимые от параметризации предсказательные модели. При этом классический алгоритм с гауссовской предсказательной моделью в целом показал себя в бенчмарке [17] лучше, чем эти модификации. В работе [8] же предлагается эвристика для снижения производительности ценой снижения точности вычислений, поскольку хранятся вероятности для фиксированного числа длин пробега. Такой

⁹<https://github.com/zhaokg/Rbeast> — дата обращения 23.05.2025

¹⁰<https://github.com/promised-ai/changepoint> — дата обращения 23.05.2025

¹¹<https://github.com/pymc-devs/pymc> — дата обращения 09.05.2025

подход также затрудняет учёт отброшенных длин пробега внутри предсказательной модели и функции выживаемости.

Таким образом, несмотря на сильные стороны классического алгоритма, нет широко используемых и проработанных его реализаций, тем более в составе более крупных и универсальных инструментов. Также в этих реализациях не уделяется должного внимания устранению недостатков байесовского подхода или пробелов в нём, таких как сильная зависимость от исходных параметров, принятие решения о наличии разладки на основе полученных оценок и рост сложности по мере работы. *Для того, чтобы устранить эти изъяны, требуется использование новых эвристик.* Наконец, байесовский подход позволяет оценить и промоделировать лишь гиперпараметры. Поскольку работа алгоритма всё ещё зависит от выбора значений оставшихся параметров, *требуется настройка их значений по результатам численного исследования на синтетических данных.*

1.3. PySATL-CPD

Работа над реализацией и исследованием байесовского алгоритма ведётся в рамках проекта PySATL-CPD, предоставляющего фреймворк для использования и исследования алгоритмов обнаружения разладки. Фреймворк создаётся в связи с отсутствием инструментов, предоставляющих достаточно широкий спектр алгоритмов, возможность их исследования и расширяемость.

Было найдено исследование алгоритмов [17], имеющее исходный код¹². В нём рассматривается довольно много алгоритмов, однако эта работа не является фреймворком и не имеет инженерной поддержки добавления новых алгоритмов.

Среди крупных инструментов для обнаружения разладки стоит отметить RUPTURES¹³, ROERICH¹⁴ и CLASPY¹⁵. Они предоставляют некото-

¹²<https://www.turing.ac.uk/news/publications/evaluation-change-point-detection-algorithms> — дата обращения 09.05.2025

¹³<https://github.com/deepcharles/ruptures> — дата обращения 09.05.2025

¹⁴<https://github.com/HSE-LAMBDA/roerich> — дата обращения 09.05.2025

¹⁵<https://github.com/ermshaua/claspy> — дата обращения 09.05.2025

рый набор алгоритмов, но среди них, например, нет байесовских. Также они в целом не поддерживают работу в онлайн-режиме (то есть не позволяют обрабатывать новые данные по мере их поступления сколько угодно долго).

Данная работа посвящена байесовскому подходу, но помимо него PySATL-CPD также включает в себя другие алгоритмы:

- барьерные классификационные;
- основанные на отношении плотностей;
- графовые (построенные на дистанционных и kNN -графах).

В проекте используется линтер RUFF, аннотация типов проверяется с помощью MYPY, тестирование проводится с помощью PYTEST (с расчётом покрытия тестами), в том числе всё вышеперечисленное используется в CI. Управление зависимостями осуществляется с помощью POETRY.

Таким образом, несмотря на существование нескольких крупных инструментов для обнаружения разладки, они все имеют существенные недостатки. Также во всех из них не предоставляется рассматриваемый в данной работе алгоритм. Важным аспектом использования алгоритмов обнаружения разладки является возможность сравнить их и оценить применимость в тех или иных условиях. Хотя для этого и проводились исследования с прилагающимся кодом, гибкость такого подхода оказывается ограниченной по сравнению с более автоматизированными фреймворками. Всё это является аргументом в пользу развития проекта PySATL-CPD.

2. Постановка задачи

Целью работы является реализация онлайн-версии байесовского алгоритма, расширение списка предсказательных моделей (в том числе добавление адаптивной версии), настройка параметров алгоритма и оценка его эффективности. Для её выполнения были поставлены следующие задачи:

1. Адаптация алгоритма под задачу онлайн-обнаружения разладки.
2. Реализация онлайн-версии байесовского алгоритма и соответствующей функциональности для интеграции в PySATL-CPD.
3. Добавление альтернативных предсказательных моделей: экспоненциальной и адаптивной.
4. Настройка параметров и оценка эффективности онлайн-алгоритма.

3. Адаптация байесовского подхода к обнаружению разладки к прикладному использованию

В данном разделе описывается байесовский подход и используемые в нём построения оценок. В частности, рассматриваются составляющие его компоненты: функция выживаемости и предсказательная модель. Общие принципы и константная функция выживаемости были взяты из оригинальной работы [1]. В подразделе 3.1 приводится сформулированное в ходе работы определение байесовского алгоритма обнаружения разладки. В подразделе 3.3 предлагается добавить в алгоритм этап обучения предсказательной модели. Наконец, в подразделе 3.4 конце описывается адаптация алгоритма к онлайн-задаче обнаружения разладки.

3.1. Рекуррентная оценка длины пробега

Ключевым понятием для байесовского подхода к обнаружению разладки является *длина пробега* — количество шагов с последней разладки. Возможность её рекуррентной оценки позволяет декомпозировать алгоритм на непосредственно распределение длины пробега, предсказательную модель (в частном случае — функцию правдоподобия) и функцию выживаемости. Представим вероятность длины пробега с учётом прошлых данных как $P(r_t | \mathbf{x}_{1:t}) = P(r_t, \mathbf{x}_{1:t}) / P(\mathbf{x}_{1:t})$.

Совместную вероятность можно разложить в сумму вида

$$\begin{aligned} P(r_t, \mathbf{x}_{1:t}) &= \sum_{r_{t-1}} P(r_t, r_{t-1}, \mathbf{x}_{1:t}) = \\ &= \sum_{r_{t-1}} P(r_t, x_t | r_{t-1}, \mathbf{x}_{1:t-1}) P(r_{t-1}, \mathbf{x}_{1:t-1}) = \\ &= \sum_{r_{t-1}} P(r_t | r_{t-1}) P(x_t | r_{t-1}, \mathbf{x}_t^{(r)}) P(r_{t-1}, \mathbf{x}_{1:t-1}). \end{aligned}$$

Поскольку размер длины пробега ограничен текущим временем ра-

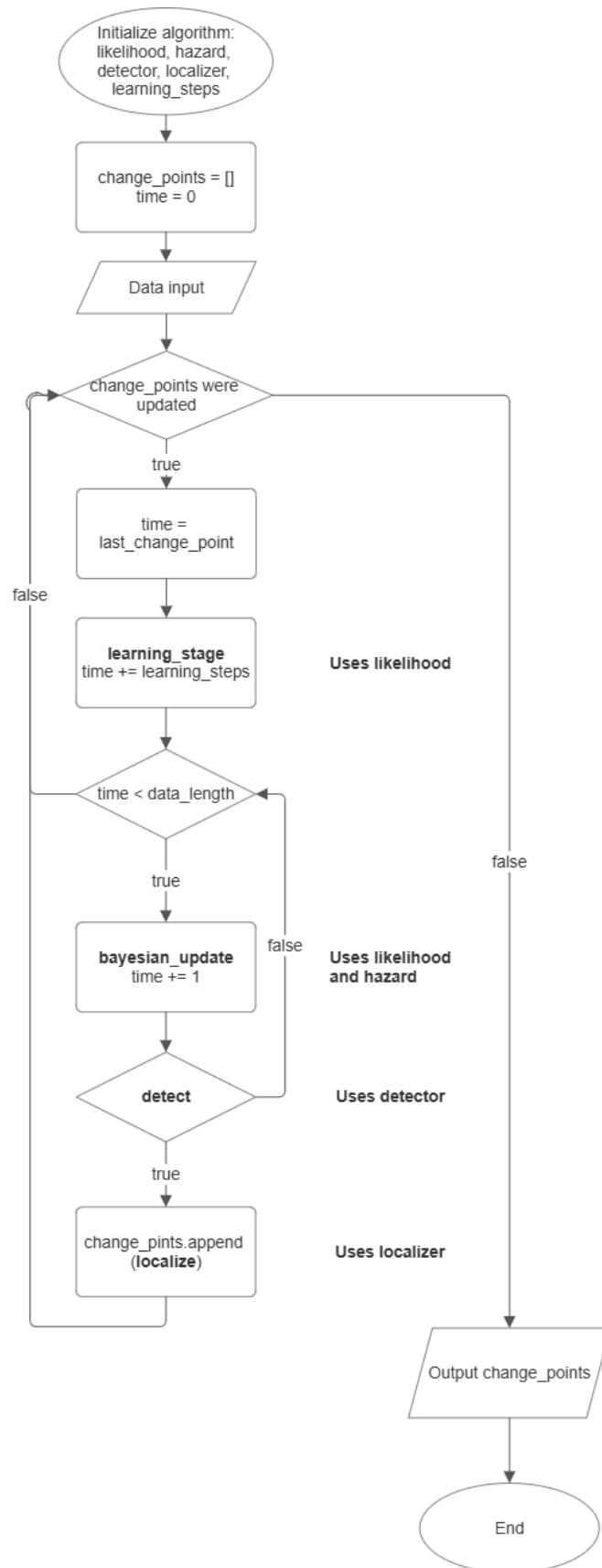


Рис. 2. Байесовский алгоритм обнаружения разладки

боты алгоритма, сумма вероятностей всегда будет вычисляться по конечному носителю. Это позволяет представить распределение длины пробега как массив. В таком случае получаем более строгую формулировку алгоритма (рис. 2), рассмотренную в прошлых работах [18].

3.2. Функция выживаемости

Функция выживаемости (hazard function) позволяет задать априорную вероятность обнуления длины пробега (т.е. появления разладки) на каждом шаге:

$$H(\tau) = \frac{P_{\text{gap}}(g = \tau)}{\sum_{t=\tau}^{\infty} P_{\text{gap}}(g = t)}, \quad (1)$$

где $P_{\text{gap}}(g)$ — дискретное априорное распределение на промежутке между разладками.

Функция выживаемости задаётся отдельно и параметризует алгоритм. Это также подразумевает, что она имеет некоторый явный вид, поэтому нет необходимости считать сумму ряда в знаменателе формулы 1. Тогда условная априорная вероятность появления разладки:

$$P(r_t | r_{t-1}) = \begin{cases} H(r_{t-1} + 1) & \text{если } r_t = 0, \\ 1 - H(r_{t-1} + 1) & \text{если } r_t = r_{t-1} + 1, \\ 0 & \text{иначе.} \end{cases}$$

Как частный случай, если $P_{\text{gap}}(g)$ — дискретное экспоненциальное, то функция выживаемости — константа: $H(\tau) = 1/\lambda$.

3.3. Предсказательная модель

Предсказательная модель (prediction model) оценивает правдоподобие принадлежности данных распределению с оценёнными параметрами. Также, рассматривая её как плотность, мы можем оценить вероятность наблюдения с учётом прошлых данных: $P(x_t | r_{t-1}, \mathbf{x}_t^{(r)})$, где $\mathbf{x}_t^{(r)}$ — вектор наблюдений для текущего пробега; назовём это **предска-**

зательной вероятностью.

Хорошим частным случаем предсказательной модели является **функция правдоподобия с сопряжённым априорным/апостериорным распределениями**. Она позволяет обновлять параметры в явном виде по выведенным формулам вместо таких вычислительно трудоёмких действий, как интегрирование и решение оптимизационных задач. Более того, количество таких операций растёт линейно с каждым шагом, поэтому подобная оптимизация оказывается критически важной.

Байесовский подход подразумевает наличие некоторых априорных предположений о данных, с которыми работает алгоритм. Это может создавать сложности в случаях, когда мы ожидаем от инструмента большей автономности при работе и не хотим ограничивать себя параметризацией слишком сильно. Как минимум, внутри зафиксированного класса распределений функции правдоподобия позволяют оценивать по обучающей выборке априорные значения параметров. Несмотря на то, что предсказательные модели способны корректировать своё состояние, плохой выбор априорных гиперпараметров может привести к большому количеству ложно положительных разладок. Для борьбы с этой проблемой может быть в алгоритм может быть добавлен **этап обучения предсказательной модели**, наступающий в начале его работы и после каждой разладки. Случаи с неизвестным семейством распределений оказываются уже более сложными и требуют специальных эвристик. Такие модификации алгоритма способны расширить его применимость.

Стоит отметить, что функции правдоподобия со свойством сопряжённости существуют не для всех распределений и в целом покрывают довольно малую часть возможных случаев. Поэтому важным является исследование поведения алгоритма в случаях, для которых у нас нет теоретических гарантий, поскольку он всё ещё может сохранять общую работоспособность ценой некоторого падения качества результатов. Более подробно предсказательные модели рассматриваются в разделе 4.

В оригинальной работе не приводилось детального описания пред-

сказательных моделей, но указывалось, что стоит обратить внимание на экспоненциальные-сопряжённые модели. В целом, для описанных в разделе 4 гауссовской и предсказательной модели использовались известные формулы, однако важно учитывать специфику их использования: каждой длине пробега необходимо сопоставлять только соответствующие наблюдения и значения гиперпараметров. Для гауссовской модели были найдены адаптированные под задачу формулы [4]. Эвристика выбора предсказательной модели была предложена в рамках представленной работы.

3.4. Адаптация алгоритма к онлайн-задаче

Как уже ранее упоминалось, онлайн-алгоритм должен позволять пошагово обрабатывать данные по мере их поступления, принимать решение о наличии разладки и поддерживать работу на больших временных рядах. Для этого требуется адаптация предложенного в исходной статье [1] алгоритма.

В дополнение к уже описанной оценке вероятности появления разладки, необходимо также добавить этап принятия решения. Для этого предлагается ввести **детектор**, отвечающий за определение наличия разладки (например, порог на вероятность максимальной длины пробега или на падение этой вероятности за шаг), и **локализатор**, отвечающий за определение местоположения разладки (например, наиболее вероятная длина пробега за вычетом максимальной). Это существенно для онлайн-алгоритма, поскольку состояние алгоритма на будущих шагах зависит от принятия решения о его изменении на текущем.

При обнаружении разладки удалять все предшествующие ей наблюдения и соответствующие вероятности, поскольку данные до и после разладки по предположению независимы, а доступ алгоритм имеет лишь к данным после начала моделирования длины пробега. Тогда в данных с регулярными разладками не будет происходить неограниченного роста времени и памяти, затрачиваемых на каждый шаг.

В случае длительного отсутствия в данных разладки предлагается

введение эвристики, параллельно основному алгоритму запускающей вспомогательный и обеспечивающее переключение между ними через некоторое время (рис. 3). Важно заметить, что это ведёт к потере всей информации до начала работы вспомогательного алгоритма, поэтому возможно ухудшение результатов. Однако такая эвристика превращает алгоритм из квадратичного по памяти и времени в линейный на достаточно больших входных данных. При этом на этапе подготовки вспомогательного алгоритма возвращается всегда результат работы основного, поскольку он содержит в себе больше информации. В случае, если была найдена разладка, обучение второго алгоритма прерывается, а отсчёт начинается заново (рис. 4). Возможны варианты с более тонкой настройкой, завершающие обучение вспомогательного алгоритма и переключающиеся на него. Это может сэкономить время за счёт уже проведённого обучения, однако требует, чтобы второй алгоритм начал работу не ранее разладки и ведёт к возможному пропуску данных между разладкой и началом подготовки вспомогательного алгоритма. Поэтому в текущем виде было решено оставить обработку разладки основным алгоритмом по тому же принципу, что и в используемом байесовском алгоритме.

From the heuristic's perspective it always works with the main algorithm and sometimes copies the prepared supporting instance, making it the main one

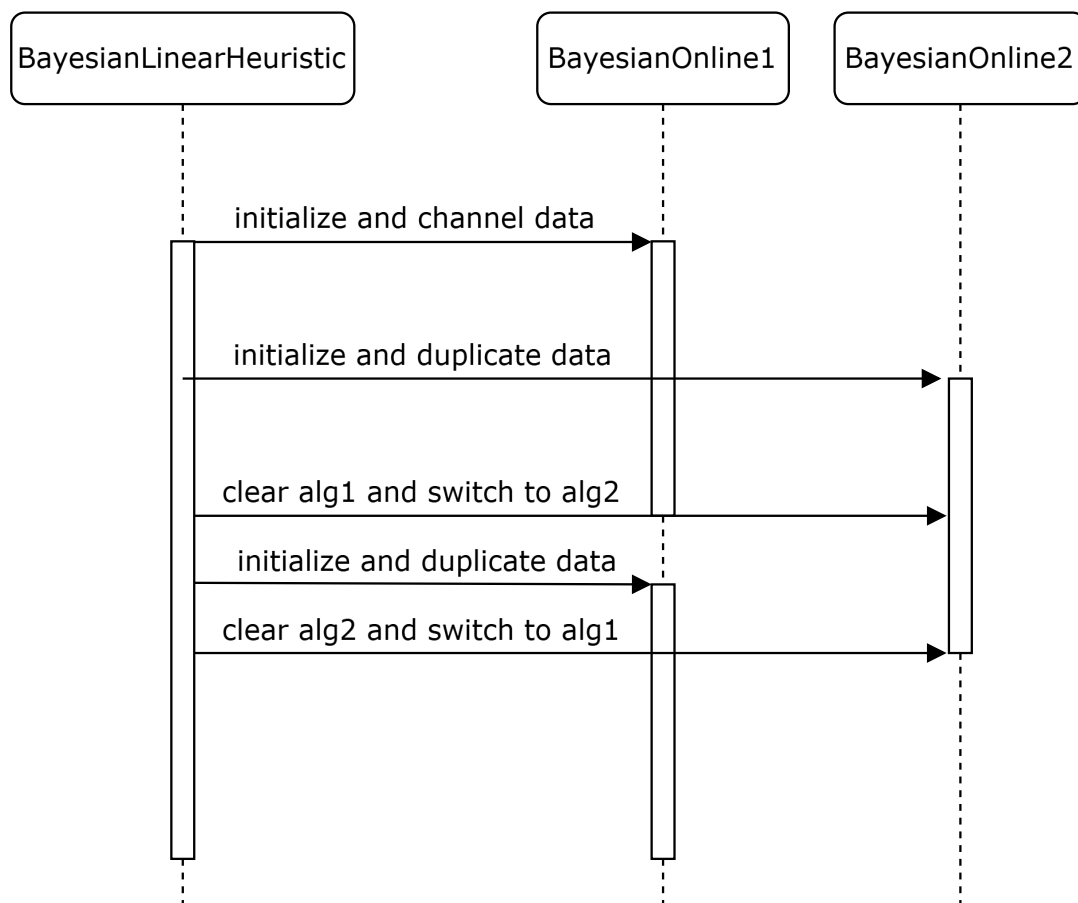


Рис. 3. Общий принцип работы линейной эвристики

4. Предсказательные модели

Как уже ранее упоминалось, предсказательные модели позволяют оценивать правдоподобие принадлежности наблюдения к распределению с оценёнными параметрами (при оценке в том числе используются прошлые наблюдения). Поскольку количество разладок заранее не известно и происходить они могут между разнообразными распределениями, в алгоритм требуется добавить **этап обучения предсказательной модели** для получения априорных значений гиперпараметров.

В данном разделе более подробно рассматриваются предсказательные модели, основанные на гауссовской и экспоненциальной функциях правдоподобия с сопряжёнными априорными/апостериорными распределениями. Приводятся формулы для исходной оценки априорных ги-

From algorithm's perspective it always works with the main algorithm and sometimes copies the prepared supporting instance, making it main

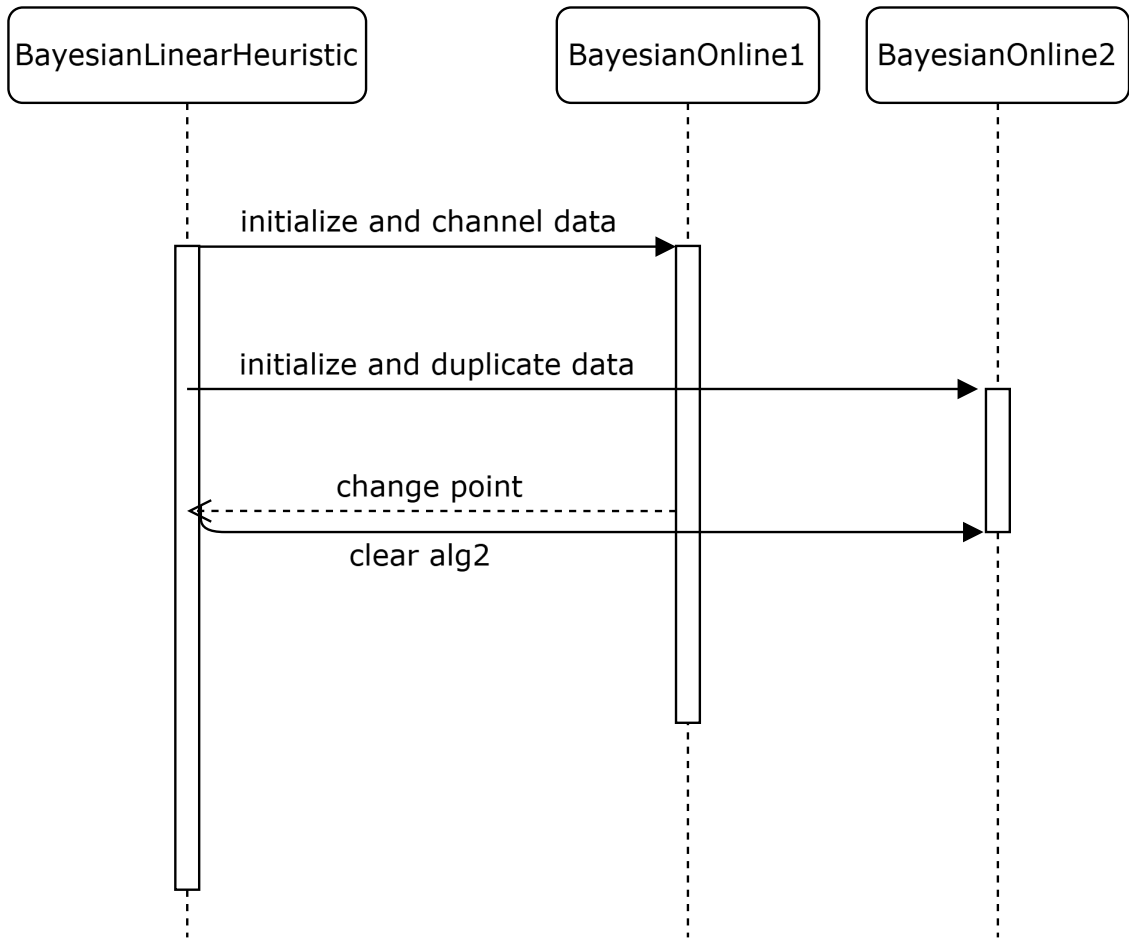


Рис. 4. Обработка разладки линейной эвристикой

перпараметров на обучающей выборке, обновления априорных гиперпараметров до апостериорных и вычисления предсказательной вероятности для полученного наблюдения. В конце рассматривается, как на основе этих моделей можно сделать наивную эвристику для адаптации к различным данным.

4.1. Гауссовская предсказательная модель

При реализации базовой версии алгоритма первой появилась гауссовская предсказательная модель, использующая сопряжённое нормальное-обратное-гамма распределение [9].

- Обучение гиперпараметров:

$$\mu_0 = \bar{x}^{learn}$$

$$k_0 = n$$

$$\alpha_0 = n/2$$

$$\beta_0 = \sum_{i=1}^n \frac{(x_i^{learn} - \bar{x}^{learn})^2}{2}$$

- Обновление гиперпараметров:

$$\mu_{t+1} = \frac{k_t \mu_t + x_t^{obs}}{k_t + 1}$$

$$k_{t+1} = k_t + 1$$

$$\alpha_{t+1} = \alpha_t + 1/2$$

$$\beta_{t+1} = \beta_t + \frac{k_t}{k_t + 1} \frac{(x_t^{obs} - \mu_t)^2}{2}$$

- Предсказательная вероятность:

$$P(x_{t+1}^{obs} | r_t, \mathbf{x}_{t+1}^{(r)}) = t_{2\alpha_{t+1}} \left(x_{t+1}^{obs} | \mu_{t+1}, \frac{\beta_{t+1}(k_{t+1} + 1)}{k_{t+1}\alpha_{t+1}} \right)$$

4.2. Экспоненциальная предсказательная модель

Поскольку гауссовская предсказательная модель показывала себя хуже в случае экспоненциального распределения и распределения Вейбулла для синтетических данных, было решено добавить другую предсказательную модель. Вероятной причиной здесь была асимметричность распределения (причём его носитель существенно отличается от нормального распределения) и то, что гауссовской предсказательной моделью многие большие значения считывались как выбросы. Несмотря на то, что распределение Вейбулла является более общим по отношению к экспоненциальному, для его функции правдоподобия не существует [12] сопряжённого распределения, моделирующего сразу оба параметра. Поэтому выбор был сделан в пользу экспоненциальной предсказательной модели:

- Обучение гиперпараметров:

$$\alpha_0 = n;$$

$$\beta_0 = \sum_{i=1}^n x_i^{learn}$$

- Обновление гиперпараметров:

$$\alpha_{t+1} = \alpha_t + 1$$

$$\beta_{t+1} = \beta_t + x_t^{obs}$$

- Предсказательная вероятность:

$$P(x_{t+1}^{obs} | r_t, \mathbf{x}_{t+1}^{(r)}) = \text{Lomax}(x_{t+1}^{obs} | \beta_{t+1}, \alpha_{t+1})$$

4.3. Адаптивная предсказательная модель

Выбор фиксированной модели на этапе конфигурации алгоритма является достаточно сильным ограничением. Во-первых, это требует априорного предположения о распределении данных, что не всегда доступно пользователю. Во-вторых, это ограничивает нас в том, какие распределения данных мы можем обрабатывать (например, случай смены нескольких семейств распределений в данных с несколькими разладками). Однако свойство сопряжённости, которое желательно сохранять в целях сокращения вычислительной трудоёмкости, накладывает существенные теоретические ограничения на пути к обобщению предсказательной модели. Поэтому при работе было принято решение сосредоточиться на эвристиках, позволяющих выбирать из уже существующих моделей наиболее подходящую.

Для выбора между моделями требуется способ их сравнения. Осуществлять выбор можно на этапе обучения предсказательной модели в добавок к уже проводящейся там оценке априорных гиперпараметров. В качестве наивной эвристики предлагается на этапе обучения создать

несколько моделей (как минимум, две), оценить для каждой из них априорные значения гиперпараметров и с помощью уже известной для каждой из моделей формулы предсказательной вероятности оценить правдоподобие этих гиперпараметров на всё той же обучающей выборке, т.е. сравнить, какая из оценок оказалась лучше. Поскольку наблюдения предполагаются независимыми и одинаково распределёнными, мы можем оценить правдоподобие выборки как произведение правдоподобий каждого из наблюдений. В блоке Алгоритм 1 приводится псевдокод этой эвристической адаптивной модели.

Стоит учитывать, что при большом размере обучающей выборки обе оценки правдоподобия могут пострадать от вычислительных ошибок, оказавшись слишком малыми числами. В ходе исследования на выборках размеров 5, 10, 20 и 50 эта проблема не проявилась.

В данной работе приводится лишь наивная эвристика, поскольку более аккуратный подбор предсказательной модели является отдельной большой задачей. Для оптимального выбора модели в ней могут использоваться критерии согласия, но стоит учитывать и возможные сложности, возникающие при таком подходе. Например, даже если ни одна из моделей не описывает распределение в точности, всё ещё необходимо выбрать из них наиболее подходящую. При этом дальнейшие изменения в этой области не затрагивают структуру алгоритма, что хорошо с инженерной точки зрения.

Algorithm 1 Heuristic Adaptive Model

```
1: function LEARN(array[float] LearningSample)
2:   GAUSSIANCONJUGATE.LEARN(LearningSample)
3:   EXPONENTIALCONJUGATE.LEARN(LearningSample)
4:
5:   GaussianPrior  $\leftarrow$  GAUSSIANCONJUGATE.PRIORPROBABILITY()
6:   ExponentialPrior  $\leftarrow$  EXPONENTIALCONJUGATE.PRIORPROBABILITY()
7:
8:   if GaussianPrior  $\geq$  ExponentialPrior then
9:     SelectedModel  $\leftarrow$  GAUSSIANCONJUGATE
10:  else
11:    SelectedModel  $\leftarrow$  EXPONENTIALCONJUGATE
12:  end if
13: end function
14:
15: function UPDATE(float Observation)
16:   return SelectedModel.UPDATE(Observation)
17: end function
18:
19: function PREDICT(float Observation)
20:   return SelectedModel.PREDICT(Observation)
21: end function
22:
23: function CLEAR
24:   SelectedModel  $\leftarrow$  NONE
25: end function
```

5. Реализация и интеграция алгоритма с эвристиками

В данном разделе описывается интеграция байесовского алгоритма обнаружения разладки в PySATL-CPD. Приводится описание внешней функциональности, предоставляющей интерфейс для использования алгоритма. Далее рассматривается реализация и интеграция предсказательных моделей.

5.1. Реализация и интеграция онлайн-алгоритма

В прошлых семестрах был создан прототип байесовского алгоритма для выборок фиксированного размера¹⁶. Также в проекте PySATL-CPD уже существовала функциональность для получения входных данных и запуска алгоритма на них. Чтобы интегрировать в него онлайн-алгоритм, потребовалось создать новые интерфейсы и по возможности унифицировать их с уже существующей функциональностью¹⁷:

1. Был добавлен интерфейс `OnlineAlgorithm`, имеющий методы для детекции (по наблюдению из временного ряда возвращает, была ли разладка) и для локализации (по наблюдению из временного ряда возвращает в случае наличия разладки, когда она была).
2. Был создан класс `OnlineCpdCore`, который объединяет в себе онлайн-алгоритм и объект, отвечающий за доступ к данным (`DataProvider`), представляя алгоритм как итератор над данными, что соответствует его задуманному поведению.
3. Был создан класс `OnlineCpdSolver`, который инкапсулирует в себе запуск алгоритма на выбранных данных и возвращения совокупного результата по завершении работы. В случае локализации данные возвращаются обёрнутыми в класс

¹⁶<https://github.com/PySATL/pysatl-cpd/pull/15> — nick: alexdtat, дата обращения 24.05.2025

¹⁷<https://github.com/PySATL/pysatl-cpd/pull/39> — nick: alexdtat, дата обращения 24.05.2025

`CpdLocalizationResults`, предоставляющем некоторые возможности для базового анализа результатов (в том числе для размеченных данных).

4. Был добавлен интерфейс `ICpdSolver`, описывающий общее поведение `OnlineCpdSolver` и уже существовавшего `CpdSolver`. Несмотря на то, что разные классы алгоритмов имеют разное поведение, при рассмотрении итоговых результатов их работы на данных фиксированного размера эта разница несущественна, а общий интерфейс позволит в дальнейшем унифицировать анализ этих результатов.

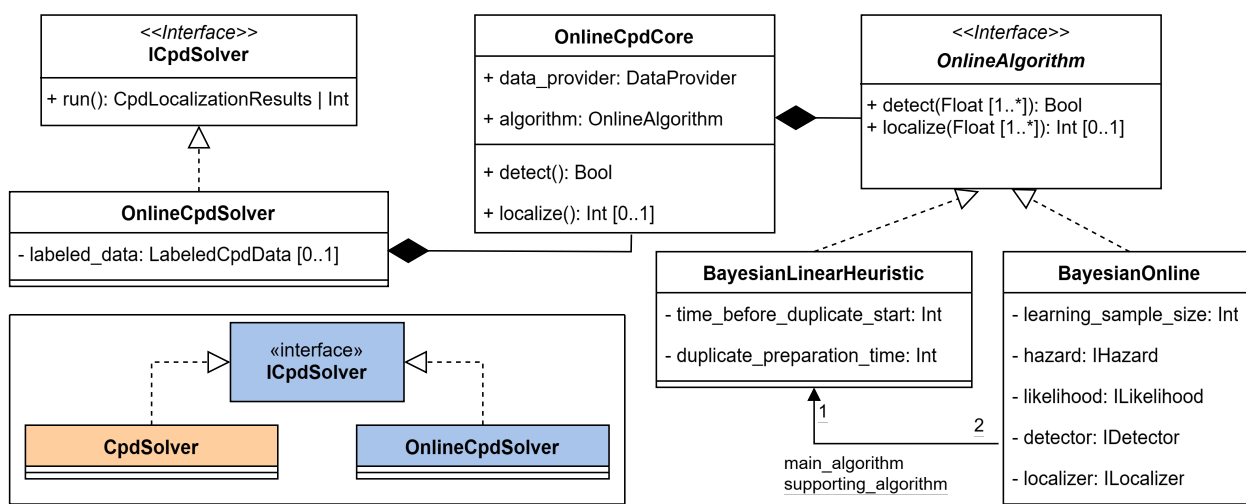


Рис. 5. Интеграция байесовского онлайн-алгоритма в PySATL-CPD. Синим выделены добавленная в её ходе функциональность, оранжевым — существовавшая на момент реализации базового алгоритма.

Помимо этого была реализована¹⁸ эвристика переключения между алгоритмами-объектами по прошествии некоторого количества шагов без разладки в целях ограничения роста длины пробега. В коде она тоже является классом, реализующим интерфейс `OnlineAlgorithm`. Внутри неё используются 1–2 объекта `BayesianOnline`-алгоритма. Более подробно эта эвристика описывалась ранее.

¹⁸<https://github.com/PySATL/pysatl-cpd/pull/52> — nick: alexdtat, дата обращения 24.05.2025

5.2. Предсказательные модели

Предсказательные модели реализуют интерфейс `ILikelihood`. Изначально была реализована гауссовская предсказательная модель¹⁹, после чего, по результатам исследований, было решено добавить экспоненциальную²⁰. Благодаря наличию свойства сопряжённости у выбранных функций правдоподобия данные модели выполняют достаточно простые арифметические операции над параметрами вместо решения оптимизационных задач и интегрирования, однако при необходимости в этот интерфейс вписываются и более общие решения.

Поскольку адаптивная предсказательная модель²¹ использует описанные выше предсказательные модели и требует оценки правдоподобия по обучающей выборке для выбора более подходящей из них, был создан дополнительный интерфейс `ILikelihoodWithPriorProbability` с соответствующим методом. Уже существующим моделям было необходимо лишь добавить эту функциональность, для чего были созданы их наследники, также реализующие этот новый интерфейс.

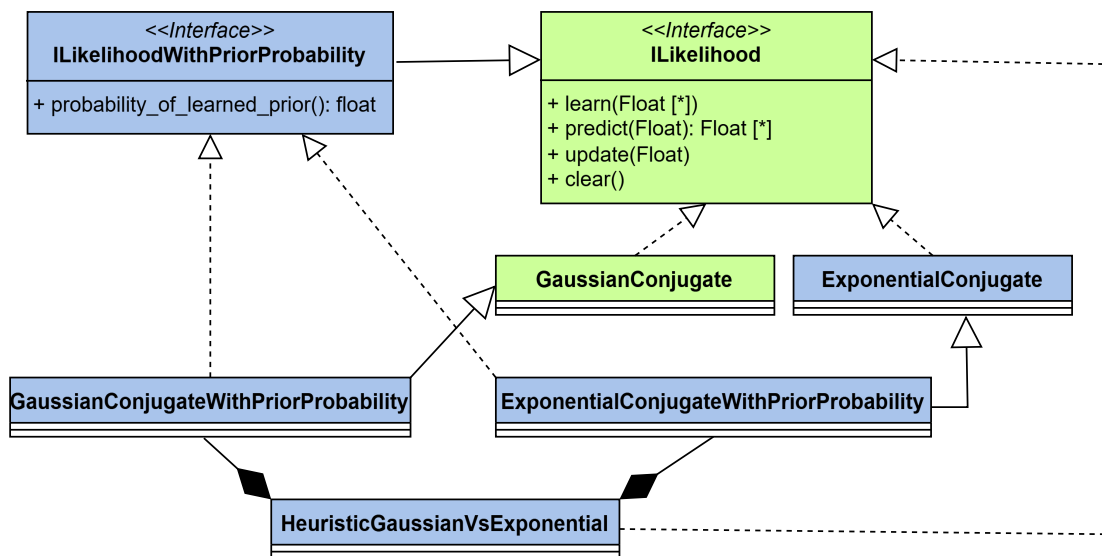


Рис. 6. Предсказательные модели. Зелёным выделена функциональность, входившая в базовый алгоритм, синим — добавленная при реализации онлайн-алгоритма.

¹⁹<https://github.com/PySATL/pysatl-cpd/pull/27> — nick: alexdtat, дата обращения 24.05.2025

²⁰<https://github.com/PySATL/pysatl-cpd/pull/46> — nick: alexdtat, дата обращения 24.05.2025

²¹<https://github.com/PySATL/pysatl-cpd/pull/48> — nick: alexdtat, дата обращения 24.05.2025

6. Настройка гиперпараметров и численное исследование онлайн-алгоритма

Для завершения алгоритма не хватает оценки параметров, которые можно получить моделированием на синтетических данных. В данном разделе описывается численное исследование алгоритма и предсказательных моделей, в том числе позволяющее осуществить эту настройку. Сначала приводится краткое описание рассматриваемых данных. Затем описываются результаты, полученные для гауссовской предсказательной модели. Анализ мощностей и задержек указывает на необходимость рассмотрения альтернативных моделей. Далее описывается исследование эвристической предсказательной модели, показавшее в большинстве случаев не худшие результаты. На основе анализа мощностей и задержек делается вывод о возможности использования эвристики понижения сложности алгоритма, описанной ранее, на данных без разладки. Наконец, рассматриваются результаты настроенного алгоритма на реальных данных, демонстрирующие его применимость.

6.1. Параметры эксперимента и генерация синтетических данных

Исследование проводилось на синтетических данных размера 500 значений, разладка (в случае наличия) была в точке 250. При генерации данных были выбраны следующие распределения с параметрами до разладки и после (если она была) соответственно $\mu_0 = 1.0$, $\sigma_0 = 1.0$ и $\mu_1 = 10.0$; $\sigma_1 = 5.0$:

- нормальное распределение $\mathcal{N}$ с параметрами;
- равномерное распределение U с параметрами $a_0 = 0.0$, $b_0 = 1.0$ и $a_1 = 1.0$, $b_1 = 4.0$;
- бета-распределение $\mathcal{B}$ с параметрами $\alpha_0 = 0.5$, $\beta_0 = 0.5$ и $\alpha_1 = 5.0$, $\beta_1 = 5.0$;

- экспоненциальное распределение Exp с параметрами $\lambda_0 = 1.0$ и $\lambda_1 = 5.0$;
- распределение Вейбулла W с параметрами $k_0 = 1.0$, $\lambda_0 = 0.5$ и $k_1 = 1.0$, $\lambda_1 = 5.0$.

Для каждой из конфигураций было сгенерировано по 1000 наборов данных.

Численное исследование проводилось на устройстве Lenovo ThinkPad T15 с процессором Intel(R) Core(TM) i5-10310U CPU @ 1.70GHz 2.21 GHz (процессор x64), оперативной памятью 32 Гб и операционной системой Windows 10 Pro версии 22H2 (64-разрядная операционная система).

6.2. Исследование гауссовской предсказательной модели

На синтетических данных в прошлых семестрах было проведено исследование гауссовской предсказательной модели [18]. Сначала на дополнительно сгенерированных данных с нормальным распределением были определены пороги `ThresholdDetector`, соответствующие уровням значимости 10%, 5%, 1% и 0.5%. Затем с этими значениями порогов проводилось исследование на данных, описанных в подразделе 6.1. Были введены понятия штатного (модель совпадает с данными до или после разладки) и нештатного (остальные случаи) режимов работы. Размер обучающей выборки был выбран 50.

На этом этапе проводился расчёт распределения длины пробега для каждого наблюдения из данных с использованием заглушки в качестве детектора-заглушки, чтобы не происходило распознавания разладок. Это позволило один раз рассчитать распределение длины пробега на каждом шаге и применять разные значения порога, оценивая положения первой разладки. При таком подходе, однако, нельзя оценивать поведение алгоритма после первой разладки, поэтому ложное и преждевременное определение разладки (т.е. в случае большого уровня значимо-

сти) будет уменьшать количество корректно определённых разладок (т.е. уменьшать мощность).

При исследовании штатного режима был поставлен **RQ1**: какова мощность алгоритма при работе в штатном?

$\alpha_{\mathcal{N}}$	0.1	0.05	0.01	0.005
$\beta \backslash \text{Порог}$	0.27	0.18	0.07	0.04
$\beta_{\mathcal{N}-\mathcal{B}}$	0.794	0.824	0.864	0.878
$\beta_{\mathcal{N}-\text{Exp}}$	0.779	0.805	0.84	0.863
$\beta_{\mathcal{N}-U}$	0.87	0.893	0.917	0.929
$\beta_{\mathcal{N}-W}$	0.783	0.811	0.853	0.863
$\beta_{\mathcal{N}-\mathcal{N}}$	0.968	0.975	0.981	0.981
$\beta_{\mathcal{B}-\mathcal{N}}$	0.997	0.998	1.0	1.0
$\beta_{\text{Exp}-\mathcal{N}}$	0.512	0.574	0.681	0.724
$\beta_{U-\mathcal{N}}$	0.998	1.0	1.0	1.0
$\beta_{W-\mathcal{N}}$	0.47	0.523	0.623	0.665

Таблица 1. Локализация с гауссовской предсказательной моделью в штатном режиме

При исследовании нештатного режима был поставлен **RQ2**: возможно ли применение алгоритма во внештатном режиме?

$\beta \backslash \text{Порог}$	0.27	0.18	0.07	0.04
$\beta_{\mathcal{B}-\mathcal{B}}$	0.945	0.96	0.976	0.983
$\beta_{\mathcal{B}-\text{Exp}}$	0.818	0.839	0.869	0.881
$\beta_{\mathcal{B}-U}$	0.993	0.996	0.998	0.998
$\beta_{\mathcal{B}-W}$	0.813	0.832	0.862	0.876
$\beta_{\text{Exp}-\mathcal{B}}$	0.309	0.353	0.439	0.477
$\beta_{\text{Exp}-\text{Exp}}$	0.337	0.393	0.471	0.508
$\beta_{\text{Exp}-U}$	0.406	0.465	0.543	0.594
$\beta_{\text{Exp}-W}$	0.374	0.425	0.518	0.544
$\beta_{U-\mathcal{B}}$	0.892	0.917	0.946	0.955
$\beta_{U-\text{Exp}}$	0.829	0.851	0.875	0.888
β_{U-U}	0.987	0.991	0.996	0.997
β_{U-W}	0.851	0.877	0.904	0.914
$\beta_{W-\mathcal{B}}$	0.324	0.37	0.456	0.498
$\beta_{W-\text{Exp}}$	0.373	0.421	0.516	0.561
β_{W-U}	0.374	0.423	0.504	0.538
β_{W-W}	0.351	0.416	0.508	0.553

Таблица 2. Локализация с гауссовской предсказательной моделью в нештатном режиме

Во-первых, оказалось, что мощность алгоритма при решении задачи определения наличия разладки не менее 0.97, что является очень хорошим результатом. Во-вторых, оказалось, что мощность алгоритма при решении задачи корректной локализации существенно зависит от класса распределений. Алгоритм продемонстрировал хорошие результаты в штатном режиме 1 и при работе с равномерным распределением и бета-распределением 2, но заметно хуже справился в случае экспоненциального распределения и распределения Вейбулла 2. Это можно связать с асимметричностью и существенно отличающимся носителем в сравнении с нормальным распределением, из-за чего многие значения определялись предсказательной моделью как слишком большие выбросы.

Таким образом, можно констатировать в ответ на **RQ1** достаточно высокий уровень мощности при работе в штатном режиме. Отвечая на **RQ2**, стоит отметить, что разница между различными конфигурациями оказалась существеннее. В целом, алгоритм всё ещё может применяться даже во внештатном режиме, но с учётом заметного снижения качества результатов. Причём, в обоих режимах экспоненциальное распределение и распределение Вейбулла составляли сложность для алгоритма. В связи с этим выводом было решено добавить более подходящую модель и эвристику, позволяющую во время обучения выбирать одну из реализованных моделей.

6.3. Исследование адаптивной предсказательной модели

После реализации адаптивной предсказательной модели было проведено её исследование. Стоит отметить важное отличие от исследования базового алгоритма с гауссовской предсказательной моделью: теперь для сохранения инкапсуляции и рассмотрения поведения алгоритма после обнаружения первой разладки (что существенно для онлайн-алгоритма) исследование проводилось не с детектором-заглушкой, а с полноценным пороговым детектором. Поэтому было решено зафиксиро-

ровать набор порогов, полученных для базового алгоритма в случае нормального распределения, и оценить качество работы алгоритма с ними.

Исследование проводилось на тех же самых данных, но с учётом изменения методологии потребовало другой обработки результатов:

- **TP (true positive)** — разладка была обнаружена в пределах 25 шагов от размеченной.
- **FP (false positive)** — была обнаружено более чем одна разладка в пределах 25 шагов от размеченной или была обнаружена разладка вне этого отрезка.
- **FN (false negative)** — не было обнаружено ни одной разладки в пределах 25 шагов от размеченной.

Аналогично исследованию базового алгоритма, **TN (true negative)** результаты не учитываются, поскольку их определение проблематично, а информативность этой величины мала. Связано это с тем, что класс TN оказывается на порядки больше всех остальных и может непропорционально менять размер в ходе работы в зависимости от размера входных данных.

При исследовании был сформулирован **RQ3**: позволяет ли эвристическая адаптивная предсказательная модель добиться повышения мощности?

Исследование продемонстрировало наличие у эвристической адаптивной модели следующей особенности поведения: при локализации разладки раньше её настоящего положения возможен неудачный выбор предсказательной модели при обучении, что приведёт к ложному обнаружению ещё одной разладки, прежде чем будет выбрана корректная модель. Тем не менее, такое поведение не исключает обнаружение разладки и не вмешивается в работу алгоритма на данных без разладки, что также необходимо учитывать. Для устранения этого недостатка могут быть использованы другие эвристики, например, отбрасывание

некоторого сегмента в начале обучающей выборки, чтобы убрать остаток данных до разладки с другим распределением (после проведения предварительного анализа, каким должен быть размер этого сегмента).

Также было замечено, что алгоритм плохо себя показал в конфигурациях распределений бета–бета и Вейбулл–равномерное. Это связано с тем, что адаптивная предсказательная модель в обоих случаях всегда выбирала экспоненциальное правдоподобие на этапе обучения, тогда как для всех распределений, кроме Вейбулла, на прошлом этапе хорошо справлялось гауссовское правдоподобие. Вероятно, это связано с тем, что в целом выборка, сосредоточенная на отрезке $[0; 1]$, оказывается более правдоподобной с точки зрения экспоненциального распределения, но изменения плотности внутри этого носителя лучше улавливает гауссовское правдоподобие. При этом в целом с экспоненциальным распределением и с распределением Вейбулла адаптивная предсказательная модель справилась существенно лучше.

Таким образом, в ответ на **RQ3** можно констатировать существенное улучшение качества результатов, за исключением 2 частных случаев (бета–бета и Вейбулл–равномерное). Устранение таких недостатков требует разработки продвинутых способов подбора оптимальной модели и других улучшений, но может обходиться при наличии априорных знаний о предметной области использованием более подходящей модели, например, гауссовской.

6.4. Анализ задержек локализации

Задержки локализации позволяют нам оценивать, через сколько шагов алгоритм с большой вероятностью обнаружит появившуюся разладку. Во-первых, это может быть ценным само по себе в контексте предметной области (например, если требуется как можно более быстрая реакция на какое-то событие). Во-вторых, при наличии этой оценки можно делать выводы о возможности использования эвристики понижения сложности алгоритма на данных без разладки.

Сформулируем **RQ4**: показывает ли эвристическая адаптивная

предсказательная модель задержки локализации достаточно малые, чтобы можно было использовать эвристику уменьшения сложности? Для ответа на него рассмотрим несколько важных частных случаев (разладки между парами распределений из гауссовского семейства, экспоненциального и Вейбулла) с квантилями 0.5, 0.75 и 0.9, и оценим длину пробега в целом по различным конфигурациям.

Анализ задержек при локализации (количество шагов от локализованной разладки до момента её определения) показал зависимость от класса распределения. Также работа прошлого семестра показала ожидаемое смещение в сторону более позднего определения наличия разладки при уменьшении порога. Тем не менее, во всех случаях на гистограмме задержек наблюдалось смещение пика в сторону нулевой (в хороших случаях все задержки были в окрестности 0) и последующий постепенный спад.

Delays Distribution - normal-normal

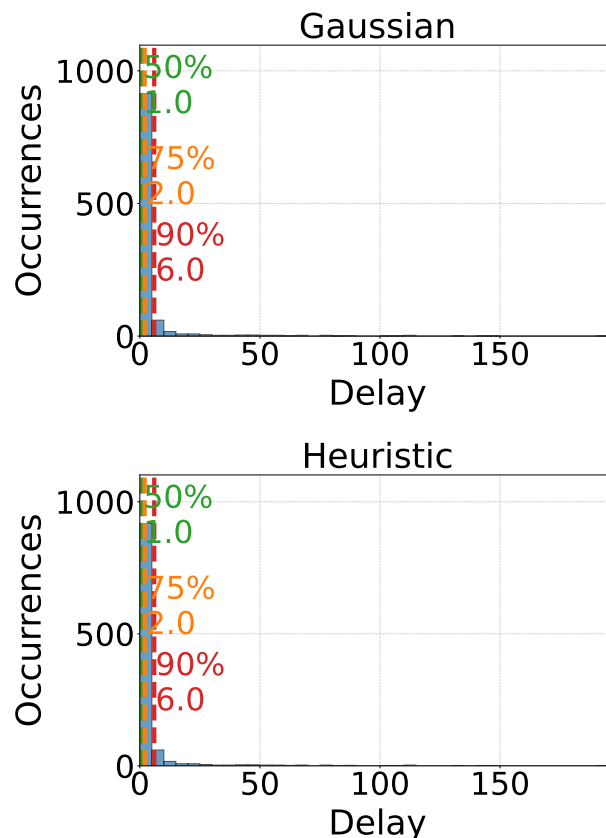


Рис. 7. Задержки локализации для конфигурации нормальное–нормальное

Delays Distribution - exponential-exponential

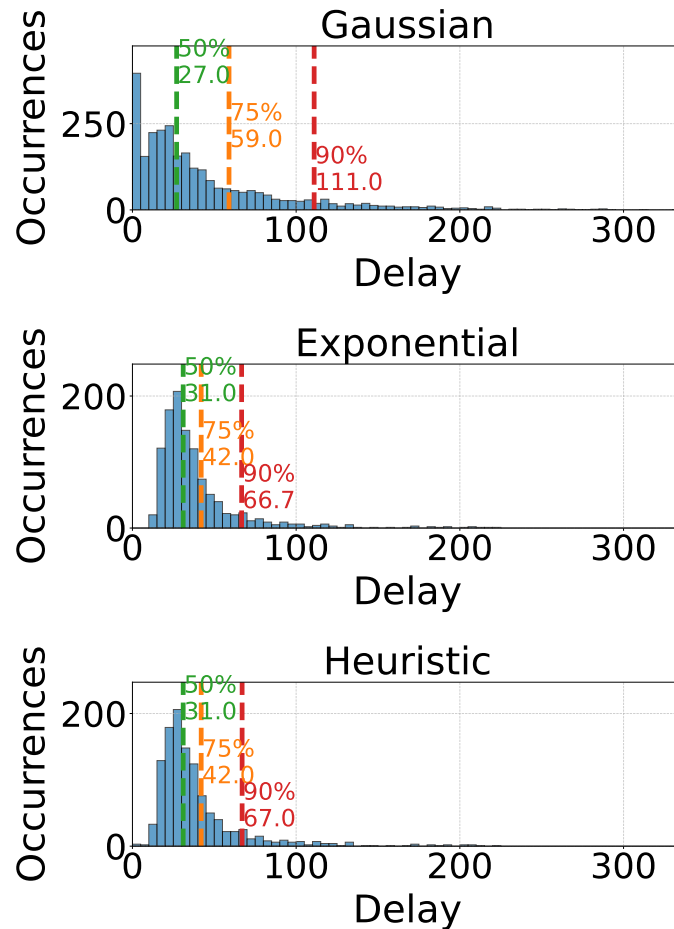


Рис. 8. Задержки локализации для конфигурации экспоненциальное–экспоненциальное

Наконец, анализ задержек показал, что, за вычетом упомянутых выше плохих случаев, максимальная по всем конфигурациям медиана задержки оказалась равна 100, а максимальное среднее — 117. При этом мощности оказались не менее 0.79. Для задержек экспоненциальной и эвристической адаптивной моделей при конфигурациях экспоненциальное–экспоненциальное (рис. 8) и Вейбулл–Вейбулл (рис. 9) уменьшились квантили 0.5, 0.75 и 0.9 по сравнению с гауссовской (также стоит учитывать, что у гауссовской многие малые задержки были связаны с ложными срабатываниями на значениях, оцениваемых ей как слишком большие выбросы). В случае конфигурации нормальное–нормальное (рис. 7) результаты эвристической адаптивной модели не отличаются от гауссовской. Стоит отметить, что экспоненци-

Delays Distribution - weibull-weibull

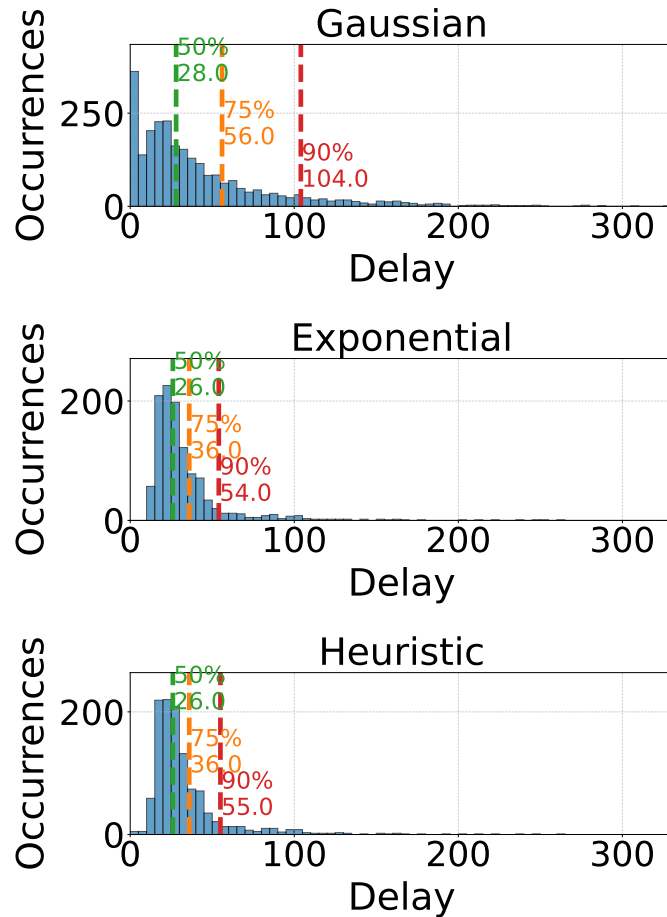


Рис. 9. Задержки локализации для конфигурации Вейбулл–Вейбулл

альная модель в последнем случае показывает себя плохо, поскольку все отрицательные значения она воспринимает как невозможные с точки зрения моделируемого распределения, из-за чего тут же появляются ложные разладки.

Всё описанное выше, отвечая на **RQ4**, указывает на возможность использования линейной эвристики в большинстве случаев с высокой мощностью и задержками, не выходящими за пределы охватываемого отдельным алгоритмом сегмента данных. Для близости к условиям исследования алгоритма стоит выбрать размер рабочего окна, близкий к 500, а размер обучающей выборки — близкий к 250.

Хотя основным приоритетом было исследование поведения алгоритма, также была проведена базовая оценка его производительности. Среднее время работы по всем экспериментам всех конфигураций на

выборке из 500 наблюдений оказалось 0.12 секунды при стандартном отклонении 0.04 секунды.

6.5. Результаты на реальных данных

Для оценки качества работы алгоритма на реальных данных было решено использовать данные из работы [1], доступные для загрузки по ссылке²². Алгоритм был сконфигурирован постоянной функцией выживаемости и адаптивной предсказательной моделью. Частота для функции выживаемости бралась обратная размеру данных, размер обучающей выборки был 20.

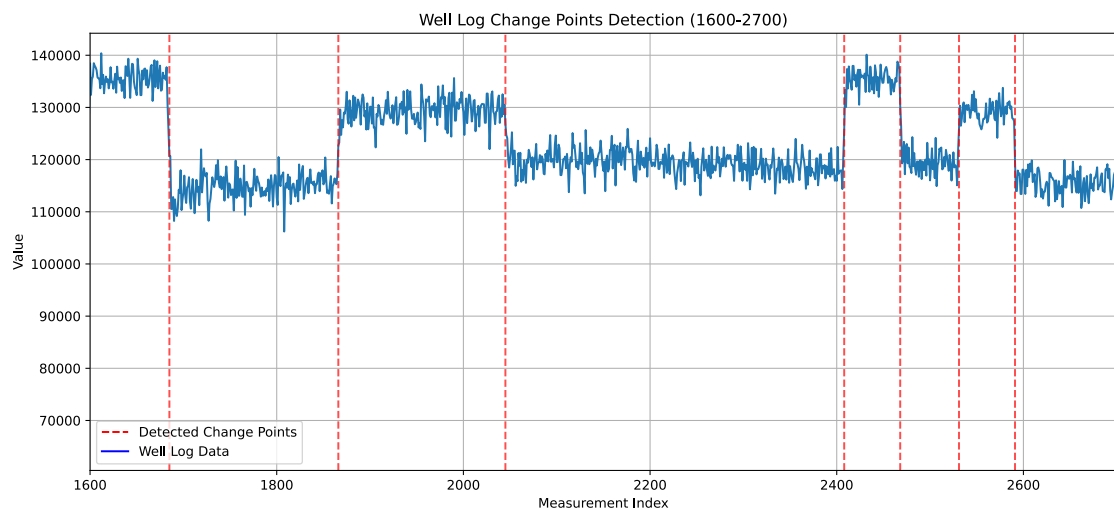


Рис. 10. Результат работы алгоритма на данных WELL-LOG

Данные WELL-LOG — это наблюдаемый ядерно-магнитный отклик во время бурения скважины. В статье они моделировались гауссовской функцией правдоподобия с изменяющимся матожиданием. С зафиксированным значением порога 0.04 на рассматриваемом промежутке 1600–2700 были получены (рис. 10) следующие разладки: 1685, 1866, 2045, 2408, 2468, 2531, 2591. К сожалению, для этих данных не приводится разметка, однако количество и примерное местоположение разладок соответствуют отмеченным в статье на графике.

Данные 1972–1975 Dow Jones Returns — это дневная доходность промышленного индекса Доу Джонса. В статье они моделировались

²²<http://oa.ee.tsinghua.edu.cn/~ouzhijian/software/DppCode.rar> — дата обращения 09.05.2025

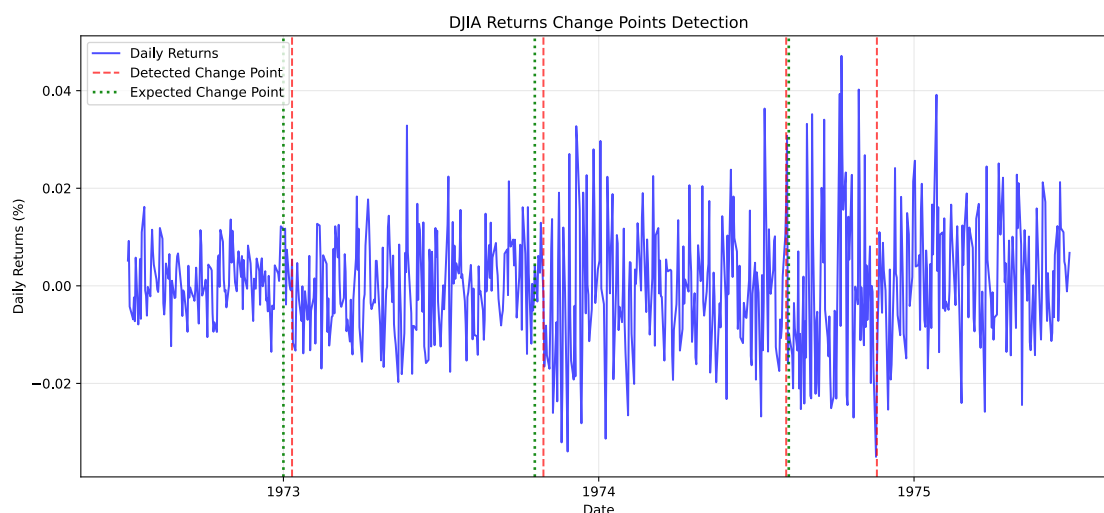


Рис. 11. Результат работы алгоритма на данных 1972–1975 Dow Jones RETURNS

гауссовской функцией правдоподобия с нулевым матожиданием. Из рассматриваемых трёх примеров в них оказалось сложнее всего получить точные результаты. Возможно, это связано с тем, что рыночные данные зачастую имеют сложную структуру и не всегда независимы и одинаково распределены. В статье отмечены даты событий, которые могли оказать существенное влияние на индекс Доу Джонс: 30.01.1973, 19.10.1973 и 09.08.1974. При значении порога 0.04 были найдены разладки, соответствующие 01.11.1973 и 29.10.1973. Первый результат может быть объяснён тем, что и в статье вероятная длина пробега указывает на более раннюю разладку. Второй результат отклоняется лишь на 6 наблюдений от указанного события, что является хорошим результатом. А вот третья разладка не была обнаружена. Порог 0.07 выдаёт первую разладку ближе к искомой дате, но всё ещё не фиксирует третью: 11.01.1973 и 29.10.1973. Порог 0.18 уже фиксирует разладку в окрестности третьей, но также приводит к обнаружению ещё одной, не отмеченной в исходных данных: 11.01.1973, 29.10.1973, 20.08.1974 и 19.11.1974. Наконец, порог 0.27 (рис. 11) приводит к обнаружению третьей разладки ближе к искомому событию, но всё ещё приводит к обнаружению четвёртой разладки: 11.01.1973, 29.10.1973, 06.08.1974 и 19.11.1974. На возможное обнаружение четвёртой разладки алгоритмом также указывает размывание вероятностной массы для длины пробега

ближе к концу графика в статье.

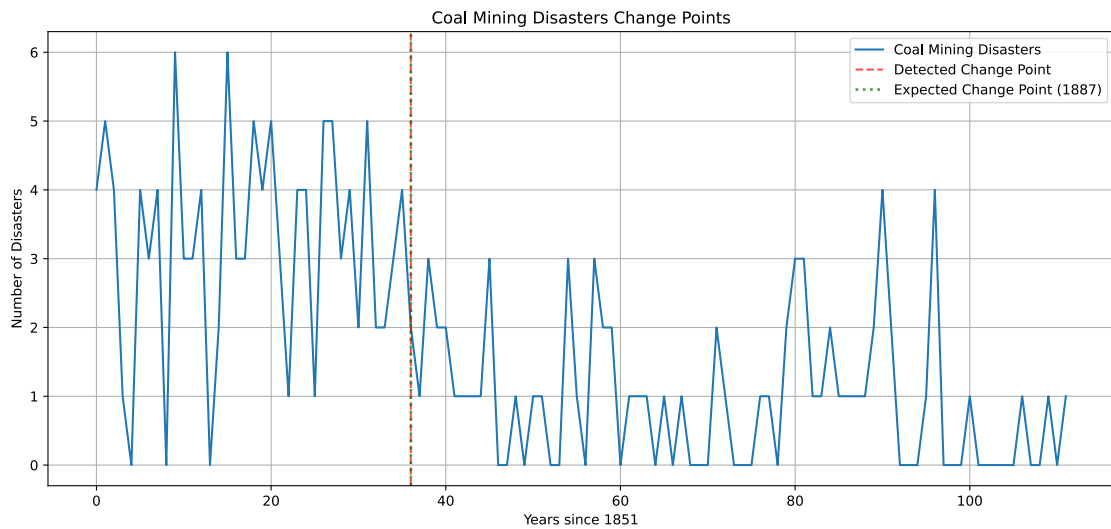


Рис. 12. Результат работы алгоритма на данных COAL MINE DISASTER

Данные COAL MINE DISASTER — это количество катастроф на угольных шахтах (в статье указано еженедельное, однако удалось найти лишь разбивку по годам), в которых погибло десять и более человек в период с 1851 по 1962 год. В отличие от прошлых примеров, в этом случае в статье моделировался пуассоновский процесс, который не входит в рассматриваемую здесь эвристическую адаптивную модель. Тем не менее, при пороге 0.04 была точно локализована разладка в 1887 году (рис. 12), что соответствует отмеченному событию, повлиявшему на частоту катастроф. Важно заметить, что на вход алгоритму подавалось количество катастроф за год, а не их суммарное накопленное количество за предшествующее время.

Заключение

Была проведена адаптация байесовского алгоритма под онлайн-задачу: выделены детектор и локализатор, реализована очистка данных до последней разладки, предложена эвристика для борьбы с линейным ростом затрачиваемых времени и памяти каждым шагом алгоритма.

Был реализован и интегрирован в PySATL-CPD байесовский онлайн-алгоритм. Для этого были выделены в качестве интерфейсов отдельные компоненты алгоритма и создана функциональность для использования онлайн-алгоритмов (аналогичная существовавшей для остальных алгоритмов).

Помимо реализованной ранее гауссовской предсказательной модели были добавлены экспоненциальная и эвристическая адаптивная эвристическая. Последняя, в свою очередь, потребовала добавления возможности сравнения гауссовской и экспоненциальной моделей на обучающей выборке для последующего выбора более подходящей.

Была проведена настройка параметров алгоритма для различных моделей. При исследовании на синтетических данных были вычислены пороги и размеры обучающих выборок, а также были оценены мощность и задержки, на основе чего было принято решение о допустимости использования эвристики снижения сложности алгоритма на больших входных данных ценой потери информации о старых наблюдениях. Также были воспроизведены результаты из исходной работы [1].

Исходный код доступен в репозитории (nick: alexdtat):

<https://github.com/PySATL/pysatl-cpd>

Результаты данной работы также были ранее представлены²³ на конференции «МАТ-МЕХ. НАУКА 2025» на секции «Вычислительная стохастика и статистические модели». Также они подаются на конференцию “DAMDID 2025”²⁴.

²³<https://se.math.spbu.ru/math-sci/2025/sessions/> — дата обращения 23.05.2025

²⁴<https://damdid2025.frccsc.ru/> — дата обращения 24.05.2025

Список литературы

- [1] Adams Ryan Prescott, MacKay David J. C. Bayesian Online Change-point Detection. — 2007. — 0710.3742.
- [2] Altamirano Matias, Briol François-Xavier, Knoblauch Jeremias. Robust and Scalable Bayesian Online Changepoint Detection. — 2023. — 2302.04759.
- [3] Aminikhanghahi Samaneh, Cook Diane Joyce. A survey of methods for time series change point detection // Knowledge and Information Systems. — 2016. — Vol. 51. — P. 339 – 367. — URL: <https://api.semanticscholar.org/CorpusID:15595198>.
- [4] Cakmak Ayse, Reinertsen Erik, Nemati Shamim, Clifford Gari D. Benchmarking changepoint detection algorithms on cardiac time series. — 2024. — [2404.12408](#).
- [5] [Change-point detection in time-series data based on subspace identification](#) // Proceedings of the 7th IEEE International Conference on Data Mining, ICDM 2007. — Proceedings - IEEE International Conference on Data Mining, ICDM. — 2007. — P. 559–564. — 7th IEEE International Conference on Data Mining, ICDM 2007 ; Conference date: 28-10-2007 Through 31-10-2007.
- [6] Change-point detection in time-series data by relative density-ratio estimation / Song Liu, Makoto Yamada, Nigel Collier, Masashi Sugiyama // [Neural Networks](#). — 2013. — Vol. 43. — P. 72–83. — URL: <https://www.sciencedirect.com/science/article/pii/S0893608013000270>.
- [7] Fleming Matt. Taming Noisy Benchmark Results Using Change Point Detection. — Dublin : USENIX Association, 2024. — oct.
- [8] Malladi Rakesh, Kalamangalam Giridhar P, Aazhang Behnaam. On-line Bayesian change point detection algorithms for segmentation of

- epileptic activity // 2013 Asilomar Conference on Signals, Systems and Computers. — 2013. — P. 1833–1837.
- [9] Murphy Kevin P. Conjugate Bayesian analysis of the Gaussian distribution. — 2007. — URL: <https://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf>.
 - [10] Rosenbaum Paul R. An exact distribution-free test comparing two multivariate distributions based on adjacency // Journal of the Royal Statistical Society Series B: Statistical Methodology. — 2005. — Vol. 67, no. 4. — P. 515–530.
 - [11] Saatçi Yunus, Turner Ryan D, Rasmussen Carl E. Gaussian process change point models // Proceedings of the 27th International Conference on Machine Learning (ICML-10). — 2010. — P. 927–934.
 - [12] Soland Richard. Bayesian Analysis of the Weibull Process With Unknown Scale and Shape Parameters // [Reliability, IEEE Transactions on](#). — 1969. — 12. — Vol. R-18. — P. 181 – 184.
 - [13] Von Toussaint Udo. Bayesian inference in physics // Reviews of Modern Physics. — 2011. — Vol. 83, no. 3. — P. 943.
 - [14] Wei Li, Keogh Eamonn. [Semi-supervised time series classification](#) // Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. — KDD '06. — New York, NY, USA : Association for Computing Machinery, 2006. — P. 748–753. — URL: <https://doi.org/10.1145/1150402.1150498>.
 - [15] Zhao Zhan, Koutsopoulos Haris N, Zhao Jinhua. Detecting pattern changes in individual travel behavior: A Bayesian approach // Transportation research part B: methodological. — 2018. — Vol. 112. — P. 73–88.
 - [16] [A regularized kernel-based approach to unsupervised audio segmentation](#) / Zaïd Harchaoui, Félicien Vallet, Alexandre Lung-Yut-Fong, Olivier Cappé. — 2009. — 04. — P. 1665–1668.

- [17] van den Burg Gerrit J. J., Williams Christopher K. I. An Evaluation of Change Point Detection Algorithms. — 2022. — [2003.06222](#).
- [18] Татьянаенко Алексей. Адаптация и исследование байесовского подхода к обнаружению разладки для PySATL-CPD-Module. — 2024. — URL: https://se.math.spbu.ru/thesis/texts/Tat'janenko_Aleksej_Dmitrievich_Spring_practice_3rd_year_2024_text.pdf.
- [19] Татьянаенко Алексей. Байесовское обнаружение разладки в реальном времени. — 2024. — URL: https://se.math.spbu.ru/thesis/texts/Tat'janenko_Aleksej_Dmitrievich_Autumn_practice_3rd_year_2023_text.pdf.