МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра теории вероятностей и математической статистики

МЯСОЕДЕНКОВ
Андрей Викторович

Последовательный статистический анализ данных, образующих марковское случайное поле

Магистерская диссертация

Научный руководитель:
доктор физ.-мат. наук,
профессор А.Ю. Харин

Допущена к защите
« ____ » _____ 2025 г.
Зав. кафедрой теории вероятностей
и математической статистики,
доктор физ.-мат. наук,
профессор А.Ю. Харин

Минск, 2025

РЕФЕРАТ

Магистерская диссертация, 50 страниц, 9 рисунков, 13 таблиц, 16 источников.

Ключевые слова: ПРОВЕРКА ГИПОТЕЗ, КРИТЕРИЙ ВАЛЬДА, ПОСЛЕДОВАТЕЛЬНЫЙ КРИТЕРИЙ, МАРКОВСКОЕ СВОЙСТВО, ОТНОШЕНИЕ ПРАВДОПОДОБИЯ, МАРКОВСКОЕ СЛУЧАЙНОЕ ПОЛЕ, ОТКЛОНЕНИЕ ТЬЮКИ–ХЬЮБЕРА, ГАУССОВО МАРКОВСКОЕ СЛУЧАЙНОЕ ПОЛЕ, МЕТОД МОНТЕ–КАРЛО, УСТОЙЧИВЫЙ ПОСЛЕДОВАТЕЛЬНЫЙ КРИТЕРИЙ, СГЛАЖИВАНИЕ ДАННЫХ, ФИЛЬТРАЦИЯ 1€, ФИЛЬТР КАЛМАНА.

Объекты исследования: последовательный анализ наблюдений, поступающих из марковского случайного поля.

Цель исследования: получение оценок среднего количества наблюдений для принятия решения о правильности гипотезы, а также оценок средних ошибок I и II рода.

Методы исследования: системный подход, вероятностное моделирование методом Монте-Карло, изучение соответствующей литературы и электронных источников, постановка задачи и её решение.

Области применения: эпидемиологическое моделирование, анализ динамики популяции, распознавание речи, генерация текстов, прогнозирование трендов на рынке.

Результаты: разработано программное обеспечения для анализа численных характеристик последовательного критерия.

РЭФЕРАТ

Магістарская дысертацыя, 50 старонак, 9 малюнкаў, 13 табліц, 16 крыніц.

Ключавыя словы: ПРАВЕРКА ГІПОТЭЗ, КРЫТЭРЫЙ ВАЛЬДА, ПАСЛЯДОЎНЫ КРЫТЭРЫЙ, МАРКАЎСКАЯ ЎЛАСЦІВАСЦЬ, АДНОСІНЫ ПРАЎДАПАДОБНАСЦІ, МАРКАЎСКАЕ ВЫПАДКОВАЕ ПОЛЕ, АДХІЛЕННЕ ЦЮКІ–Х'ЮБЕРА, ГАУСАВА МАРКАЎСКАЕ ВЫПАДКОВАЕ ПОЛЕ, МЕТАД МОНТЭ–КАРЛА, УСТОЙЛІВЫ ПАСЛЯДОЎНЫ КРЫТЭРЫЙ, ЗГЛАДЖВАННЕ ДАДЕЗНЫХ, ФІЛЬТРАЦЫЯ 1€, ФІЛЬТР КАЛМАНА.

Аб'екты даследавання: паслядоўны аналіз назіранняў, якія паступаюць з маркаўскага выпадковага поля.

Мэта даследавання: атрыманне ацэнак сярэдняй колькасці назіранняў для прыняцця рашэння аб правільнасці гіпотэзы, а таксама ацэнак сярэдніх памылак I і II роду.

Метады даследавання: сістэмны падыход, імавернаснае мадэляванне метадам Монтэ-Карла, вывучэнне адпаведнай літаратуры і электронных крыніц, пастаноўка задачы і яе вырашэнне.

Вобласці прымянення: эпідэміялагічнае мадэляванне, аналіз дынамікі папуляцыі, распазнанне мовы, генерацыя тэкстаў, прагназаванне трэндаў на рынку.

Вынікі: распрацавана праграмнае забеспячэнні для ацэнкі характарыстык паслядоўнага крытэрыя.

ABSTRACT

Master's thesis, 50 pages, 9 figures, 13 tables, 16 sources.

Keywords: HYPOTHESIS TESTING, WALD TEST, SEQUENTIAL TEST, MARKOV PROPERTY, LIKELIHOOD RATIO, MARKOV RANDOM FIELD, TUKEY–HUBER ERROR, GAUSSIAN MARKOV RANDOM FIELD, MONTE CARLO METHOD, ROBUST SEQUENTIAL TEST, DATA SMOOTHING, 1€ FILTER, KALMAN FILTER.

Objects of study: sequential analysis of observations coming from the Markov random field.

Purpose of the study: getting estimates of the average observations num-ber for making a decision on the correct hypothesis, as well as estimates of the average type I and type II errors.

Research methods: systematic approach, probabilistic Monte Carlo modeling, study of relevant literature and electronic sources, problem statement and its solution.

Areas of use: epidemiological models, population dynamics, speech recognition, text generator, market trends forecasting.

Results: software has been developed for performance estimation of the sequential probability ratio test.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Последовательный анализ является техникой проведения статистического исследования теории принятия решений. Характерной чертой метода является не определенное заранее количество наблюдений, которые будут необходимы в процессе испытания. Решение об окончании проводимого эксперимента на каждой стадии напрямую зависит от всех результатов предыдущих наблюдений.

Достоинством данного метода в применении к проверке статистических гипотез является то, что он позволяет создать такую методику статистической проверки, которая требует в среднем существенно меньшего числа наблюдений, чем равная ей по надежности последовательная или непоследовательная статистическая проверка, в которой заранее известно, сколько количество наблюдений будет необходимо.

Частным случаем последовательного анализа является последовательный критерий отношения вероятностей, который впервые был предложен *Абрахамом Вальдом* в 1943 г. Вальд переформулировал некоторые существующие на тот момент проблемы как проблемы последовательного анализа и ввёл статистику отношения правдоподобий для решения данной задачи. Данная техника была представлена главным образом в качестве инструмента проверки для определения того, соответствует ли некоторая партия товара своим производственным требованиям.

Сравнение представленного последовательного критерия с любым другим последовательным или непоследовательным критерием, равного по мощности, показало, что он дает наибольший возможный выигрыш в среднем числе наблюдений. Было доказано, что при определенных условиях выигрыш в количестве наблюдений, которые необходимы при проверке простой гипотезы относительно единственной конкурирующей гипотезы, может быть двукратным.

На выводах, сделанных Вальдом, в дальнейшем было построено большое количество последовательных критериев, которые обобщают методы или являются их частными случаями. Одним из таких типов последовательного критерия отношения вероятностей является *M–нарный последовательный критерий*, обобщающий статистическое исследование на случай трёх и более гипотез, что зачастую встречается на практике.

Оптимальные свойства последовательного метода проверки гипотез хорошо изучены для случая независимых одинаково распределенных случайных величин, однако нередко в прикладных задачах данные поступают из более сложных источников. В таких ситуациях поступающие данные не являются независимыми и одинаково распределенными. Примером таких наблюдений, является поступающая информация о последовательности наблюдений, составляющих некоторое марковское случайное поле, а также, в частности, гауссово марковское случайное поле.

Данный метод рассматривается как хорошая альтернатива или как доминирующий метод в ситуациях, когда отсутствует возможность сбора большого объёма данных для проведения анализа, проверки гипотез и статистического тестирования.

Глава 1

Основные теоретические сведения

1.1 Последовательный критерий

Пусть M_m – пространство выборок $(x_1, x_2, ..., x_m)$ объёма $m > 0$, полученных из генеральной совокупности, и пространство параметров разбито на две непересекающиеся области $\Theta = \theta_0 \cup \theta_1$. Пусть определены две простые гипотезы о параметре распределения элементов выборки

$$\begin{aligned} \mathcal{H}_0 : &\quad \theta = \theta_0; \\ \mathcal{H}_1 : &\quad \theta = \theta_1. \end{aligned} \tag{1.1}$$

Для каждого объема выборки m пространство M_m разбивается на три попарно не пересекающиеся области R_m^0, R_m^1, R_m и вводится нерандомизированное решающее правило вида

$$d = d(X) = \begin{cases} 0, & X \in R_m^0; \\ 1, & X \in R_m^1; \\ 2, & X \in R_m. \end{cases} \tag{1.2}$$

Следуя данному решающему правилу, после того как в результате наблюдений будет определена величина x_1, мы принимаем гипотезу $\mathcal{H}_0$, если x_1 попадает в область R_1^0. Если x_1 попадает в область R_1^1, то гипотеза $\mathcal{H}_0$ отвергается. Иначе, если x_1 попадает в R_1, проводится дальнейшее наблюдение (x_1, x_2). По такому же принципу определяется к какой из трёх областей R_2^0, R_2^1, R_2 относится выборка (x_1, x_2).

Аналогично поступают для каждой последующей выборки объема $m > 0$, т.е. $(x_1, x_2, ..., x_m)$, пока не будет принято решение о принятии одной из гипотез $\mathcal{H}_0$ или $\mathcal{H}_1$, либо пока m не достигнет заранее определённого порога, ограничивающего дальнейшее проведение эксперимента.

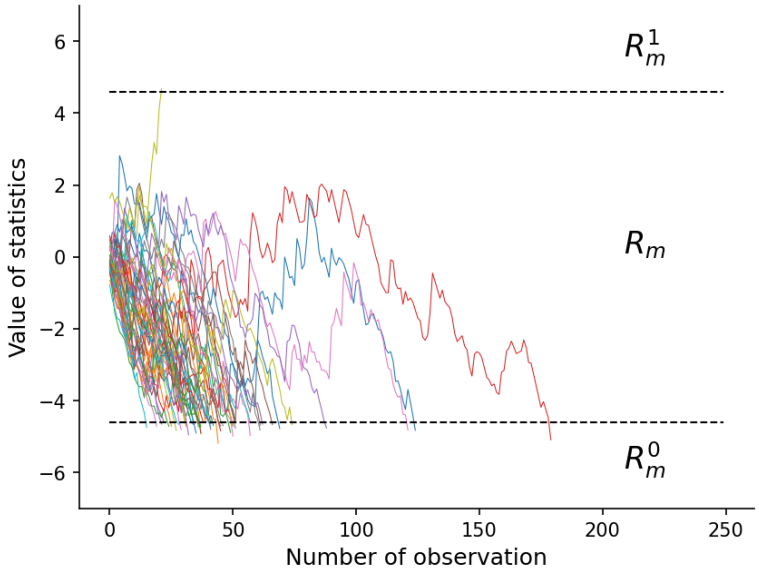


Рис. 1.1: Пример разделения пространства и
изменения статистик, помогающих принимать решения.

Таким образом, для того чтобы построить последовательный критерий, необходимо выбрать способ разбиения пространства M_m на три попарно не пересекающиеся области, а также их связь с величинами ошибок первого и второго рода.

1.2 Последовательный критерий отношения вероятностей

Пусть $f(x, \theta)$ означает функцию плотности вероятности, если случайная величина x имеет абсолютно непрерывное распределение, или функцию вероятности, если случайная величина x распределена дискретно. В данном случае считаем, что генеральная совокупность бесконечна.

Обозначим последовательные наблюдения x через $x_1, x_2, ..., x_m, ...$. Для любого $m > 0$ вероятность получения выборки $(x_1, x_2, ..., x_m)$ определяется в зависимости от рассматриваемой гипотезы.

Для гипотез $\mathcal{H}_0$ и $\mathcal{H}_1$ вероятность имеет вид соответственно

$$\begin{aligned} p_{0m} &= f(x_1, \theta_0) f(x_2, \theta_0) ... f(x_m, \theta_0) \\ p_{1m} &= f(x_1, \theta_1) f(x_2, \theta_1) ... f(x_m, \theta_1) \end{aligned} \tag{1.3}$$

Последовательный критерий отношения вероятностей для проверки нулевой гипотезы $\mathcal{H}_0$ относительно альтернативной $\mathcal{H}_1$ определяется следующим образом [1]:

- выбираются два положительных числа A и B, такие что $A > B$;
- на m стадии эксперимента вычисляется отношение $\frac{p_{1m}}{p_{0m}}$;
- результат сравнивается с A и B.

Эксперимент продолжается и производится дополнительное наблюдение, если

$$B < \frac{p_{1m}}{p_{0m}} < A, \tag{1.4}$$

Процесс оканчивается отклонением гипотезы $\mathcal{H}_0$, если

$$\frac{p_{1m}}{p_{0m}} \geq A, \tag{1.5}$$

Процесс оканчивается принятием гипотезы $\mathcal{H}_0$, если

$$\frac{p_{1m}}{p_{0m}} \leq B, \tag{1.6}$$

Для лучшего процесса вычисления данного отношения рассматривается статистика логарифма отношения правдоподобия $\log \frac{p_{1m}}{p_{0m}}$. Данное отношение может быть записано как сумма m членов

$$\log \frac{p_{1m}}{p_{0m}} = \log \frac{f(x_1, \theta_1)}{f(x_1, \theta_0)} + ... + \log \frac{f(x_m, \theta_1)}{f(x_m, \theta_0)}. \tag{1.7}$$

Следовательно, при каждом новом поступлении данных в проводимый эксперимент достаточно лишь добавить соответствующее новое слагаемое.

Тогда (1.4) примет вид

$$\log B < \sum_{k=1}^{m} \log \frac{f(x_k, \theta_1)}{f(x_k, \theta_0)} < \log A. \tag{1.8}$$

Для того, чтобы критерий имел наперед заданную силу (α, β), необходимо иметь некое представление об постоянных A и B. Оказывается, для данного последовательного критерия существуют точные верхние и нижние грани для этих постоянных

$$sup(A) = \frac{1-\beta}{\alpha}, \quad inf(B) = \frac{\beta}{1-\alpha}. \tag{1.9}$$

Вальд успешно доказал [2], что если желателен такой последовательный критерий, у которого вероятность ошибки первого рода не превосходит α, а вероятность ошибки второго рода не превосходит β, то достаточно полагать величины A и B равными их точным граням, а именно

$$A = \frac{1-\beta}{\alpha}, \quad B = \frac{\beta}{1-\alpha}. \tag{1.10}$$

1.3 M–нарный последовательный критерий отношения вероятностей

Пусть определены M простых гипотез

$$\mathcal{H}_i : \quad \theta = \theta_i, \quad i \in \mathcal{I}, \tag{1.11}$$

где $\mathcal{I} = \{1, 2, ..., M\}, \theta_i \neq \theta_j$, если $i \neq j$.

Пусть известны априорные вероятности гипотез, которые задаются соответственно $\pi_i = P(\theta = \theta_i), i \in \mathcal{I}$.

Введём апостериорные вероятности гипотез на основе полученных $n \geq 1$ наблюдений $p_n = (p_n^1, p_n^2, ..., p_n^M)$, где $p_n^j = P(\theta = \theta_j | x_1, x_2, ..., x_n)$.

Момент окончания эксперимента и принятая гипотеза определяется следующим образом

$$N_a = inf\{n \geq 1, \exists k : p_n^k > \frac{1}{1 + A_k}\}, \tag{1.12}$$

$$d_a = \mathcal{H}_m, m = \arg \max_{1 \leq j \leq M} p_{N_a}^j, \tag{1.13}$$

где $A_k \in (0, 1]$.

Пусть наблюдения имеют плотность распределения $f(x, \theta)$. Используя теорему Байеса, апостериорные вероятности могут быть записаны в виде

$$p_n^k = \frac{\pi_k \prod_{i=1}^n f(x_i, \theta_k)}{\sum_{j=1}^M \pi_j \prod_{i=1}^n f(x_i, \theta_j)}, n \geq 1, k \in \mathcal{I} \tag{1.14}$$

Формула (1.14) является выражением для построения M-нарного последовательного критерия, однако на практике удобно использовать данную рекуррентную формулу

$$p_{n+1}^k = P(\theta = \theta_k | x_1, x_2, ..., x_n, x_{n+1}) = \frac{p_n^k f(x_{n+1}, \theta_k)}{\sum_{j=1}^M p_n^j f(x_{n+1}, \theta_j)}, n \geq 0, \tag{1.15}$$

где $p_0^k = \pi_k, k \in \mathcal{I}$.

Величины A_k должны быть выбраны так, чтобы M-нарный критерий имел наперед заданную силу. Для достижения этого существует соотношение, которое показывает связь между A_k и ошибкой данного критерия

$$A_k = \min\{1, \frac{\overline{\alpha}_k^0}{\pi_k}\}, \quad k \in \mathcal{I} \tag{1.16}$$

где $\overline{\alpha}_k^0$ – вероятность неправильного принятия гипотезы $\mathcal{H}_k$.

1.4 Марковское случайное поле

Неориентированный граф $G = (V, E)$ и множество случайных величин $X = (X_v)_{v \in V}$ образуют марковское случайное поле, если выполняются марковские свойства:

- *парное марковское свойство*: любые две несмежные случайные величины являются условно независимыми (при условии всех остальных случайных величин)

$$X_u \perp\!\!\!\perp X_v \quad | \quad X_{V \setminus \{u,v\}} \tag{1.17}$$

- *локальное марковское свойство*: случайная величина является условно независимой со всеми другими случайными величинами (при условии смежных случайных величин)

$$X_v \perp\!\!\!\perp X_{V \setminus \{v \cup N(v)\}} \quad | \quad X_{N(v)} \tag{1.18}$$

- *глобальное марковское свойство*: любые два подмножества случайных величин являются условно независимыми (при условии разделяющего подмножества, т.е. каждый путь из A в B проходит через S)

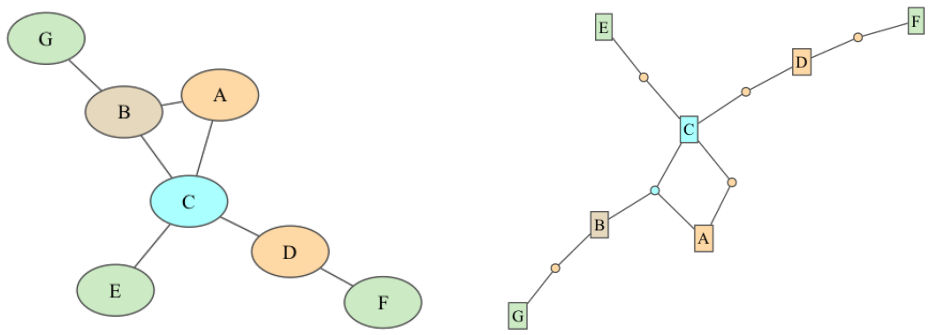
$$X_A \perp\!\!\!\perp X_B \quad | \quad X_S \tag{1.19}$$

Поскольку марковские свойства сложно проверить для произвольного распределения, обычно используется класс марковских случайных полей, распределение которых может быть факторизовано на множестве клик графа $cl(G)$.

Для множества случайных величин $P(X = x)$ это вероятность того, что поле находится в конфигурации x. Пусть заданы функции φ_c, которые называют функциями потенциалов. Совместное распределение вероятностей в таком случае выражается в факторизованном виде формулой

$$P(X = x) = \frac{1}{Z} \prod_{c \in cl(G)} \varphi_c(x_c), \quad Z = \sum_x \prod_{c \in cl(G)} \varphi_c(x_c) \tag{1.20}$$

Теорема *Хаммерсли-Клиффорда* утверждает [3], что строго положительное распределение может быть представлено как марковское случайное поле тогда и только тогда, когда это распределение может быть факторизовано видом (1.20).



(a) Неориентированный граф

(b) Факторный граф

Рис. 1.2: Пример графа марковского случайного поля и его представления в виде факторного графа. Цвет характеризует количество соседей и число переменных в функциях потенциалов.

1.5 Гауссово марковское случайное поле

Множество случайных величин $X = (X_v)_{v \in V}$ образуют гауссово марковское случайное поле $\mathcal{N}(\mu, Q^{-1})$ со средним μ и матрицей точности $Q = \Sigma^{-1} > 0$ по отношению к графу $G = (V, E)$, если совместное распределение может быть представлено в виде [5]

$$P(X = x) = (2\pi)^{-n/2}|Q|^{1/2}exp(-\frac{1}{2}(x-\mu)^T Q(x-\mu)) \tag{1.21}$$

и при отсутствующих рёбрах в графе между вершинами v_i и v_j, где $i \neq j$, элементы матрицы $q_{ij} = 0$.

1.6 Моделирование марковского случайного поля

Изучаемые вероятностные модели часто бывают достаточно сложны и точные методы моделирования могут быть медленными. Для некоторых моделей данная задача, в общем, является NP-трудной [4]. Поэтому для генерации выборки из марковского случайного поля на практике используется метод *семплирования по Гиббсу*. Это позволяет не вычислять напрямую совместную плотность распределения, а аппроксимировать её с помощью условного распределения. Для n-мерной случайной величины $X = (x_1, ..., x_n)$ за $k \leq K$ шагов процесс выглядит так:

- Выбирается некоторое начальное значение $X^{(0)} = (x_1^0, ..., x_n^0)$ случайно или, например, при помощи *EM алгоритма*;

- На k шаге выбирается индекс переменной $1 \leq i \leq n$, которая будет обновлена;
- При фиксированных остальных переменных $x_j, j \neq i$, выбранное x_i удобным образом генерируется из распределения

$$P(x_i | x_1 = x_1^{(k-1)}, ..., x_n = x_n^{(k-1)}) \tag{1.22}$$

- После обновления значения x_i получаем реализацию $X^{(k)}$.

Марковская цепь, получаемая при таком процессе, сходится к истинному совместному распределению марковского случайного поля [6], но количество шагов K для достижения этого быть достаточно большим.

Для моделирования гауссового марковского поля существуют отдельные специальные методы [5].

Один из них состоит в следующем:

- Вычислить разложение *Холецкого* матрицы $Q = LL^T$;
- Сгенерировать случайную величину $z \sim \mathcal{N}(0,\, I)$;
- Найти решение v системы $L^T v = z$;
- После вычисления $X = \mu + v$ получаем реализацию случайно величины X.

1.7 Применение последовательных критериев к марковскому случайному полю

Учитывая вышеизложенное, последовательный критерий для марковского случайного поля будет иметь вид

$$\ln B < \sum_{t=0}^{T-1} \ln \frac{P^{(1)}(X = x_t)}{P^{(0)}(X = x_t)} < \ln A, \quad T > 1, \tag{1.23}$$

где A и B определяются соотношениями (1.10).

В свою очередь, M–нарный последовательный критерий для марковского случайного поля будет иметь вид

$$p_T^k = \frac{\pi_k \prod_{t=0}^{T-1} P^{(k)}(X = x_t)}{\sum_{j=1}^{M} \pi_j \prod_{t=1}^{T-1} P^{(j)}(X = x_t)} < \frac{1}{1 + A_k}, \quad T > 1 \tag{1.24}$$

где A_k определяются соотношениями (1.16).

1.8 Отклонения типа Тьюки-Хьюбера

Данные, поступающие в реальном времени, вероятнее всего, будут «загрязнены». Если существует некоторое «загрязняющее» распределение, которое влияет на значения поступающих наблюдений, то такой тип отклонений называется отклонениями Тьюки-Хьюбера.

Пусть имеется источник наблюдений, подверженный отклонениям Тьюки-Хьюбера [7], распределение которого имеет следующий вид

$$\bar{P}_k(x) = (1-\varepsilon_k)P_k(x) + \varepsilon_k\tilde{P}_k(x), \quad x \in X, \quad k = 0,1 \tag{1.25}$$

где $\varepsilon_k \in [0, \varepsilon_{k+}]$ – неизвестная вероятность «засорения», $\tilde{P}_k(x)$ является «засоряющим» распределением.

Рассмотрим случай двух гипотез

$$\begin{aligned} \mathcal{H}_0: &\quad \bar{P}_0(x) = (1-\varepsilon_{0+})P^{(0)}(x) + \varepsilon_{0+}Q^{(0)}(x), \\ \mathcal{H}_1: &\quad \bar{P}_1(x) = (1-\varepsilon_{1+})P^{(1)}(x) + \varepsilon_{1+}Q^{(1)}(x), \end{aligned} \tag{1.26}$$

Если не нарушаются базовые свойства гипотетический модели,

$$\begin{gathered} \sum_{x\in\tilde{X}} \bar{P}_1(x) > \sum_{x\in\tilde{X}} \bar{P}_0(x), \\ \tilde{X} = \{x : P_1(x) > P_0(x)\}, \end{gathered} \tag{1.27}$$

то можно построить такие $Q^{(0)}$ и $Q^{(1)}$ в гипотезах (1.26), что условные вероятности ошибок будут гарантированно не превосходить теоретических значений [8].

Вводятся два множества индексов K_1 и K_2 таким образом

$$K_1 = \{k \in \{1,...,L-1\} : \frac{(1-\varepsilon_{1+})\sum_{i=1}^{k} P_1(i)}{\varepsilon_{0+} + (1-\varepsilon_{0+})\sum_{i=1}^{k} P_0(i)} > \frac{(1-\varepsilon_{1+})P_1(k+1)}{(1-\varepsilon_{0+})P_0(k+1)}\}$$

$$K_2 = \{k \in \{2,...,L\} : \frac{\varepsilon_{1+} + (1-\varepsilon_{1+})\sum_{i=k}^{L} P_1(i)}{(1-\varepsilon_{0+})\sum_{i=k}^{L} P_0(i)} < \frac{(1-\varepsilon_{1+})P_1(k-1)}{(1-\varepsilon_{0+})P_0(k-1)}\}$$

Согласно принципу минимакса обозначим индексы $k_1 = min(K_1)$ и $k_2 = max(K_2)$, причем доказано, что $k_1 < k_2$.

Подсчитаем значения a_{k_1} и b_{k_2}

$$\begin{aligned} a_{k_1} &= \frac{\varepsilon_{0+} + (1-\varepsilon_{0+})\sum_{i=1}^{k_1} P_0(i)}{(1-\varepsilon_{1+})\sum_{i=1}^{k_1} P_1(i)} \\ b_{k_2} &= \frac{\varepsilon_{1+} + (1-\varepsilon_{1+})\sum_{i=k_2}^{L} P_1(i)}{(1-\varepsilon_{0+})\sum_{i=k_2}^{L} P_0(i)} \end{aligned} \tag{1.28}$$

Отсюда распределения $Q^{(0)}$, $Q^{(1)}$ строятся следующим образом

$$\begin{aligned} Q^{(0)} &= \begin{cases} \frac{a_{k_1}(1-\varepsilon_{1+})P_1(x)-(1-\varepsilon_{0+})P_0(x)}{\varepsilon_{0+}}, & x \le k_1, \\ 0, & x > k_1, \end{cases} \\ Q^{(1)} &= \begin{cases} \frac{b_{k_2}(1-\varepsilon_{0+})P_0(x)-(1-\varepsilon_{1+})P_1(x)}{\varepsilon_{1+}}, & x \ge k_2, \\ 0, & x < k_2, \end{cases} \end{aligned} \tag{1.29}$$

И последовательный критерий будет иметь вид

$$\Lambda = \sum_{t=0}^{T} \ln \frac{(1-\varepsilon_{1+})P^{(1)}(x_t) + \varepsilon_{1+}Q^{(1)}(x_t)}{(1-\varepsilon_{0+})P^{(0)}(x_t) + \varepsilon_{0+}Q^{(0)}(x_t)} \tag{1.30}$$

$$\ln B < \Lambda < \ln A \tag{1.31}$$

где A и B определяются соотношениями (1.10).

1.9 Фильтрация потока данных

В данном пункте будут рассмотрены различные виды фильтрации данных. Они применяются для уменьшения влияния «выбросов» в наблюдениях при известной информации о данных в прошлом.

Простое экспоненциальное сглаживание выражается формулами 1.32, где α – коэффициент сглаживания данных.

$$s_0 = z_0, \quad s_t = \alpha z_t + (1-\alpha)z_{t-1}, \quad t > 0, \quad 0 \le \alpha \le 1 \tag{1.32}$$

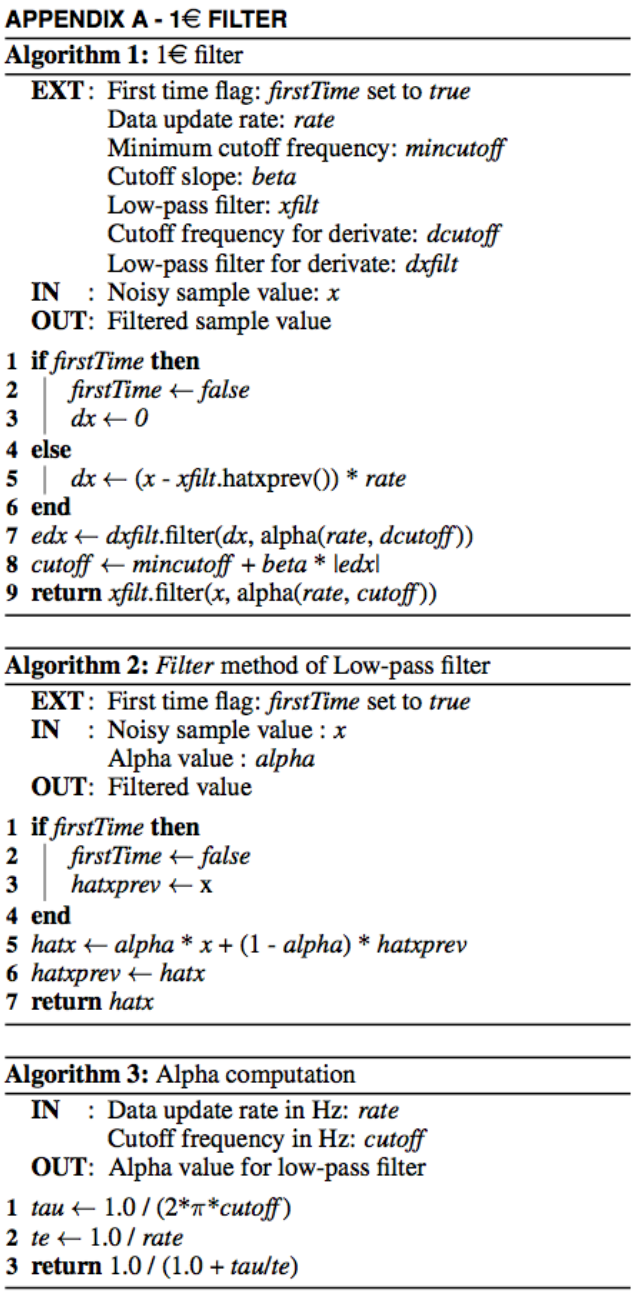
Двойное экспоненциальное сглаживание имеет вид 1.33, где α – также называется коэффициентом сглаживания данных.

$$\begin{gathered} s'_0 = z_0, \quad s''_0 = z_0 \\ s'_t = \alpha z_t + (1-\alpha)s'_{t-1}, \quad t > 0 \\ s''_t = \alpha s'_t + (1-\alpha)s''_{t-1}, \quad t > 0 \\ 0 \le \alpha \le 1 \\ \tilde{z}_{t+m} = a_t + mb_t \\ a_t = 2s'_t - s''_t, \quad b_t = \frac{\alpha}{1-\alpha}(s'_t - s''_t) \end{gathered} \tag{1.33}$$

Фильтр 1€ использует низкочастотный фильтр первого порядка с адаптивной частотой спрямления: на небольшой скорости изменения

низкая частота спрямления уменьшает разброс данных, а как только скорость увеличивается - частота спрямления подстраивается таким образом, чтобы уменьшить отставание от измеряемых данных [12].

Алгоритм фильтрации [13] с помощью этого фильтра изображен на следующем рисунке 1.3.



APPENDIX A - 1€ FILTER

Algorithm 1: 1€ filter

EXT: First time flag: *firstTime* set to *true*
Data update rate: *rate*
Minimum cutoff frequency: *mincutoff*
Cutoff slope: *beta*
Low-pass filter: *xfilt*
Cutoff frequency for derivate: *dcutoff*
Low-pass filter for derivate: *dxfilt*
IN : Noisy sample value: *x*
OUT: Filtered sample value

```
1 if firstTime then
2 |  firstTime ← false
3 |  dx ← 0
4 else
5 |  dx ← (x - xfilt.hatxprev()) * rate
6 end
7 edx ← dxfilt.filter(dx, alpha(rate, dcutoff))
8 cutoff ← mincutoff + beta * |edx|
9 return xfilt.filter(x, alpha(rate, cutoff))
```

Algorithm 2: *Filter* method of Low-pass filter

EXT: First time flag: *firstTime* set to *true*
IN : Noisy sample value : *x*
Alpha value : *alpha*
OUT: Filtered value

```
1 if firstTime then
2 |  firstTime ← false
3 |  hatxprev ← x
4 end
5 hatx ← alpha * x + (1 - alpha) * hatxprev
6 hatxprev ← hatx
7 return hatx
```

Algorithm 3: Alpha computation

IN : Data update rate in Hz: *rate*
Cutoff frequency in Hz: *cutoff*
OUT: Alpha value for low-pass filter

```
1 tau ← 1.0 / (2*π*cutoff)
2 te ← 1.0 / rate
3 return 1.0 / (1.0 + tau/te)
```

Рис. 1.3: 1€ фильтр.

Параметрами 1€ фильтра являются минимальная частота спрямления и его крутизна, а также частота спрямления для производной и длительность времени между поступлениями двух последовательных наблюдений.

Математическое представление данного фильтра:

$$\begin{gathered} \alpha = \frac{1}{1+\frac{\tau}{T_e}} \\ \tau = \frac{1}{2\pi f_c} \\ \hat{X}_i = (X_i + \frac{\tau}{T_e}\hat{X}_{i-1})\frac{1}{1+\frac{\tau}{T_e}} \\ f_c = f_{c_{min}} + \beta|\dot{\hat{X}}_i| \end{gathered} \tag{1.34}$$

Фильтр Калмана состоит из двух последовательных шагов: предсказание и уточнение. Для этапа предсказания строится некоторая математическая модель, которая будет предсказывать следующее наблюдение k, зная информацию о предыдущем наблюдении $k-1$. После этого производится измерение реального наблюдения k и на этапе уточнения берется некоторая смесь этих двух значений: предсказанного и измеренного [14].

Математически фильтр Калмана выражается следующим образом:

$$\begin{gathered} \bar{x} = x * f_x(.) \\ x = \mathcal{L} * \bar{x} \end{gathered} \tag{1.35}$$

Общая схема фильтрации [15] с помощью этого фильтра изображена на следующем рисунке 1.4.

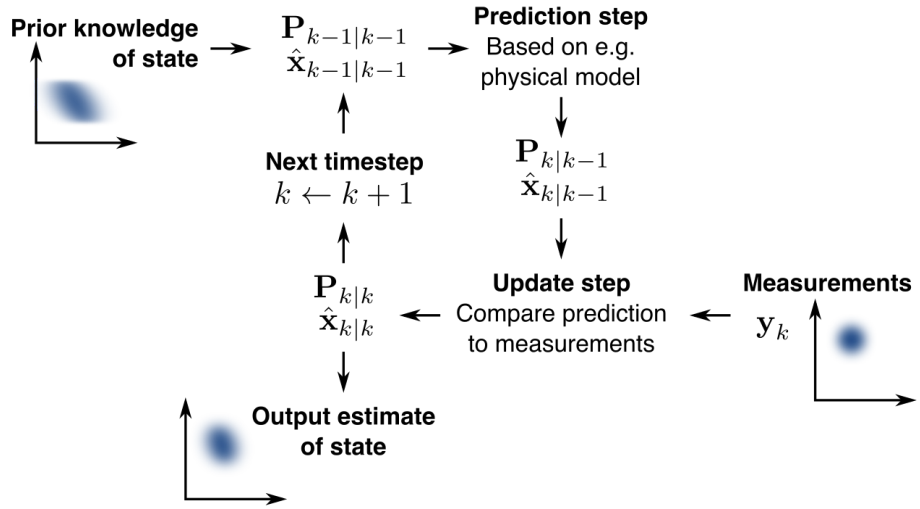


Рис. 1.4: Схема фильтра Калмана.

Параметрами фильтра Калмана являются дисперсия системы, модель предсказания следующего наблюдения, включая значение шума этой модели, а также дисперсия измерений наблюдений.

Глава 2

Последовательные критерии для гауссового марковского поля

2.1 Параметры исследований

Исследование последовательных критериев проводились с помощью метода *Монте-Карло*. Количество итераций во всех экспериментах равнялось $N_{iter} = 10^5$. Максимальное количество поступающих наблюдений ограничивалось числом $N_{max} = 10^9$.

Исследовались численные характеристики последовательных критериев, которыми являются оценки ошибок первого и второго рода, а также математическое ожидание необходимого количества наблюдений построенных критериев для принятия окончательного решения.

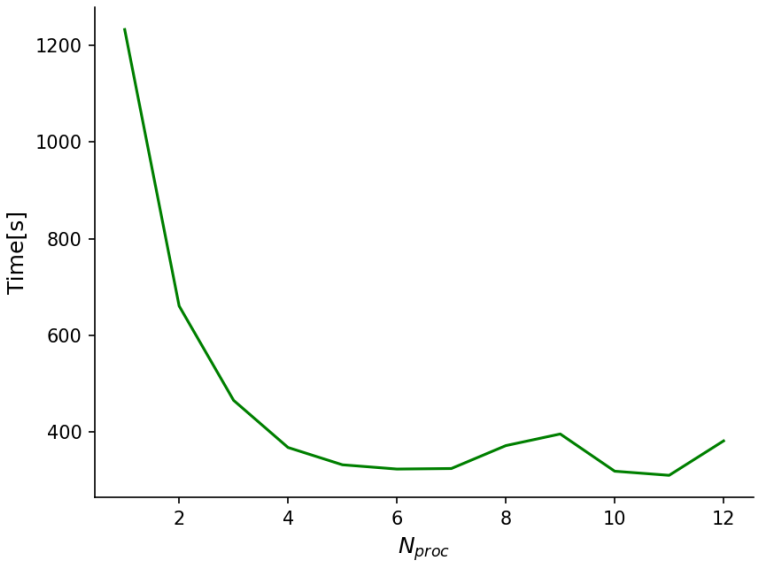


Рис. 2.1: Время выполнения эксперимента
в зависимости от количества процессов
в *Pool* для *multiprocessing*.

Для увеличения скорости получения результатов использовалось несколько процессов в *python* через библиотеку *multiprocessing*. Зависимость времени выполнения одного эксперимента от количества заданных процессов на процессоре с шестью ядрами *Intel Core i7 2.6 GHz* изображена на рисунке 2.1.

Для параллельного моделирования использовалось $N_{proc} = 6$ процессов с меньшим $\frac{N_{iter}}{N_{proc}}$ количеством итераций метода Монте-Карло в каждом из них. Результаты всех процессов объединялись, получая таким образом суммарно N_{iter} итераций, а затем получались оценки последовательных критериев путем их усреднения.

Для воспроизводимости результатов случайное состояние фиксировалось числом $42 + i$, где i – номер процесса.

2.2 Оценка численных характеристик

Последовательный критерий, в частности, для гауссового марковского поля размерности 3, строился в соответствии с неравенством (1.23) для проверки двух гипотез о параметрах поля с $\mu = (0.5, 0.2, 0.3)$:

$$H_0 : Q^{-1} = \begin{pmatrix} 4 & 0 & 3 \\ 0 & 2 & 0 \\ 3 & 0 & 4 \end{pmatrix};$$

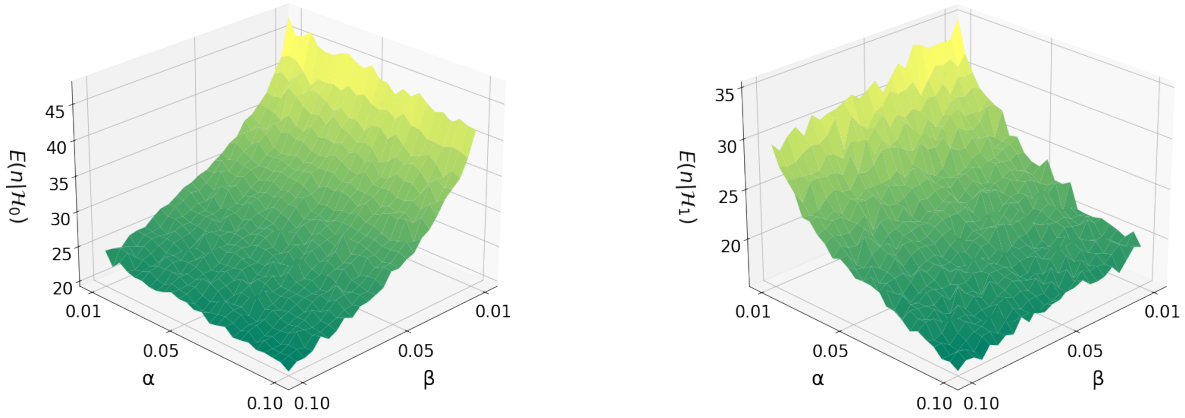
$$H_1 : Q^{-1} = \begin{pmatrix} 4 & 0 & 2 \\ 0 & 2 & 0 \\ 2 & 0 & 4 \end{pmatrix}.$$

Результаты исследования представлены в таблице 2.1. Они показывают, что значения оценок ошибок I и II рода построенного критерия меньше теоретических значений. Также из результатов видно, что при более строгих значениях теоретических ошибок, количество наблюдений $E(n|\mathcal{H}_0)$ и $E(n|\mathcal{H}_1)$, необходимых для принятия решения в ту или иную пользу, растет.

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.0555	0.09	20.3	16.0
0.1	0.05	0.0549	0.0443	26.6	17.8
0.1	0.01	0.0548	0.0088	41.4	19.3
0.05	0.1	0.0281	0.0877	21.9	20.1
0.05	0.05	0.0274	0.0442	28.4	21.9
0.05	0.01	0.0277	0.0085	43.6	23.5
0.01	0.1	0.0051	0.0859	23.5	29.3
0.01	0.05	0.0053	0.0419	30.3	31.5
0.01	0.01	0.0058	0.0087	45.7	33.7

Таблица 2.1: Характеристики критерия.

Более подробная зависимость числа наблюдений от ошибок при справедливости гипотезы H_0 в виде поверхностей представлена на рисунке 2.2a и при справедливости гипотезы H_1 – на рисунке 2.2b.



(a) $E(n|\mathcal{H}_0)$ (b) $E(n|\mathcal{H}_1)$

Рис. 2.2: Зависимость числа наблюдений от ошибок *I* и *II* рода.

2.3 Оценка численных характеристик *M*–нарного последовательного критерия

В дополнении к предыдущим двум гипотезам была добавлена третья гипотеза. В данном случае проверялись три гипотезы о равенстве параметров:

$$H_1 : Q^{-1} = \begin{pmatrix} 4 & 0 & 3 \\ 0 & 2 & 0 \\ 3 & 0 & 4 \end{pmatrix};$$

$$H_2 : Q^{-1} = \begin{pmatrix} 4 & 0 & 2 \\ 0 & 2 & 0 \\ 2 & 0 & 4 \end{pmatrix};$$

$$H_3 : Q^{-1} = \begin{pmatrix} 5 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 5 \end{pmatrix}.$$

Вектор $\mu = (0.5, 0.2, 0.3)$ во всех трех гипотезах одинаков.

Априорная вероятность гипотезы, из которой поступали наблюдения равнялась $\pi_k = 0.5$, а вероятности остальных гипотез равнялись 0.25.

При проведении испытаний построенного M – нарного последовательного критерия получили следующие результаты, которые приведены в таблице 2.2.

α	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\alpha}_3$	$E(n\|\mathcal{H}_1)$	$E(n\|\mathcal{H}_2)$	$E(n\|\mathcal{H}_3)$
0.1	0.011	0.0614	0.0275	11.4	17.0	10.6
0.05	0.0058	0.0336	0.0145	17.7	26.6	16.9
0.01	0.0012	0.0097	0.0046	32.9	47.5	31.7

Таблица 2.2: Характеристики M-нарного критерия.

Оценки ошибок были получены путем взвешенной суммы, например, для второго столбца таким образом: $\hat{\alpha}_1 = \pi_2\hat{\alpha}_{21} + \pi_3\hat{\alpha}_{31}$, где $\hat{\alpha_{21}} = \hat{P}(H_2|H_1)$ и $\hat{\alpha_{31}} = \hat{P}(H_3|H_1)$ – оценки вероятностей принятия гипотез H_2 и H_3 при справедливости гипотезы H_1 соответственно.

Все полученные оценки не превосходят теоретические значения. Наиболее близкими к теоретическим границам значений являются оценки $\hat{\alpha}_2$ из второго столбца.

Глава 3

Последовательные критерии для марковского случайного поля

3.1 Параметры исследований

Исследование последовательных критериев проводились также с помощью метода *Монте-Карло* с теми же значениями количества итераций $N_{iter} = 10^5$ и ограничения числа наблюдений $N_{max} = 10^9$.

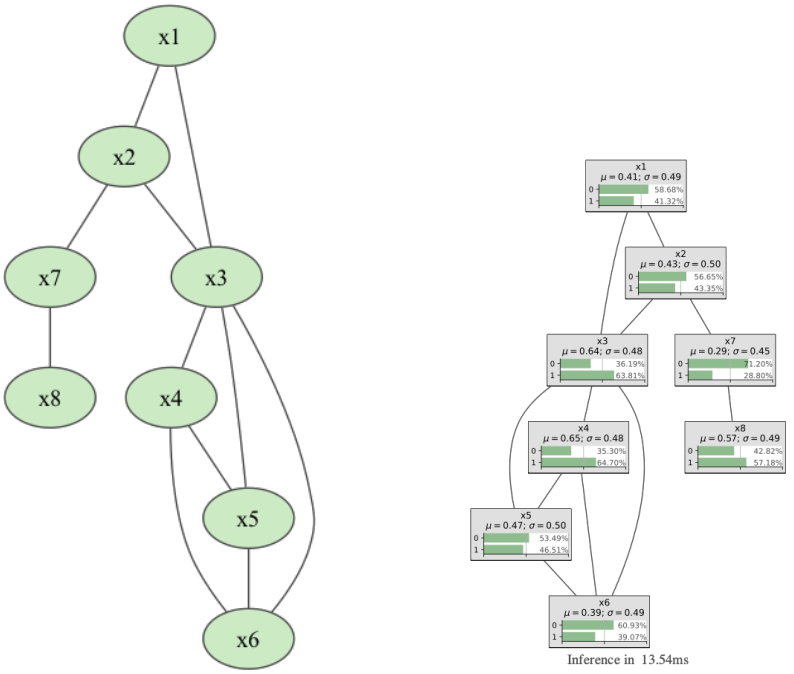
Предметом исследования также являлись оценки ошибок первого и второго рода и математическое ожидание необходимого количества наблюдений для принятия окончательного решения.

Для моделирования и визуализации марковских случайных полей использовались библиотеки *pgmpy* [9] и *pyAgrum* [10]. В них реализовано моделирование поступления событий методом семплирования по Гиббсу, а также визуальное представление случайных полей в различном виде. Визуализация исследуемого марковского поля в виде неориентированного графа представлена на рисунках 3.1.

В данных модулях также присутствует моделирование марковского поля с помощью алгоритма Shafer-Shenoy [11]. В отличие от аппроксимирующего семплирования по Гиббсу данный метод является точным. На рисунке 3.1b изображены результаты этого метода, показывающие маргинальные вероятности состояний каждой вершины.

3.2 Оценка численных характеристик

Были проведены исследования марковского поля с 8 переменными, каждая из которых может принимать только два значения из множества $\{0, 1\}$.



(a) Граф, представляющий связи между переменными

(b) Вероятности переменных после Shafer-Shenoy

Рис. 3.1: Визуализация марковского случайного поля и маргинальные вероятности

Критерий строился в соответствии с неравенством (1.23) для проверки двух гипотез о функциях потенциалов марковского поля.

Функции потенциалов, заданных на указанных в нижнем индексе вершинах графа, для гипотез H_0 и H_1 в виде векторов представлены в таблицах 3.1 и 3.2 соответственно.

	$\phi_{(x_1,x_2,x_3)}$	$\phi_{(x_3,x_4,x_5,x_6)}$	$\phi_{(x_2,x_7)}$	$\phi_{(x_7,x_8)}$
H_0	$\begin{pmatrix} 0.05 \\ 0.96 \\ 0.81 \\ 0.52 \\ 0.51 \\ 0.12 \\ 0.09 \\ 0.99 \end{pmatrix}$	$\begin{pmatrix} 0.48 \\ 0.37 \\ 0.34 \\ 0.45 \\ 0.77 \\ 0.14 \\ 0.83 \\ 0.18 \\ 0.11 \\ 0.48 \\ 0.22 \\ 0.2 \\ 0.81 \\ 0.53 \\ 0.83 \\ 0.28 \end{pmatrix}$	$\begin{pmatrix} 0.84 \\ 0.39 \\ 0.48 \\ 0.16 \end{pmatrix}$	$\begin{pmatrix} 0.57 \\ 0.62 \\ 0.36 \\ 0.83 \end{pmatrix}$

Таблица 3.1: Векторы для функций потенциалов гипотезы H_0.

	$\phi_{(x_1,x_2,x_3)}$	$\phi_{(x_3,x_4,x_5,x_6)}$	$\phi_{(x_2,x_7)}$	$\phi_{(x_7,x_8)}$
H_1	$\begin{pmatrix}0.35\\0.96\\0.81\\0.22\\0.51\\0.12\\0.09\\0.69\end{pmatrix}$	$\begin{pmatrix}0.28\\0.37\\0.34\\0.45\\0.37\\0.14\\0.83\\0.18\\0.11\\0.48\\0.22\\0.2\\0.81\\0.53\\0.63\\0.28\end{pmatrix}$	$\begin{pmatrix}0.74\\0.39\\0.48\\0.36\end{pmatrix}$	$\begin{pmatrix}0.57\\0.62\\0.26\\0.83\end{pmatrix}$

Таблица 3.2: Векторы для функций потенциалов гипотезы H_1.

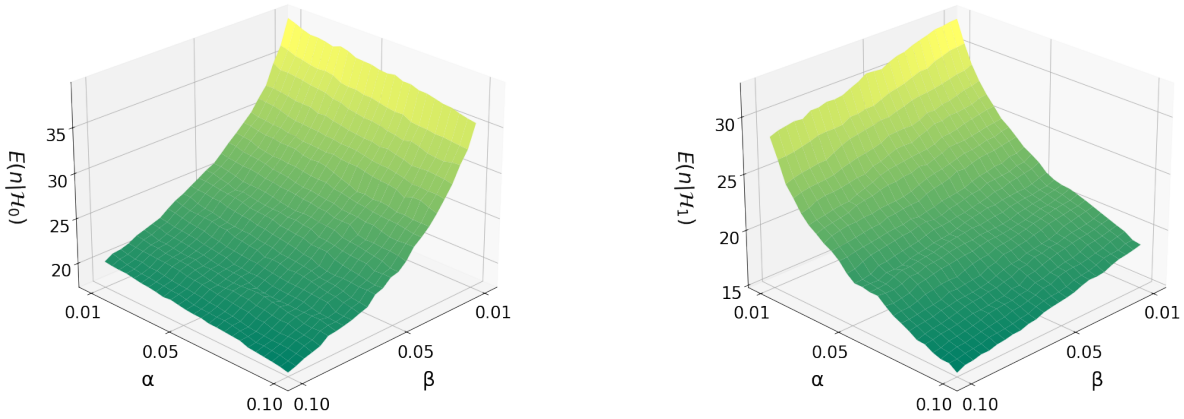
Результаты проведенного эксперимента приведены в таблице 3.3. Исходя из них следует, что оценки ошибок $\hat{\alpha}$ и $\hat{\beta}$ не превосходят теоретических значений. Это значит, что построенный тест имеет наперёд заданную силу (α, β).

Среднее количество наблюдений $E(n|\mathcal{H}_0)$ и $E(n|\mathcal{H}_1)$ увеличивается при уменьшении значений априорных ошибок. Стоит заметить, что изменение значений ошибок первого рода приводит к большему изменению ожидаемого количества наблюдений при условии, что данные поступают из случайного марковского поля, соответствующей параметрам гипотезы $\mathcal{H}_1$.

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.0512	0.0817	18.2	15.1
0.1	0.05	0.0508	0.0409	23.7	16.6
0.1	0.01	0.0497	0.0078	36.4	18.0
0.05	0.1	0.0245	0.0806	19.6	18.9
0.05	0.05	0.0242	0.0403	25.3	20.6
0.05	0.01	0.0239	0.008	38.3	22.1
0.01	0.1	0.0043	0.0787	20.9	28.2
0.01	0.05	0.0041	0.0396	26.6	30.2
0.01	0.01	0.0046	0.0079	39.9	32.1

Таблица 3.3: Характеристики критерия.

Описанные зависимости представлены на графиках 3.2a и 3.2b.



(a) $E(n|\mathcal{H}_0)$ (b) $E(n|\mathcal{H}_1)$

Рис. 3.2: Зависимость числа наблюдений от ошибок I и II рода.

Глава 4

Устойчивость последовательных критериев

4.1 Оценка численных характеристик при отклонениях Тьюки-Хьюбера

Исследуем, что будет происходить с оценками ошибок I и II рода при наличии отклонений в распределении поступающих наблюдений. Изначальные распределения двух гипотез описываются функциями потенциалов марковского случайного поля из таблиц 3.1 и 3.2. В качестве загрязняющего распределения выступает марковское случайное поле с функциями потенциалов, представленных в таблице 4.1.

$\tilde{\phi}_{(x_1,x_2,x_3)}$	$\tilde{\phi}_{(x_3,x_4,x_5,x_6)}$	$\tilde{\phi}_{(x_2,x_7)}$	$\tilde{\phi}_{(x_7,x_8)}$
$\begin{pmatrix} 0.35 \\ 0.96 \\ 0.81 \\ 0.22 \\ 0.51 \\ 0.12 \\ 0.09 \\ 0.69 \end{pmatrix}$	$\begin{pmatrix} 0.28 \\ 0.37 \\ 0.34 \\ 0.45 \\ 0.37 \\ 0.14 \\ 0.83 \\ 0.18 \\ 0.11 \\ 0.48 \\ 0.22 \\ 0.2 \\ 0.81 \\ 0.53 \\ 0.63 \\ 0.28 \end{pmatrix}$	$\begin{pmatrix} 0.84 \\ 0.39 \\ 0.48 \\ 0.16 \end{pmatrix}$	$\begin{pmatrix} 0.57 \\ 0.62 \\ 0.26 \\ 0.83 \end{pmatrix}$

Таблица 4.1: Функции потенциалов «загрязняющего» распределения.

В данном случае значение загрязняющей вероятности ε из выражения 1.26 равнялись 0.08 и 0.15. Параметры для моделирования оставались такими же. Результаты представлены в таблицах 4.2 и 4.3 соответственно.

По результатам видно, что оценки ошибок превосходят теоретические значения во всех строках. Это произошло из-за того, что параметры

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.096	0.0851	19.6	15.2
0.1	0.05	0.0974	0.0434	25.7	16.8
0.1	0.01	0.0983	0.0088	40.1	18.2
0.05	0.1	0.0542	0.0846	21.8	19.1
0.05	0.05	0.0549	0.0425	28.3	20.8
0.05	0.01	0.056	0.0085	43.4	22.4
0.01	0.1	0.0139	0.083	24.4	28.4
0.01	0.05	0.0145	0.0421	31.1	30.5
0.01	0.01	0.015	0.0088	46.9	32.5

Таблица 4.2: Характеристики критерия с отклонениями при $\varepsilon = 0.08$.

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.1445	0.0893	20.7	15.3
0.1	0.05	0.151	0.0454	27.3	16.9
0.1	0.01	0.159	0.0095	43.1	18.5
0.05	0.1	0.0924	0.0884	23.7	19.2
0.05	0.05	0.0973	0.0444	30.9	21.0
0.05	0.01	0.1024	0.0092	47.9	22.7
0.01	0.1	0.0322	0.0861	27.8	28.6
0.01	0.05	0.0344	0.0443	35.7	30.8
0.01	0.01	0.036	0.0092	54.4	32.9

Таблица 4.3: Характеристики критерия с отклонениями при $\varepsilon = 0.15$.

распределения марковского случайного поля, из которого поступали наблюдения «сдвинулись» в сторону параметров конкурирующей гипотезы. Таким образом, последовательный тест больше ошибался и произошло увеличение среднего количества наблюдений.

4.2 Повышение устойчивости при отклонениях Тьюки-Хьюбера

Далее будет описано повышение устойчивости последовательных критериев для случайных марковских полей разными способами. Сначала было проведено моделирование методом Монте-Карло последовательного критерия с модификацией в соответствии с неравенствами (1.30),

(1.31) в логарифмическом виде для проверки двух гипотез:

$$\begin{aligned} H_0 &: (\theta, \theta^*) = (\theta_0(P_0), \theta_0^*(Q_0)); \\ H_1 &: (\theta, \theta^*) = (\theta_1(P_1), \theta_1^*(Q_1)). \end{aligned} \tag{4.1}$$

Функции потенциалов и параметры метода Монте-Карло оставались неизменными, т.е. эквивалентны предыдущему пункту 4.1. Используемые значения $\varepsilon_{0+} = \varepsilon_{1+} = \varepsilon = 0.08$.

Результаты сгруппированы в виде таблицы 4.4. Произошел значительный рост числа необходимых наблюдений для принятия окончательного решения, но оценки ошибок значительно меньше теоретических, а значит критерий имеет наперед заданную силу и является робастным при отклонениях Тьюки-Хьюбера с $\varepsilon = 0.08$.

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.0017	0.0115	65.8	80.7
0.1	0.05	0.0014	0.003	85.8	83.9
0.1	0.01	0.0014	0.0001	132.4	86.0
0.05	0.1	0.0002	0.0107	67.5	105.6
0.05	0.05	0.0003	0.003	87.6	108.9
0.05	0.01	0.0002	0.0001	134.2	111.1
0.01	0.1	0.0	0.0096	68.7	163.2
0.01	0.05	0.0	0.0028	88.8	166.8
0.01	0.01	0.0	0.0001	135.5	169.0

Таблица 4.4: Характеристики устойчивого критерия.

Для вероятностей загрязнения больших, чем 0.08, критерий дает оценки среднего числа наблюдений, которые в два раза превосходят последние полученные, что выглядит очень дорогостоящим решением. Причем, начиная с некоторых значений ε, критерий и вовсе отказывался работать и всегда принимал противоположную гипотезу. Поэтому были рассмотрены следующие методы для более высоких значений вероятности загрязнения, в частности для $\varepsilon = 0.15$.

Результаты для критерия с фильтрацией согласно формулам 1.32 с помощью *простого экспоненциального сглаживания* представлены в таблице 4.5. Параметр сглаживания значений статистики был выбран равным $\alpha = 0.6$.

Результаты для критерия с фильтрацией согласно формулам 1.33 с помощью *двойного экспоненциального сглаживания* представлены в таб-

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.0684	0.0268	39.8	25.7
0.1	0.05	0.0669	0.0086	53.0	27.2
0.1	0.01	0.0657	0.0007	83.3	28.1
0.05	0.1	0.033	0.0251	43.9	33.1
0.05	0.05	0.0315	0.0081	57.7	34.6
0.05	0.01	0.0312	0.0008	88.9	35.5
0.01	0.1	0.0058	0.023	47.6	49.9
0.01	0.05	0.0052	0.0076	61.7	51.6
0.01	0.01	0.006	0.0006	93.7	52.6

Таблица 4.5: Характеристики устойчивого критерия.

лице 4.6. Параметр сглаживания значений статистики был выбран равным $\alpha = 0.1$. В результатах видно, что за счет дополнительной «двойной» фильтрации критерию понадобилось значительно большее количество наблюдений в среднем, чем в предыдущем случае.

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.0311	0.0056	63.3	42.0
0.1	0.05	0.0304	0.0025	73.9	42.9
0.1	0.01	0.0293	0.0007	94.1	43.6
0.05	0.1	0.0178	0.0052	66.0	49.8
0.05	0.05	0.0181	0.0025	76.5	50.6
0.05	0.01	0.0178	0.0006	96.7	51.2
0.01	0.1	0.0079	0.0048	68.2	64.5
0.01	0.05	0.0079	0.002	78.8	65.3
0.01	0.01	0.0074	0.0005	99.0	65.7

Таблица 4.6: Характеристики устойчивого критерия.

Результаты для критерия с фильтрацией согласно формулам 1.34 с помощью *фильтра 1€* представлены в таблице 4.7. Значения параметров были выбраны такими: минимальная частота спрямления 0.8, крутизна спрямления 0.1, частота спрямления для производной 0.2, а длительность между поступлениями новых событий зависит от текущего количества уже поступивших событий $\frac{k}{4}$.

В целом, данный критерий ведет себя похожим образом, как критерий с простым экспоненциальным сглаживанием. При некоторых значениях теоретических ошибок ему понадобилось меньше наблюдений, а

при некоторых – больше.

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.0766	0.0249	40.7	25.0
0.1	0.05	0.0756	0.0078	54.2	26.3
0.1	0.01	0.0729	0.0006	85.6	27.2
0.05	0.1	0.0379	0.0235	45.2	31.9
0.05	0.05	0.0376	0.0072	59.3	33.4
0.05	0.01	0.0371	0.0006	91.6	34.2
0.01	0.1	0.0071	0.0214	49.5	48.0
0.01	0.05	0.0075	0.0069	64.0	49.6
0.01	0.01	0.007	0.0005	97.3	50.5

Таблица 4.7: Характеристики устойчивого критерия.

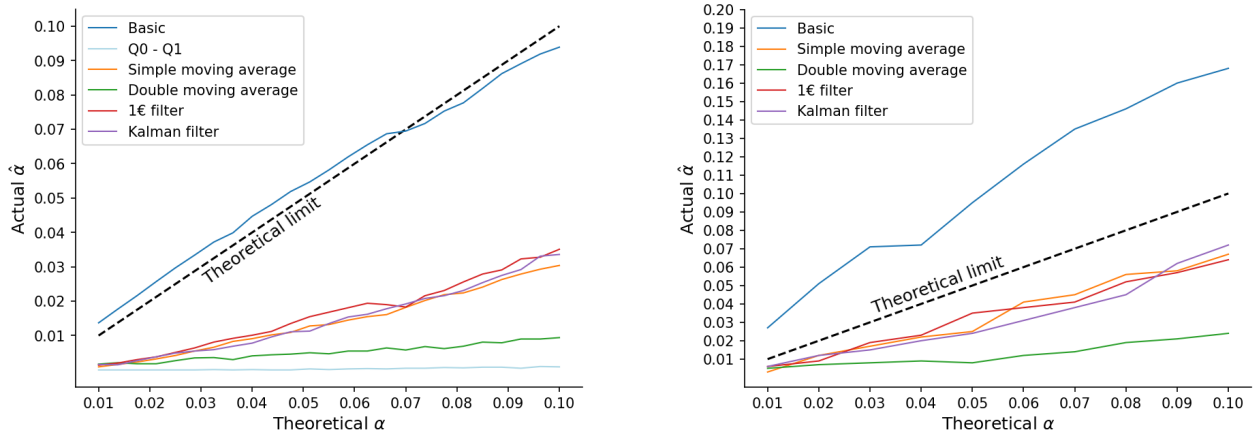
Результаты для критерия с фильтрацией согласно формулам 1.35 с помощью метода Калмана представлены в таблице 4.8. В качестве реализации использовался модуль *filterpy* [16]. Параметры были выбраны следующим образом: модель предсказания состоит чисто из шума с параметром дисперсии 1, дисперсия системы и дисперсия измерений наблюдений также равнялись 1.

В этом случае при справедливости гипотезы H_0 чаще понадобится меньшее количество наблюдений для завершения последовательного теста, чем во всех предыдущих построенных критериях. А фаворитом в наименьшем количестве наблюдений для завершения теста при справедливости гипотезы H_1 стал критерий с *фильтром 1€*.

α	β	$\hat{\alpha}$	$\hat{\beta}$	$E(n\|\mathcal{H}_0)$	$E(n\|\mathcal{H}_1)$
0.1	0.1	0.0698	0.0258	39.3	25.4
0.1	0.05	0.0684	0.0086	52.1	26.8
0.1	0.01	0.0677	0.0008	81.5	27.7
0.05	0.1	0.0333	0.0243	43.4	32.5
0.05	0.05	0.0331	0.008	56.6	34.0
0.05	0.01	0.0315	0.0008	87.1	34.9
0.01	0.1	0.0061	0.022	47.0	48.9
0.01	0.05	0.0059	0.0077	60.6	50.5
0.01	0.01	0.0062	0.0007	91.7	51.5

Таблица 4.8: Характеристики устойчивого критерия.

Сравнение исследованных базового и построенных робастных методов при наличии отклонений в поступающих наблюдениях при двух значениях ε представлено на следующих графиках 4.1a и 4.1b.



(a) $\varepsilon = 0.08$ (b) $\varepsilon = 0.15$

Рис. 4.1: Сравнение оценок ошибки I рода
для последовательных критериев.

Теоретическое значение ошибки II рода в обоих случаях было зафиксировано на уровне 0.05. На графике при $\varepsilon = 0.08$ видно, как, начиная с теоретических значений меньших 0.07, обычный базовый критерий ошибается больше положенного значения, хотя до этого был близок к теоретическому порогу, который изображен черной линией. Остальные критерии в данном случае всегда ошибаются меньше положенного и приближаются к порогу только при достаточно малых значениях теоретических ошибок I рода α.

При $\varepsilon = 0.15$ ситуация меняется и обычный базовый критерий ошибается всегда, а остальные критерии значительно приближаются к теоретическому порогу. Однако, их оценки ошибок все еще остаются в пределах положенных значений и не пересекают заданный порог.

Заметно, что наиболее сильной фильтрацией обладает критерий с *двойным экспоненциальным сглаживанием*, а три *других* критерия, за исключением критерия $Q0 - Q1$ ведут себя довольно схожим образом. Критерий $Q0 - Q1$ дает чрезвычайно низкие значения оценок ошибок I рода, однако он требует слишком много наблюдений и при больших ε перестает работать и почти всегда ошибается, поэтому на втором графике данный критерий отсутствует.

Глава 5

Разработанное программное обеспечение

5.1 Листинг программы

Для проведения исследования и моделирования процесса проверки гипотез было разработано программное обеспечение на языке Python.

Представление параметров проверяемых гипотез.

```
from pgmpy.factors.discrete import DiscreteFactor

H0_GMRF = {
    "mu" : [0.5, 0.2, 0.3],
    "Q_inv" : [
        [4., 0., 3.],
        [0., 2., 0.],
        [3., 0., 4.],
    ]
}

H1_GMRF = {
    "mu" : [0.5, 0.2, 0.3],
    "Q_inv" : [
        [4., 0., 2.],
        [0., 2., 0.],
        [2., 0., 4.],
    ]
}

H2_GMRF = {
    "mu" : [0.5, 0.2, 0.3],
    "Q_inv" : [
        [5., 1., 2.],
        [1., 2., 0.],
        [2., 0., 5.],
    ]
}
```

```
MRF_EDGES = [
    ('x1', 'x2'),
    ('x1', 'x3'),
    ('x2', 'x3'),
    ('x3', 'x4'),
    ('x3', 'x5'),
    ('x3', 'x6'),
    ('x4', 'x5'),
    ('x4', 'x6'),
    ('x5', 'x6'),
    ('x2', 'x7'),
    ('x7', 'x8'),
]

H0_MRF = {
    'edges' : MRF_EDGES,
    'factors' : [
        DiscreteFactor(
            ['x1', 'x2', 'x3'], [2, 2, 2],
            [0.05, 0.96, 0.81, 0.52, 0.51, 0.12, 0.09, 0.99]
        ),
        DiscreteFactor(
            ['x3', 'x4', 'x5', 'x6'], [2, 2, 2, 2],
            [0.48, 0.37, 0.34, 0.45, 0.77, 0.14, 0.83, 0.18, 0.11, 0.48,
                0.22, 0.2 , 0.81, 0.53, 0.83, 0.28]
        ),
        DiscreteFactor(['x2', 'x7'], [2, 2], [0.84, 0.39, 0.48, 0.16]),
        DiscreteFactor(['x7', 'x8'], [2, 2], [0.57, 0.62, 0.36, 0.83]),
    ]
}

H1_MRF = {
    'edges' : MRF_EDGES,
    'factors' : [
        DiscreteFactor(
            ['x1', 'x2', 'x3'], [2, 2, 2],
            [0.35, 0.96, 0.81, 0.22, 0.51, 0.12, 0.09, 0.69]
        ),
        DiscreteFactor(
            ['x3', 'x4', 'x5', 'x6'], [2, 2, 2, 2],
            [0.28, 0.37, 0.34, 0.45, 0.37, 0.14, 0.83, 0.18, 0.11, 0.48,
                0.22, 0.2 , 0.81, 0.53, 0.63, 0.28]
        ),
        DiscreteFactor(['x2', 'x7'], [2, 2], [0.74, 0.39, 0.48, 0.36]),
        DiscreteFactor(['x7', 'x8'], [2, 2], [0.57, 0.62, 0.26, 0.83]),
    ]
```

```
}

H2_MRF = {
    'edges' : MRF_EDGES,
    'factors' : [
        DiscreteFactor(
            ['x1', 'x2', 'x3'], [2, 2, 2],
            [0.35, 0.96, 0.81, 0.22, 0.51, 0.12, 0.09, 0.69]
        ),
        DiscreteFactor(
            ['x3', 'x4', 'x5', 'x6'], [2, 2, 2, 2],
            [0.28, 0.37, 0.34, 0.45, 0.37, 0.14, 0.83, 0.18, 0.11, 0.48,
                0.22, 0.2 , 0.81, 0.53, 0.63, 0.28]
        ),
        DiscreteFactor(['x2', 'x7'], [2, 2], [0.84, 0.39, 0.48, 0.16]),
        DiscreteFactor(['x7', 'x8'], [2, 2], [0.57, 0.62, 0.26, 0.83]),
    ]
}
```

Реализация классов марковских случайных полей и гауссовых случайных полей.

```
import numpy as np
from scipy.stats import norm, multivariate_normal

from pgmpy.models import MarkovNetwork
from pgmpy.sampling import GibbsSampling

class GMRF:
    def __init__(self, mu, Q_inv):
        self.mu = mu
        self.Q = np.linalg.inv(Q_inv)
        self.Q_inv = Q_inv
        self.Q_det = np.linalg.det(self.Q)
        self.Q_inv_det = np.linalg.det(self.Q_inv)
        self.L = np.linalg.cholesky(self.Q)
        self.Lt = self.L.T
        self.norm_coefficient = (2*np.pi)**(-len(self.mu)*0.5) *
            self.Q_det**(0.5)
        self.x_prev = np.zeros(len(self.mu))
        self.h = np.dot(self.Q, self.mu)

    def sample(self):
        z = np.random.multivariate_normal(mean=np.zeros(len(self.mu)),
            cov=np.diag(np.ones(len(self.mu))), size=1)[0]
        v = np.linalg.solve(self.Lt, z)
        return self.mu + v
```

```
    def generator(self):
        while True:
            yield self.sample()

    def proba(self, x):
        return self.norm_coefficient *
            np.exp(-0.5 * np.dot(x - self.mu, np.dot(self.Q, x - self.mu)))

    def get_params(self, x, i):
        s1, s2 = 0, 0
        for j in range(i):
            s1 += self.Q[j][i] * x[j]
        for j in range(i + 1, len(self.mu)):
            s2 += self.Q[j][i] * self.x_prev[j]
        loc = 1 / self.Q[i][i] * self.h[i] - s1 - s2
        scale = 1 / self.Q[i][i] ** 0.5
        return loc, scale

    def gibbs_sampling(self):
        x = np.zeros_like(self.x_prev)
        for i in range(len(x)):
            loc, scale = self.get_params(x, i)
            x[i] = norm(loc=loc, scale=scale).rvs()
        return x


class MRF:
    def __init__(self, edges, factors):
        self.model = MarkovNetwork(edges)
        assert(len(factors) > 0)
        self.model.add_factors(*factors)

        self.factor_product = factors[0].copy()
        for i in range(1, len(factors)):
            self.factor_product.product(factors[i], inplace=True)

        self.factor_product.normalize(inplace=True)

        self.sampler = GibbsSampling(self.model)

    def generator(self, max_sample_size, seed):
        return self.sampler.generate_sample(size=max_sample_size, seed=seed)

    def proba(self, x):
        return self.factor_product.get_value(
                **{item.var : item.state for item in x})
```

Функции построенных последовательных критериев, в том числе устойчивых последовательных критериев.

```
import numpy as np

def smoothing_factor(t_e, cutoff):
    r = 2 * np.pi * cutoff * t_e
    return r / (r + 1)


def exponential_smoothing(a, x, x_prev):
    return a * x + (1 - a) * x_prev


class OneEuroFilter(object):
    def __init__(self, x0, dx0=0.0, min_cutoff=.05, beta=0.2,
                 d_cutoff=0.2, init_time=0):
        """Initialize the one euro filter."""
        # The parameters.
        self.data_shape = x0.shape
        self.min_cutoff = np.full(x0.shape, min_cutoff)
        self.beta = np.full(x0.shape, beta)
        self.d_cutoff = np.full(x0.shape, d_cutoff)
        # Previous values.
        self.x_prev = x0.astype(np.float32)
        self.dx_prev = np.full(x0.shape, dx0)
        self.t_prev = init_time

    def __call__(self, x, t):
        """Compute the filtered signal."""
        assert x.shape == self.data_shape

        t_e = t - self.t_prev
        t_e = np.full(x.shape, t_e)

        # The filtered derivative of the signal.
        a_d = smoothing_factor(t_e, self.d_cutoff)
        dx = (x - self.x_prev) / t_e
        dx_hat = exponential_smoothing(a_d, dx, self.dx_prev)

        # The filtered signal.
        cutoff = self.min_cutoff + self.beta * np.abs(dx_hat)
        a = smoothing_factor(t_e, cutoff)
        x_hat = exponential_smoothing(a, x, self.x_prev)

        # Memorize the previous values.
        self.x_prev = x_hat
        self.dx_prev = dx_hat
```

```
        self.t_prev = t

        return x_hat


def sprt(observer, h0, h1, alpha, beta, max_sample_size):
    A = np.log((1 - beta) / alpha)
    B = np.log(beta / (1 - alpha))

    x, n = next(observer), 1
    p0 = h0.proba(x)
    if p0 == 0:
        return 1, n
    p1 = h1.proba(x)
    if p1 == 0:
        return 0, n
    z = np.log(p1) - np.log(p0)

    while B < z and z < A and n < max_sample_size:
        x, n = next(observer), n + 1
        p0 = h0.proba(x)
        if p0 == 0:
            return 1, n
        p1 = h1.proba(x)
        if p1 == 0:
            return 0, n
        z += np.log(p1)
        z -= np.log(p0)

    if n == max_sample_size:
        return 2, n

    return int(z >= A), n


def msprt(observer, h, prior, alpha, max_sample_size):
    A = np.clip(np.array(alpha) / np.array(prior), a_min=None, a_max=1.)
    threshold = np.log(1 / (1 + A))

    x, n = next(observer), 1
    next_proba = np.array([hm.proba(x) for hm in h])
    proba = np.log(prior) + np.log(next_proba)
    proba_ = np.array(prior) * next_proba

    mask = (proba - np.log(proba_.sum())) > threshold

    while not sum(mask) and n < max_sample_size:
```

```
            x, n = next(observer), n + 1
            next_proba = np.array([hm.proba(x) for hm in h])
            proba += np.log(next_proba)
            proba_ *= np.array(next_proba)

            mask = (proba - np.log(proba_.sum())) > threshold

        if n == max_sample_size:
            return len(h) + 1, n

        max_proba_idx = np.argmax((proba - np.log(proba_.sum()))[mask])
        idxs = np.where(mask)[0]
        return idxs[max_proba_idx], n


    def sprt_1euro(observer, h0, h1, alpha, beta, max_sample_size):
        A = np.log((1 - beta) / alpha)
        B = np.log(beta / (1 - alpha))

        x, n = next(observer), 1
        p0 = h0.proba(x)
        if p0 == 0:
            return 1, n
        p1 = h1.proba(x)
        if p1 == 0:
            return 0, n
        z = np.log(p1) - np.log(p0)
        one_euro_filter = OneEuroFilter(z, n/4, min_cutoff=0.8,
            beta=0.1, d_cutoff=0.2)

        while B < z and z < A and n < max_sample_size:
            x, n = next(observer), n + 1
            p0 = h0.proba(x)
            if p0 == 0:
                return 1, n
            p1 = h1.proba(x)
            if p1 == 0:
                return 0, n
            z += np.log(p1)
            z -= np.log(p0)
            z = one_euro_filter(z, n/4)

        if n == max_sample_size:
            return 2, n

        return int(z >= A), n
```

```
import scipy.ndimage as ndi

def running_mean_uniform_filter1d(x, N):
    return ndi.uniform_filter1d(x, N, mode='constant',
        origin=-(N//2))[:-(N-1)]


def running_mean_cumsum(x, N):
    cumsum = np.cumsum(np.insert(x, 0, 0))
    return (cumsum[N:] - cumsum[:-N]) / float(N)


def sprt_moving_average(observer, h0, h1, alpha, beta,
    max_sample_size):
    A = np.log((1 - beta) / alpha)
    B = np.log(beta / (1 - alpha))

    prev_z = []
    weighting_factor = 0.6

    x, n = next(observer), 1
    p0 = h0.proba(x)
    if p0 == 0:
        return 1, n
    p1 = h1.proba(x)
    if p1 == 0:
        return 0, n
    z = np.log(p1) - np.log(p0)
    prev_z.append(z)

    while B < z and z < A and n < max_sample_size:
        x, n = next(observer), n + 1
        p0 = h0.proba(x)
        if p0 == 0:
            return 1, n
        p1 = h1.proba(x)
        if p1 == 0:
            return 0, n
        z += np.log(p1)
        z -= np.log(p0)

        if len(prev_z) >= 1:
            z = z * weighting_factor + prev_z[-1] * (1 - weighting_factor)
            prev_z.pop(0)

        prev_z.append(z)
```

```
    if n == max_sample_size:
        return 2, n

    return int(z >= A), n


def sprt_double_moving_average(observer, h0, h1, alpha, beta,
    max_sample_size):
    A = np.log((1 - beta) / alpha)
    B = np.log(beta / (1 - alpha))
    weighting_factor = 0.1

    x, n = next(observer), 1
    p0 = h0.proba(x)
    if p0 == 0:
        return 1, n
    p1 = h1.proba(x)
    if p1 == 0:
        return 0, n
    z = np.log(p1) - np.log(p0)

    s1 = [z]
    s2 = [z]

    while B < z and z < A and n < max_sample_size:
        x, n = next(observer), n + 1
        p0 = h0.proba(x)
        if p0 == 0:
            return 1, n
        p1 = h1.proba(x)
        if p1 == 0:
            return 0, n
        z += np.log(p1)
        z -= np.log(p0)

        s1.append(weighting_factor * z + (1 - weighting_factor) * s1[-1])
        s2.append(weighting_factor * s1[-1] + (1 - weighting_factor) * s2[-1])
        a = 2 * s1[-1] - s2[-1]
        b = weighting_factor / (1 - weighting_factor) * (s1[-1] - s2[-1])
        z = a + b

    if n == max_sample_size:
        return 2, n

    return int(z >= A), n
```

```
import filterpy.kalman as kf

def sprt_kalman(observer, h0, h1, alpha, beta, max_sample_size):
    A = np.log((1 - beta) / alpha)
    B = np.log(beta / (1 - alpha))

    x, n = next(observer), 1
    p0 = h0.proba(x)
    if p0 == 0:
        return 1, n
    p1 = h1.proba(x)
    if p1 == 0:
        return 0, n
    x_, P = kf.predict(x=0, P=1, u=0, Q=1)
    z = np.log(p1) - np.log(p0)
    z, P = kf.update(x=x_, P=P, z=z, R=1)

    while B < z and z < A and n < max_sample_size:
        x, n = next(observer), n + 1
        p0 = h0.proba(x)
        if p0 == 0:
            return 1, n
        p1 = h1.proba(x)
        if p1 == 0:
            return 0, n
        x_, P = kf.predict(x=z, P=P, u=0, Q=1)
        z += np.log(p1) - np.log(p0)
        z, P = kf.update(x=x_, P=P, z=z, R=1)

    if n == max_sample_size:
        return 2, n

    return int(z >= A), n


def monte_carlo(f, iterations):
    return [f() for _ in range(iterations)]


def get_Q0_Q1(factor_product0, factor_product1, eps0, eps1):
    assert(np.prod(factor_product0.values.shape) ==
        np.prod(factor_product1.values.shape))
    assert(factor_product0.state_names == factor_product1.state_names)

    res = []
    s = {}
    for i in range(np.prod(factor_product0.values.shape)):
```

```
        row = factor_product0.assignment([i])[0]
        for item in row:
            s[item[0]] = item[1]
        res.append([factor_product1.get_value(**s) /
            factor_product0.get_value(**s), s.copy()])
    sorted_states = np.array(sorted(res, key=lambda x: -x[0]))[:,1]

    k1 = -1
    a_k1 = 0
    s = {}
    L = np.prod(factor_product0.values.shape)
    for k in range(L - 1):
        p1, p0 = 0, 0
        for i in range(k):
            p1 += factor_product1.get_value(**sorted_states[i])
            p0 += factor_product0.get_value(**sorted_states[i])

        lhs = (1 - eps1) * p1 / (eps0 + (1 - eps0) * p0)

        rhs = (1 - eps1) * factor_product1.get_value(**sorted_states[k + 1]) /
            ((1 - eps0) * factor_product0.get_value(**sorted_states[k + 1]))

        if lhs > rhs:
            a_k1 = 1 / lhs
            k1 = k
            break

    k2 = -1
    b_k2 = 0
    s = {}
    for k in range(1, L):
        p1, p0 = 0, 0
        for i in range(k, L):
            p1 += factor_product1.get_value(**sorted_states[i])
            p0 += factor_product0.get_value(**sorted_states[i])

        lhs = (eps1 + (1 - eps1) * p1) / ((1 - eps0) * p0)

        rhs = (1 - eps1) * factor_product1.get_value(**sorted_states[k - 1]) /
            ((1 - eps0) * factor_product0.get_value(**sorted_states[k - 1]))

        if lhs < rhs:
            b_k2 = lhs
            k2 = k

    Q0 = np.zeros(L)
    for i in range(k1):
```

```
            p1 = factor_product1.get_value(**sorted_states[i])
            p0 = factor_product0.get_value(**sorted_states[i])

            Q0[i] = (a_k1 * (1 - eps1) * p1 - (1 - eps0) * p0) / eps0

        Q1 = np.zeros(L)
        for i in range(k2, L):
            p1 = factor_product1.get_value(**sorted_states[i])
            p0 = factor_product0.get_value(**sorted_states[i])

            Q1[i] = (b_k2 * (1 - eps0) * p0 - (1 - eps1) * p1) / eps1

        factor_product_q0 = factor_product0.copy()
        for i in range(len(Q0)):
            factor_product_q0.set_value(Q0[i], **sorted_states[i])

        factor_product_q1 = factor_product1.copy()
        for i in range(len(Q1)):
            factor_product_q1.set_value(Q1[i], **sorted_states[i])

        return factor_product_q0, factor_product_q1


    def mix(lhs, rhs, eps):
        assert(np.prod(lhs.values.shape) == np.prod(rhs.values.shape))
        assert(lhs.state_names == rhs.state_names)

        states = []
        s = {}
        for i in range(np.prod(lhs.values.shape)):
            row = lhs.assignment([i])[0]
            for item in row:
                s[item[0]] = item[1]
            states.append(s.copy())

        mixed = lhs.copy()
        for i in range(len(states)):
            value = (1 - eps) * lhs.get_value(**states[i]) +
                eps * rhs.get_value(**states[i])
            mixed.set_value(value, **states[i])

        return mixed


    def is_valid_base_property(factor_product0, factor_product1,
        factor_product0_, factor_product1_):
        U = []
```

```
s = {}
for i in range(np.prod(factor_product0.values.shape)):
    row = factor_product0.assignment([i])[0]
    for item in row:
        s[item[0]] = item[1]
    if factor_product1.get_value(**s) > factor_product0.get_value(**s):
        U.append(s.copy())

if not len(U):
    return False

s0, s1 = 0, 0
for item in U:
    s0 += factor_product0_.get_value(**item)
    s1 += factor_product1_.get_value(**item)

return s1 > s0
```

ЗАКЛЮЧЕНИЕ

В данной работе была рассмотрена процедура проверки статистических гипотез в последовательном анализе, которая впервые была предложенная Вальдом. В частности, были в подробностях рассмотрены *последовательный критерий отношения вероятностей* и *M–нарный последовательный критерий отношения вероятностей* для случая статистической проверки трёх гипотез.

Показано построение и реализация данных последовательных критериев проверки статистических гипотез о параметрах марковского случайного поля и гауссового марковского случайного поля.

Исследовано влияние погрешности в измерениях поступающих наблюдений в случае, когда существует влияние некоторого иного, «загрязняющего» распределения. Исследовано влияние такого «загрязняющего» распределения в зависимости от разной вероятности загрязнения ε поступающих наблюдений.

В результате работы были получены оценки численных характеристик построенных последовательных критериев и продемонстрированы зависимости в виде графиков плоскостей. Показана сходимость последовательных критериев для частных случаев марковских случайных полей и гауссовых марковских случайных полей.

Дальнейшими шагами в исследовании может являться сравнение полученных результатов последовательных критериев с равными им по надежности непоследовательными критериями в применении к различным марковским случайным полям для получения оценки выигрыша между последовательными и непоследовательными. В данной работе были рассмотрены критерии статистической проверки только простых гипотез, поэтому в контексте следующих интересов стоит упомянуть об последовательном критерии проверки сложных гипотез, как обобщение рассмотренных случаев.

Стоит продолжить изучать робастность последовательного критерия, т.е. устойчивость к «засорениям» или различного рода помехам, в применении к марковским случайным полям, а также условия применения построенных устойчивых последовательных критериев.

Во всех описанных ситуациях интересны оценки условного мате-

матического ожидания числа количества наблюдений для завершения последовательных критериев и оценки ошибок I и II рода, их численное сравнение.

Результаты проведённого исследования были представлены в качестве доклада на конференции студентов и аспирантов факультета прикладной математики и информатики Белорусского государственного университета.

ЛИТЕРАТУРА

[1] Mukhopadhyay, N., de Silva, B.M. (2008). Sequential Methods and Their Applications (1st ed.). Chapman and Hall/CRC. – 504 p.

[2] Вальд, А. Последовательный анализ / А. Вальд – Москва: Государственное издание физико-математической литературы, 1960 – 328 с.

[3] Hammersley J. M., Clifford P. Markov fields on finite graphs and lattices, 1971

[4] Chaohui Wang, Nikos Komodakis, Nikos Paragios. Markov Random Field Modeling, Inference & Learning in Computer Vision & Image Understanding: A Survey. Computer Vision and Image Understanding, 2013, 117 (11), pp.Page 1610-1627

[5] H. Rue, L. Held, Gaussian Markov random fields: theory and applications. Chapman and Hall/CRC, 2005 – 280 p.

[6] Casella, G., George, E. I. (1992). Explaining the Gibbs Sampler. The American Statistician, 46(3), 167–174 p.

[7] Хьюбер П. Робастность в статистике. Москва, 1984. –- 304 с.

[8] Харин А.Ю., Кишилов Д.В. Устойчивый последовательный тест проверки гипотез о дискретных распределениях при наличии «выбросов». Журнал Пермского государственного университета. Выпуск 18. 2005 – 34-42 с.

[9] Ankur Ankan, Johannes Textor (2024). pgmpy: A Python Toolkit for Bayesian Networks. Journal of Machine Learning Research, 25(265), 1–8.

[10] Ducamp, Gaspard and Gonzales, Christophe and Wuillemin, Pierre-Henri. aGrUM/pyAgrum : a Toolbox to Build Models and Algorithms for Probabilistic Graphical Models in Python. 10th International Conference on Probabilistic Graphical Models. 2020 – 609-612 p.

[11] Shenoy, P.P., Shafer, G. (2008). Axioms for Probability and Belief-Function Propagation. In: Yager, R.R., Liu, L. (eds) Classic Works of the Dempster-Shafer Theory of Belief Functions. Studies in Fuzziness and Soft Computing, vol 219. Springer, Berlin, Heidelberg. – 499–528 p.

[12] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). Association for Computing Machinery, New York, NY, USA, 2527-2530

[13] 1€ Filter [Electronic resource]. – Mode of access: https://gery.casiez.net/1euro/. - Date of access: 17.05.2025.

[14] Kalman and Bayesian Filters in Python by Roger R Labbe Jr [Electronic resource]. – Mode of access: https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/. - Date of access: 17.05.2025.

[15] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems", Trans. ASME, Journal of Basic Engineering, pp. 35-45, March 1960.

[16] FilterPy - Kalman filters and other optimal and non-optimal estimation filters in Python [Electronic resource]. – Mode of access: https://github.com/rlabbe/filterpy. - Date of access: 17.05.2025.