МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.04.02 Прикладная математика и информатика
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему Разработка алгоритма идентификации субъектов на основе теории систем со случайной структурой

Студент М.А. Михайлов
(И.О. Фамилия) (личная подпись)

Научный руководитель к.т.н., доцент, А.Б. Кузьмичев
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Оглавление

Введение

В современном мире задача идентификации субъектов уже с давних времен исследуются, для того чтобы можно было определять, что за субъект находится перед вами. Это очень популярно, как в военной промышленности, так и рамках государства.

На данный момент это применяется для обнаружения людей, машин, воздушных целей и в медицине.

Во много в мире используются нейронные сети которые позволяют формировать на базе некоторых входных данных результат. Их можно использовать повсеместно из-за их простоты развертывания, но они предполагают длительный промежуток обучения и ресурсов, и их точность будет зависеть от степени их обучения и грамотного построения нейронных связей.

Зачастую это очень трудный процесс, и по обучению образуется некий черный ящик, который начинает сам по себе функционировать, где достигает определенной точности решения, где последующее обучение нейронной сети будет уже не столь эффективным.

Для работы с физическим миром, где зачастую построение нейронных сетей не является эффективным, из-за больших энергозатрат и достижения приемлемой точности результата, используются различные методы нахождения ответа для тех или иных задач. Например численные методы, марковские процессы и различные фильтрации.

Они позволяют давать более точные результаты, в минимальный промежуток времени, но для использования этих методов требуются определенный набор знаний, для взаимодействиями с этими методами. В работе в качестве субъекта будет рассматриваться человек.

Таким образом, **практическая актуальность** работы обусловлена необходимостью разработки новых методов идентификации субъектов, которые будут работать отдельно или совместно с текущими системами

наблюдения.

Степень разработанности темы исследования. На данный момент есть очень много работ по поводу идентификации различных субъектов, например в работах авторов Чабан Л.Н.[30], А. И. Соколова, Конушин А. С.[31], Тропченко А. А.[32]. Где за основу берутся изображения и идентификация происходит в пиксельном пространстве без сбора отдельных параметров. А таких работ автором как Носов А. В.[8], Anwar, A. S в меньшенстве, где за основу берут изображения, но вычленяют отдельные характеристики, которые уже сравниваются не так много, или они разработаны для узкого направления. Обилие работ по идентификации людей или субъектов много, но мало работ основано на работе с параметрами. Первый способ имеет ряд проблем связанные с ограничением камеры, поэтому например для определение подлинности подписи идет совокупность методов, так как учитывается еще глубина и угол нажатия пишущим средством. Поэтому разработка новых методов в области наблюдения и идентификации субъектов является актуальной научной задачей.

Объект исследования: информационно-измерительная система наблюдения канала данных, основанная на теории систем со случайной структурой.

Предмет исследования: устойчивость и точность алгоритмического обеспечения системы наблюдения канала данных, основанной на теории систем со случайной структурой.

Методология и методы исследования: теория систем со случайно изменяющейся структурой, баесовские фильтрации, математического моделирования сложных систем, теория марковских процессов, компьютерного моделирования, метод обобщенного отношения правдоподобия.

Цель работы: повышение точности идентификации субъектов за счет разработки алгоритма основанного на теории систем со случайной структурой.

Для достижения поставленной цели были **решены следующие задачи**:

— проведен анализ категорий систем идентификаций и их строения;

— проведен анализ биометрических идентификаторов;

— разработан алгоритм, реализующий модернизированный метод распознавания баесовские фильтрации;

— исследована эффективность системы с разными вводными параметрами.

Научная новизна. Разработан алгоритм, реализующий метод идентификации субъектов на основе теории систем со случайной структурой, с применением модифицированного Байесовского фильтра.

Положения, выносимые на защиту:

— разработан алгоритм, реализующий модернизированный метод распознавания баесовские фильтрации;

— алгоритмическое обеспечение для генерации данных, и реализующий метод модифицированного Байесовского фильтра.

Теоретическая и практическая значимость работы. Модернизирован метод Байесовского фильтра. Разработана программа реализующий его.

Структура и объем диссертации. Магистерская диссертационная работа состоит из введения, четырех глав, заключения, списка литературы, приложения. Работа изложена на 118 страницах, имеет 17 рисунков и 9 таблиц. Библиографический список литературы имеет 34 наименований.

Во введении показана актуальность работы, сформулированы цель и задачи исследования; представлены: научная новизна, практическая значимость результатов работы, положения, выносимые на защиту; аннотация работы по главам.

В первой главе рассматривается какие существуют методы работы с алгоритмами, а также рассмотрены возможности работы с ними.

Вторая глава рассматриваются свойства биометрических идентификаторов, а также выводится их практическая оценка. Так же в этой главе рассматривается, анализируется и проектируется метод по работе с

нахождением совпадением с попадающимися классами объекта в потоке данных.

В третьей главе рассматривается перевод математической модели в практическую, а также создание начальных тестовых значений.

В четвертой идет описание созданной программы и её результат.

В заключении приведены результаты работы и основные выводы, указано на решение общей научной задачи и достижение поставленной цели.

В приложение вынесен листинг кода реализации классов.

Глава 1 Анализ подходов к идентификации субъектов

1.1 Процесс идентификации субъектов

Процесс идентификации субъектов на данный момент очень сильно развито в мире и не является новшеством.

На данный момент можно выделить основные 3 группы по которым можно аутентифицировать человека:

1. По физическому предмету — любая физическая вещь, которая принадлежит конкретному человеку или относится к нему, таким как ключи, паспорт, смарт-карты или данные на носителе, которые доступны только этому человеку.

2. По знаниям — любая информация, которая хранится в секрете и знание о которой, есть только у определенного человека или группе лиц, например пароль, секретная фраза или пин-код. Эти знания так же могут не представлять конфиденциальную информацию, которая может быть и не секретной, например девичья фамилия матери или любимый цвет.

3. По биометрическим параметрам — это поведенческое или физиологическая характеристика человека. Это части человеческого тела или действия, по которым можно различать людей друг от друга.

Формально можно использовать любую особенность, которая является индивидуальной.

Эти три способа аутентифицирования можно использовать, как по раздельно, так и в совокупности, например: ввод номера телефона, который является общедоступным и ввод пароля, который является секретной информацией.

В биометрии есть два вида метода, как можно аутентифицировать человека:

1. Идентификация, ее можно сравнить с биометрической «подписью» и

используется для определения того, кому принадлежит тот или иной параметр, где идет сравнение со всеми записями по всем людям в системе, например: отпечаток пальца на месте преступления, с уже имеющейся базой данных.

2. Верификация, сравнивает полученные характеристики с уже ранее полученными данными от этого же человека, чтобы оценить принадлежность этих параметров. Например: на производственных объектах, при поступлениях на работу, отпечатки человека заносятся в базу, чтобы когда человек приходил на работу, ему требовалось только приложить палец для проверки его личности и в дальнейшем предоставлении его прав. В отличии от идентификации, подразумевается что параметры человека уже есть в системе, и она однозначно может его определить (Процент соответствия является крайне высоким).

1.2 Методы идентификации субъектов

Один из востребованных субъектов для идентификации является сам человек. Использование биометрических параметров крайне сложно, из-за сложности их получения, из-за больших энергозатрат, а также имеются проблемы с тем, как их сравнивать или находить процент похожести между параметрами.

Поэтому в основном можно видеть, как верификация в основном происходит по одному или двум параметрам.

Самый распространенный и популярным сейчас методом является машинное зрение [22]. Оно сейчас применяется довольно широко, для распознавания лиц и в системах видео наблюдений. Хоть и много уже есть готовых продуктов, наработок и исследований, сейчас до сих пор ведутся новые исследования и разработки в этой области [24][25], многие существующие системы имеют ряд проблем с связанные с [21][23]:

1. Тяжесть алгоритма: многие системы построены на базе сложных алгоритмов, из-за чего появляется необходимость в более совершенных

устройствах для обработки и анализа данных.

2. Многопоточность: многие системы не умеют работать для нахождения нескольких субъектов в кадре или в потоке.

3. Стойкость: многие системы начинают значительно хуже работать уже при минимальных помехах.

4. Точность: это зависит как от самого алгоритма, так и от количества входных параметров для анализа субъекта.

Рассмотрим основные методы распознавания, предложенные Л. Н. Чабаном на примере автоматизированном дешифрировании данных дистанционного зондирования [27]:

1. Методы распознавания, основанные на принципе кластеризации.

2. Статистическая классификация.

3. Логические методы в распознавании образов. Алгоритм вычисления оценок.

4. Применение нейронных сетей в распознавании образов.

С точки зрения распознавания, любой метод можно представить как абстрактную систему R ,состоящую из трех множеств:

$$R=\langle A,S,P\rangle \ . \qquad (1)$$

Где:

— A — набор выделяемых классов;

— S — перечень признаков, с помощью которых можно отнести образ к тому или иному классу из множества A ;

— P — набор правил с помощью которых можно принять выбор для соотнесения образа к конкретному классу.

A и образу S ют информационную компоненту системы распознавания и тесно связаны. От того как они описаны, может применяться тот или иной метод распознавания [28][29], например:

1. Принцип сравнения с эталоном — класс описывается одним или

несколькими эталонными образами.

2. Принцип кластеризации — класс описывается набором ограничений признаков.

3. Принцип общности свойств — Соответствует способу задания множеств порождающей процедурой, которая и определяет в итоге свойства образов – элементов данного множества.

Методической компонентой системы распознавания является правила принятия решения. Выделяют три основных группы [28]: эвристические методы, математические методы и лингвистические методы.

Основным критерием выбора того или иного правила является вероятность ошибочных решений. Так если существует для конкретного метода свое правило, то выбор будет в сторону того на каких свойствах класса будет наименьшее количество ошибочных решений.

Эвристические правила основываются на априорных предположений аналитиком данных, где он определяет свойства классов. Только путем выполнения численных экспериментов можно узнать, на сколько качественно были представлены априорные предположения. Также сюда можно соотнести логические методы распознавания, так как эксперт которые формирует свойства класса может основываться на логических переменных, которые он ввел. В основном применяются в качестве тестов для нахождения основных случаев погрешностей, чтобы примерно предполагать, какие значения могут быть при использовании других методах.

Математические правила основываются на математических методах, которые минимизируют погрешность к минимуму. Основная проблема связана с тем, что не все образы могут подходить под конкретный метод, и при усложнении образа, зачастую нужно перерабатывать методы, делая их работоспособным. В основном методы бывают детерминированные и статические. В основном применяются когда образ описан набором параметров, взятые как результат измерений или численного моделирования.

Синтаксические правила основывается на порождающей функции, где применяется принцип общность свойств. Чтобы определить, принадлежность образа к классу, требуется его разобрать по правилам порождающей функцией. Основная погрешность возникает из-за не корректности описания классов а также от свойств анализируемых данных. Применяется когда образ представляется в виде сложной конфигурации, записанным атомарными элементами, у которых имеются определенные связи. Где эти связи могут быть описаны с помощью различных символов-дескрипторов.

1.3 Основные методы построения систем распознавания

Системой распознавания в основном называют интерактивной программно-техническим комплексом. Системы делятся в основном по уровню сложности на две категории [27]:

1. Простые - у этих систем выделяется один набор признаков, а также для обработки используется один метод. В основном такие системы можно встретить в мобильных устройствах либо в устройствах не обладающих высокими вычислительными характеристиками. Например, активно используется для идентификации личности человека по отпечатку пальца или лицу человека, где уже качество идентификации увеличивается не методом, а уже качеством сенсора отпечатка пальцев или видеокамеры устройства.
2. Сложные — для этих систем необходим предварительный анализ всей объектной области об изучаемых субъектах. Где в основном используется совокупность методов, также зачастую используется множество наборов признаков.

Где для сложной системы необходимо предварительно:

— собрать все характеристики объектов;

— оценить их точность;

— сделать полный набор используемых признаков объекта;

— определить характеристики в тематические категории;

— изучить поведения новых тематических категорий;

— формирование наборов признаков;

— определение перечня классов;

— улучшить набор признаков с введенными новыми классами;

— разработать общую схему решения задачи.

Где если задачу невозможно решить каким методом, то необходимо сделать декомпозицию системы, то есть разбить задачу на группу задач, где каждую задачу будет выполнять какой либо метод. Или наоборот когда не хватает признаков, то нужно взять более крупные признаки и совместно уже с ними использовать существующие наборы признаков.

Очень важно иметь возможность оптимизировать наборы признаков после того как они были сформированы:

— добавлять признаки;

— убавлять признаки;

— совмещать признаки;

— дробить признаки;

— искать иные возможности их получения из среды.

Ведь очень важно от будущей системы получать максимальное производительность, поэтому есть понятия эффективности системы [30]:

— стоимость разработки системы;

— временные затраты на его получение;

— качество полученного результата распознавания;

— затраты на обучение операторов – аналитиков данных.

Качество системы можно легко описать двумя характеристиками:

1. Достоверность — процентное соотношение количества правильно найденных классов в их тематических группах.

2. Точность — процентное соотношение количества правильно найденных классов из потока данных.

Как правило 100 процентов точности на практике никогда не достигает. Допустимой цифрой как правило является от 70 процентов и выше. Так же часто применяются пост обработки данных различными процедурами для улучшения результата.

Сам алгоритм который обрабатывает данные называется зачастую «Анализ образов». Собственно в который и входят различные методы по обработкам и нахождения классов. В зависимости от используемого метода его можно дробить на под группы, например: «Поиск, коррекция», «Поиск, фильтр, коррекция» или «Прогноз, коррекция».

Сами процедуры классификации можно поделить на два вида:

1. Классификация без обучения — применяется в основном когда на потоке данных подразумеваются только те классы которые заранее известны.
2. Классификация с обучением — применяется если на потоке данных не всегда могут быть заранее известные классы.

На данный момент множеств систем построены на базе нейронных сетей, это позволяет создавать системы быстрее, но при наличии обучающей выборки. В этом и основывается основная их проблема, что без дополнительного обучения на помехи, они сильно начинают терять точность. Поэтому как правило разрабатываются отдельные системы пред обработок для приведения данных к более чистому виду. Из за простоты разработок нейронных сетей, многие перестают рассматривать иные варианты. В качестве альтернативы будет рассмотрен алгоритм «стохастические системы с детерминированной структурой» или «системы со случайной скачкообразной структурой», который можно будет использовать в дальнейшем совместно с нейронными сетями или отдельно от них. Так как они широко используются в других областях науки. Самый распространенный пример это «Байесовская фильтрация спама», метод который основанный на наивном Байесовском классификаторе, для нахождения спама. Их и будем рассматривать за основу идентификации субъектов на основе теории систем со случайной структурой.

Вывод к главе 1

1. В современных публикациях по теме исследования, во многих случаях используются нейронные сети, где их использование в большинстве случаев обуславливается простотой разработки. Но на практике такие системы требуют больших входных данных для обучения, чтобы нейронная сеть начала находить с большим процентом соответствии.

2. Зачастую методы созданные на базе алгоритмов, показывают лучшую гибкость, за счет того, что легко можно модифицировать сам алгоритм без дополнительного обучения. Где из примера с Байесовской фильтрации спама, метод, который был придуман в 1996 году, используется до сих пор из-за легкости расчета и хорошего качества нахождений.

3. Во многих системах производится поверхностный анализ доступных данных, из-за чего зачастую используется не полный комплекс сведений об объекте.

4. В мире уже давно разработана методология создания и деления алгоритмов, и того с какими блоками данных они должны работать. Что сильно упрощает в реализации понимании того, что и как нужно использовать.

Глава 2 Разработка математической модели

2.1 Биометрические идентификаторы

Первое что необходимо сделать для построение системы это выделить все характеристики изучаемых объектов, то есть биометрические идентификации, где биометрических идентификаторов существует бесчисленное множество, но для сравнения будут взяты только те идентификаторы, которые используются в какой либо области:

1. Возраст: одна из базовых характеристик человека, которая описывается во многих медицинских документах, как правило имеет слабую динамичность благодаря чему, можно использовать, но с большей степенью погрешности в вычислениях [15].

2. Группа крови: каждый человек имеет какую либо группу крови, поэтому это хороший признак, чтобы поделить на N-ое количество групп людей [19].

3. Диагностика подписи: у разных людей она разная, где при написании подписи учитывается не только сам рисунок, но и как он был нанесен [10].

4. Жесты рук: в области машинного зрения является одной из перспективных задач, так как показывает хороший процент соответствия, а также позволяет получать уникальные данные, такие как строение кисти [8].

5. Клавиатурный почерк: удобный способ идентификации людей работающих около компьютера [12].

6. Лицо: сейчас очень много исследований и уже работающих на практике систем, которые работают на основе распознавании лиц [6].

7. Масса тела: одна из базовых характеристик человека, которая описывается во многих медицинских документах, как правило имеет слабую динамичность благодаря чему, можно использовать, но с большей степенью погрешности в вычислениях [15].

8. Отпечатки пальцев: этот признак активно используется уже в мире, уже с давних времен [7].

9. Пирсинг: используются в полиции как опознавательный параметр преступников [18].

10. Пол: характеристика, которая делит выборку почти 50 на 50.

11. Радужная оболочка: у многих людей можно увидеть разный тон цвета глаз, а также рисунок оболочки [9].

12. Размер шага: размер шага используется в системах определения преступников, так как средняя скорость шага, как правила отличается от скорости шага обычного человека [20].

13. Раса: хоть он не имеет хорошего разброса среди одной местности, но будет полезен в местах, где идет большое скопление туристов [16].

14. Распознавание голоса: по этому признаку можно идентифицировать людей, он уже применяется во многих системах [11].

15. Рост: одна из базовых характеристик человека, которая описывается во многих медицинских документах, как правило имеет слабую динамичность благодаря чему, можно использовать, но с большей степенью погрешности в вычислениях [15].

16. Скорость бега: скорость бега используется в системах определения преступников, так как средняя скорость шага, как правила отличается от скорости шага обычного человека [20].

17. Скорость шага: скорость шага используется в системах определения преступников, так как средняя скорость шага, как правила отличается от скорости шага обычного человека [20].

18. Сосуды ладони: появилось относительно недавно в системах контроля и управления доступом (СКУД) [13].

19. Стиль письма: у каждого человека есть свой стиль письма, часто используется для удостоверения достоверности авторства [17].

20. Татуировки: используются в полиции как опознавательный параметр

преступников [18].

21. Термография лица: широко применяется в методах диагностики заболеваний и динамического контроля в процессе лечения пациента [14].

22. Уши: этот признак сейчас активно рассматривается в азиатских странах и имеет различные исследования на тему того как правильно считывать данные, а также на что обращать внимания [5].

23. Хромосома: очень часто используются для идентификации личности в криминологии [26].

24. Цвет волос: одна из базовых характеристик человека, которая описывается во многих медицинских документах, как правило имеет слабую динамичность благодаря чему, можно использовать, но с большей степенью погрешности в вычислениях [15].

25. Цвет кожи: хоть он не имеет хорошего разброса среди одной местности, но будет полезен в местах, где идет большое скопление туристов [16].

26. Наличие всех частей тела: является не уникальным признаком, но при отклонении от нормы начинает формировать малые группы людей.

Все эти биометрические идентификаторы были рассмотрены с разных точек зрения, чтобы можно было их оценить, будем использовать свойства описанные Кларком [3]:

1. Всеобщность: каждый человек должен обладать этим биометрическим идентификатором.

2. Уникальность: с точки зрения биометрии нет двух людей, обладающих одинаковыми биометрическими характеристиками (Подразумевается не только количественный признак, но и качественный, например: цвет глаз не зеленый, а #00ff00 по RGB или (0.6,0,1,0) по CMYK).

3. Постоянство: биометрический идентификатором не должен со временем меняться.

4. Измеримость: биометрический идентификатором должен иметь

возможность измеряться каким либо устройством.

5. Эффективность: возможность идентификации, скорость, гибкость, потребность в ресурсах для обеспечения желаемой точности и скорости идентификации, а также факторы, возникающие в процессе идентификации и внешние факторы, влияющие на точность идентификации и скорость.

6. Приемлемость: люди не должны в целом возражать от сбора или измерения биометрического параметра.

7. Защищенность от подделки: отражает защищенность системы от обмана.

Комбинация всех этих свойств определяет эффективность биометрии и следовательно, эффективность систем аутентификации [4].

Применим это к нашим биометрическим идентификаторам в таблице 1.

Таблица 1 — Сравнение биометрических идентификаторов по базовым требованиям

Биометрия	Всеобщность	Уникальность	Постоянство	Измеримость	Эффективность	Приемлемость	Защищенность
Возраст	В	Н	В	В	Н	С	Н
Группа крови	В	Н	В	В	Н	С	В
Диагностика подписи	В	С	Н	В	Н	В	Н
Жесты рук	С	С	С	В	С	С	С
Клавиатурный почерк	Н	Н	Н	С	Н	С	С
Лицо	В	Н	С	В	Н	В	Н
Масса тела	В	Н	С	В	В	С	Н

Продолжение таблицы 1

Биометрия	Всеобщность	Уникальность	Постоянство	Измеримость	Эффективность	Приемлемость	Защищенность
Отпечатки пальцев	С	В	В	С	В	С	В
Пирсинг	Н	С	Н	С	С	С	Н
Пол	В	Н	В	В	Н	Н	В
Радужная оболочка	В	В	В	С	В	Н	В
Размер шага	С	С	С	В	С	В	Н
Раса	В	Н	В	В	С	Н	В
Распознавание голоса	В	С	Н	С	Н	В	Н
Рост	В	Н	В	В	В	В	Н
Скорость бега	С	С	С	В	Н	В	Н
Скорость шага	С	С	С	В	С	В	Н
Сосуды ладони	С	С	С	С	С	С	В
Стиль письма	В	В	С	С	Н	С	С
Татуировки	Н	С	Н	С	С	С	С
Термография лица	В	В	Н	В	С	В	В
Уши	В	С	С	В	В	С	С
Хромосомы	В	В	В	С	Н	С	В
Цвет волос	В	Н	Н	С	С	С	Н
Цвет кожи	В	Н	С	С	В	Н	Н

Продолжение таблицы 1

Биометрия	Всеобщность	Уникальность	Постоянство	Измеримость	Эффективность	Приемлемость	Защищенность
Части тела	В	Н	В	С	Н	Н	В

Где, Н — низкая, С — средняя, В — высокая. Для дальнейших расчетов оценим эти параметры как 1, 2 и 3 соответственно.

Так же зачастую частые требований к разработке систем связанных с биометрическими методами:

1. Стоимость: оценка стоимости на каждого пользователя оборудованием или программным обеспечением.

2. Удобство использования: сочетание длительности проведения идентификации и простоты использования средств идентификации.

3. Эксплуатационные характеристики: оборудование или программное обеспечение должно работать бесперебойно на длительном промежутке времени.

Применим это к нашим биометрическим идентификаторам в таблице 2.

Таблица 2 — Сравнение биометрических идентификаторов по общим требованиям

Биометрия	Цена	Удобство использования	Эксплуатация
Возраст	Средняя	Низкая	Низкая
Группа крови	Высокая	Низкая	Низкая
Диагностика подписи	Средняя	Средняя	Средняя
Жесты рук	Высокая	Высокая	Высокая

Продолжение таблицы 2

Биометрия	Цена	Удобство использования	Эксплуатация
Клавиатурный почерк	Низкая	Средняя	Низкая
Лицо	Средняя	Высокая	Средняя
Масса тела	Низкая	Высокая	Средняя
Отпечатки пальцев	Средняя	Высокая	Высокая
Пирсинг	Низкая	Средняя	Средняя
Пол	Высокая	Низкая	Средняя
Радужная оболочка	Высокая	Высокая	Средняя
Размер шага	Низкая	Средняя	Высокая
Раса	Средняя	Низкая	Низкая
Распознавание голоса	Низкая	Высокая	Высокая
Рост	Низкая	Средняя	Низкая
Скорость бега	Низкая	Средняя	Высокая
Скорость шага	Низкая	Средняя	Высокая
Сосуды ладони	Средняя	Высокая	Средняя
Стиль письма	Высокая	Средняя	Низкая
Татуировки	Низкая	Средняя	Средняя
Термография лица	Средняя	Высокая	Высокая
Уши	Средняя	Высокая	Высокая
Хромосомы	Высокая	Низкая	Высокая
Цвет волос	Низкая	Средняя	Средняя

Продолжение таблицы 2

Биометрия	Цена	Удобство использования	Эксплуатация
Цвет кожи	Низкая	Высокая	Средняя
Части тела	Высокая	Низкая	Низкая

Где, Н — низкая, С — средняя, В — высокая. Для дальнейших расчетов оценим эти параметры как 1, 2 и 3 соответственно.

Еще важным фактором при разработке биометрического метода учитывается среда использования, так как от этого зависит распространение метода:

1. Дом или Офис (в здании): чтобы пройти идентификацию не требуется сложного оборудования.

2. Мобильный: подразумевается, что можно пройти идентификацию использую технические средства мобильного телефона или переносного технического устройства.

3. Общественное место (на улице): это может быть как установленный прибор стоящий на улице, так и что-то мобильное.

Опишем биометрические идентификаторы по следующей шкале:

1. Нет — невозможно провести идентификацию биометрического признака, необходимо дополнительное исследование в другом месте. Например в лаборатории.

2. Неудобно — биометрический признак можно идентифицировать, но возможно с худшими показаниями качества.

3. Да — биометрический признак удобно идентифицируется.

Где, для дальнейших расчетов оценим эти параметры как 1, 2 и 3 соответственно.

Применим это к нашим биометрическим идентификаторам в таблице 3.

Таблица 3 — Сравнение биометрических идентификаторов по удобству

Биометрия	В здании	Мобильный	На улице
Возраст	Нет	Нет	Нет
Группа крови	Нет	Нет	Неудобно
Диагностика подписи	Да	Неудобно	Неудобно
Жесты рук	Да	Да	Да
Клавиатурный почерк	Да	Нет	Нет
Лицо	Да	Да	Да
Масса тела	Да	Нет	Да
Отпечатки пальцев	Да	Да	Да
Пирсинг	Да	Нет	Неудобно
Пол	Да	Нет	Нет
Радужная оболочка	Да	Нет	Да
Размер шага	Неудобно	Да	Да
Раса	Нет	Нет	Нет
Распознавание голоса	Да	Да	Неудобно
Рост	Да	Нет	Да
Скорость бега	Неудобно	Да	Да
Скорость шага	Неудобно	Да	Да
Сосуды ладони	Да	Нет	Неудобно
Стиль письма	Да	Неудобно	Неудобно
Татуировки	Да	Нет	Неудобно
Термография лица	Да	Нет	Неудобно

Продолжение таблицы 3

Биометрия	В здании	Мобильный	На улице
Уши	Да	Нет	Неудобно
Хромосомы	Нет	Нет	Нет
Цвет волос	Да	Да	Да
Цвет кожи	Да	Нет	Да
Части тела	Да	Да	Да

Бывают также среды по частоте использованию, так например в банках идентификация людей происходит часто, поэтому требуется быстрой работы от системы. Опишем биометрические идентификаторы по следующей шкале:

1. Метод мало практичен — невозможно провести идентификацию биометрического признака на месте, необходимо дополнительное исследование в другом месте. Например в лаборатории.

2. Средняя — биометрический признак можно идентифицировать на месте, но с худшими показаниями точности.

3. Хорошая — биометрический признак хорошо идентифицируется на месте.

Где, для дальнейших расчетов оценим эти параметры как 1, 2 и 3 соответственно.

Применим это к нашим биометрическим идентификаторам в таблице 4.

Таблица 4 — Сравнение биометрических идентификаторов по возможности множественной проверки

Биометрия	Пригодность для многократной идентификаций
Возраст	Средняя
Группа крови	Метод мало практичен

Продолжение таблицы 4

Биометрия	Пригодность для многократной идентификаций
Диагностика подписи	Средняя
Жесты рук	Хорошая
Клавиатурный почерк	Метод мало практичен
Лицо	Хорошая
Масса тела	Средняя
Отпечатки пальцев	Хорошая
Пирсинг	Средняя
Пол	Средняя
Радужная оболочка	Хорошая
Размер шага	Средняя
Раса	Средняя
Распознавание голоса	Хорошая
Рост	Средняя
Скорость бега	Метод мало практичен
Скорость шага	Средняя
Сосуды ладони	Хорошая
Стиль письма	Метод мало практичен
Татуировки	Средняя
Термография лица	Хорошая
Уши	Средняя
Хромосомы	Метод мало практичен

Продолжение таблицы 4

Биометрия	Пригодность для многократной идентификаций
Цвет волос	Средняя
Цвет кожи	Средняя
Части тела	Метод мало практичен

Рассмотрев и оценив все биометрические параметры, можно сделать вывод, что у большинства биометрических идентификаторов есть проблемы с мобильностью, а также с низкой уникальностью. Из-за чего использование этих признаков по раздельностью будет давать малый процент уникальности. В дальнейшем они будут рассмотрены для совмещения идентификаторов.

2.2 Нахождение оптимальных биометрических идентификаторов

Рассмотрев и оценив все биометрические параметры из параграфа 2.1, можно составить формулу оценки и оценить все выше описанные биометрические параметры. Введем несколько новых параметров, с помощью которых можно будет калибровать результат под разные потребности:

1. k_c — потребность в дешевизне. Чем больше, тем дешевле требуется решение.
2. k_q — потребность в качестве. Чем больше, тем точнее требуется решение.
3. k_s — потребность в удобстве. Чем больше, тем быстрее должен происходить идентификация субъекта.

И составим формулу (1) для определения качества признака:

$$S=\frac{\frac{(всеобщность+уник.+постоянство+эффект+защищеность)}{5}*\frac{2^{k_q}}{k_s}}{\left(цена+измеримость+\frac{(3-приемлемость)}{2}\right)^{k_c}}+$$
$$+\frac{\frac{(дом+мобил.+общ.место)}{3}*\frac{(удобство+экспл.)}{2}}{2}*массовость*\frac{2^{k_s}}{k_q}}{\left(\frac{цена+измеримость+(3-приемлемость)}{2}\right)^{k_c}} \quad (2)$$

Для рассмотрения идентификаторов возьмем следующие коэффициенты:

— $k_c=1, k_q=1, k_s=1$,

— $k_c=2, k_q=1, k_s=1$,

— $k_c=1, k_q=2, k_s=1$,

— $k_c=1, k_q=1, k_s=2$,

— $k_c=2, k_q=2, k_s=1$,

— $k_c=1, k_q=2, k_s=2$,

— $k_c=2, k_q=1, k_s=2$.

А также найдем среднее значение по всем этим входным коэффициентам, которое будем использовать дальше.

Где подставив все значения получаем следующую таблицу в рисунке 1:

№	Биометрические параметры	1,1,1	2,1,1	1,2,1	1,1,2	2,2,1	1,2,2	2,1,2	Среднее
6	Распознавание голоса	18,13	12,09	12,27	33,07	8,18	18,13	22,04	17,70
1	Отпечатки пальцев	13,04	5,22	9,88	22,72	3,95	13,04	9,09	10,99
2	Лицо	10,28	4,11	7,06	18,64	2,82	10,28	7,46	8,66
9	Термография лица	9,12	3,65	7,44	15,36	2,98	9,12	6,14	7,69
21	Скорость шага	8,47	4,23	6,93	14,23	3,47	8,47	7,12	7,56
22	Размер шага	8,47	4,23	6,93	14,23	3,47	8,47	7,12	7,56
3	Жесты рук	8,86	2,53	6,14	16,00	1,76	8,86	4,57	6,96
12	Цвет волос	7,60	3,80	6,20	12,80	3,10	7,60	6,40	6,79
8	Сосуды ладони	7,76	3,10	6,52	12,88	2,61	7,76	5,15	6,54
11	Масса тела	6,27	2,51	5,53	10,13	2,21	6,27	4,05	5,28
13	Цвет кожи	6,27	2,51	5,53	10,13	2,21	6,27	4,05	5,28
4	Радужная оболочка	6,71	1,92	5,93	10,86	1,69	6,71	3,10	5,28
10	Рост	5,70	2,85	6,15	8,10	3,08	5,70	4,05	5,09
15	Татуировки	5,60	2,80	5,20	8,80	2,60	5,60	4,40	5,00
16	Пирсинг	5,40	2,70	4,80	8,70	2,40	5,40	4,35	4,82
17	Уши	5,60	1,87	5,20	8,80	1,73	5,60	2,93	4,53
23	Скорость бега	4,93	2,47	4,87	7,47	2,43	4,93	3,73	4,40
5	Диагностика подписи	5,01	2,01	4,43	8,11	1,77	5,01	3,24	4,23
7	Клавиатурный почерк	2,45	1,23	3,03	3,10	1,51	2,45	1,55	2,19
25	Стиль письма	2,63	0,88	3,52	3,07	1,17	2,63	1,02	2,13
26	Хромосомы	2,40	0,80	3,80	2,20	1,27	2,40	0,73	1,94
18	Пол	2,35	0,59	2,83	3,05	0,71	2,35	0,76	1,80
24	Части тела	2,11	0,60	2,94	2,34	0,84	2,11	0,67	1,66
19	Раса	1,94	0,56	3,03	1,83	0,87	1,94	0,52	1,53
20	Возраст	1,87	0,62	2,73	1,93	0,91	1,87	0,64	1,51
14	Группа крови	1,64	0,47	2,70	1,39	0,77	1,64	0,40	1,29

Рисунок 1 — Таблица стоимости биометрических идентификаторов

Поделим биометрические идентификаторы на три группы:

1. Нахождение групп людей — идентификаторы, с помощью которых можно находить человека или определенную группу людей в большой массе людей. Например, людей с подозрением на терроризм в обществе.

2. Идентификация человека (удалённо) — идентификаторы, с помощью которых можно идентифицировать конкретного человека в малых группах людей, без прямого взаимодействия. Например, через камеру наблюдения на режимном объекте.

3. Идентификация человека (лично) — идентификаторы, с помощью которых можно идентифицировать конкретного человека, с помощь прямого взаимодействия на месте. Например, через датчик отпечатков пальцев.

4. Идентификация человека (частные) — идентификаторы, с помощью которых можно идентифицировать конкретного человека, без прямого взаимодействия с ним, но имея вводные данные о нем. Например, получив пробу крови или ДНК человека.

Опишем биометрические идентификаторы по следующей шкале:

1. Низкая — невозможно провести идентификацию биометрического признака.

2. Средняя — биометрический признак можно идентифицируется, но с худшими показаниями точности.

3. Высокая — биометрический признак хорошо идентифицируется на месте.

Применим это к нашим биометрическим идентификаторам в таблице 5.

Таблица 5: Группировка биометрических методов по их пригодности для разных потребностей

Биометрия	Групп людей	Идентификация (удалённо)	Идентификация (лично)	Идентификация (частные)
Возраст	С	С	С	В
Группа крови	Н	Н	С	В
Диагностика подписи	Н	Н	С	В
Жесты рук	С	С	В	В
Клавиатурный почерк	Н	С	С	В
Лицо	С	В	В	В
Масса тела	Н	Н	В	С
Отпечатки пальцев	Н	Н	В	С
Пирсинг	С	С	В	С
Пол	Н	С	С	В
Радужная оболочка	Н	Н	В	С

Продолжение таблицы 5

Биометрия	Групп людей	Идентификация (удалённо)	Идентификация (лично)	Идентификация (частные)
Размер шага	С	С	В	С
Раса	С	С	С	В
Распознавание голоса	Н	С	В	В
Рост	С	С	В	С
Скорость бега	С	С	В	С
Скорость шага	С	С	В	С
Сосуды ладони	Н	Н	В	С
Стиль письма	Н	С	В	В
Татуировки	С	С	В	С
Термография лица	С	С	В	С
Уши	Н	С	В	В
Хромосомы	Н	Н	Н	В
Цвет волос	С	С	В	В
Цвет кожи	С	С	С	В
Части тела	С	С	С	В

Для будущей модели будут взяты биометрические идентификаторы, чтобы найти конкретного человека из потока. Такой подход удобен для нахождение потенциального нарушителя в какой либо запрещенной зоне, когда известны все люди допуск у которых имеется. Следующие биометрические

идентификаторы будут взяты за основу:

— цвет волос, в качестве входных данных код поделенный на три целочисленных переменных из цветового пространства LAB;

— размер шага, рассматривается метров в секунду, как дробное число;

— рост, рассматривается в миллиметрах, как целочисленное число.

— наличие очков у человека, этот идентификатор не рассматривался ранее, но для нашей цели этот параметр будет полезен. Входные данные в виде ноль или один.

Специально были взяты биометрические идентификаторы, которые можно задать числом или его набором. Чтобы не использовать методы из методологии компьютерного зрения. Например, нахождение людей по лицам. В дальнейшем будет проверка и с другими идентификаторами.

2.3 Байесовский фильтр

Для начала требуется рассмотреть оригинальный метод Байесовской фильтрации [33][34], но для начала обозначим некоторые обозначение, которое будем использовать:

— k — дискретный момент времени;

— x_k — вектор фазовых координат объекта;

— z_k — вектор измерений фазовых координат объекта;

— ξ_k — вектор возмущений, действующих на объект;

— ζ_k — вектор помех, действующих на объект;

— $\Phi_k(\xi_k)$, $\Phi_k(\zeta_k)$ — функции распределения случайных векторов ξ_k , ζ_k .

— $f_k(x_k)$ — плотность вероятности случайного вектора x_k в момент k;

— $f_k(x_{k+1}|x_k)$ — условная плотность вероятности перехода вектора фазовых координат из состояния x_k в состояние x_{k+1};

— $f(x_k, s_k)$ — распределение вероятности вектора состояния (x_k, s_k);

— $f(x_k|s_k)$ — условная плотность вероятности вектора x_k при фиксированном s_k;

— s_k — вектор состояния структуры, это вектор содержащий информацию о том в каком режиме находится структура, например найден нужный объект;

— r_k — вектор индикации структуры, это может быть например состояние времени дня: вечер или день, или идут занятия или нет;

— $\bar{s}_k$— оптимальная оценка состояния структуры s_k.

— $\eta_k = [x_k, s_k, z_k, r_k]$ — является набором векторов состояний и фазовых изменений.

Сами характеристики об объекте передаются в x_k и выглядят следующим образом:

$$x_{k+1} = \varphi(s_{k+1}, x_k, s_k, \xi_k) \tag{3}$$

Информация о том на сколько он ему соответствует при некоторых характеристиках объектах и состоянии структуры:

$$q_k = \varphi(s_{k+1}, x_k, s_k) \quad , \quad s_k = \overline{1, n^{(s)}} \tag{4}$$

Где система подразумевает, что в качестве данных на каждый момент времени, будет приниматься изменения состояния системы, а также изменение фазовых координат:

$$z_{k+1} = \psi(x_{k+1}, s_{k+1}, r_{k+1}, x_k, s_k, z_k, r_k, \zeta_k) \tag{5}$$

$$\pi_{k+1}(r_{k+1}|x_{k+1}, s_{k+1}, x_k, s_k, z_k, r_k) \quad , \quad r_k = \overline{1, n^{(r)}} \tag{6}$$

Где ζ_k и ξ_k являются независящими от времени, а $q_k(.)$ и $\pi_{k+1}(.)$ заданными условными вероятностями перехода условно-марковских цепей.

Также известны: $\varphi_k(.)$ и $\psi(.)$ — это векторы детерминированных функций времени и случайных аргументов.

Для начала необходимо найти апостериорное распределение вектора

состояния на интервале $[0,k]$. Для этого основываясь по формуле Байеса запишем следующее уравнений для поиска плотности вероятности состояний:

$$f_k(x_{k+1}, s_{k+1}|z_0..z_{k+1}, r_0..r_{r+1}) = \\ = \frac{f_k(z_{k+1}, r_{k+1}|x_{k+1}, s_{k+1}, z_0..z_k, r_0..r_r) f_k(x_{k+1}, s_{k+1}|z_0..z_k, r_0..r_r)}{f_k(z_{k+1}, r_{k+1}|z_0..z_k, r_0..r_k)} \quad (7)$$

За счет использования свойств марковских векторов $[x_k, s_k]$ и $\eta_k = [x_k, s_k, z_k, r_k]$ применим формулу (7) к формулам (3)-(6), чтобы получить рекуррентный алгоритм:

$$\hat{f}_k(x_{k+1}, s_{k+1}) = \sum_{s_k} \int_{-\infty}^{+\infty} \frac{\varepsilon^*(\eta_{k+1}, \eta_k)}{\overline{f}_k(z_{k+1}, r_{k+1})} dx_k. \quad (8)$$

Где символом «$\hat{\square}$» обозначаем апостериорные характеристики вектора значений. А с нормированным коэффициентом мы получаем следующее:

$$\overline{f}_k(z_{k+1}, r_{k+1}) = \sum_{s_{k+1}} \sum_{s_k} \iint_{-\infty}^{+\infty} \varepsilon(\eta_{k+1}, \eta_k) dx_k dx_{k+1}, \quad (9)$$

где

$$\varepsilon^*(\eta_{k+1}, \eta_k) = f_k(z_{k+1}|x_{k+1}, s_{k+1}, r_{k+1}, \eta_k) \times \\ \times \pi(r_{k+1}|x_{k+1} s_{k+1}, \eta_k) f_k(x_{k+1}|s_{k+1}, x_k, s_k) q(s_{k+1}|x_k, s_k) \hat{f}_k(x_k, s_k) \quad (10)$$

и с начальными условиями

$$\hat{f}_k(x_0, s_0) = f_k(x_0, s_0) \quad (11)$$

то

$$\hat{f}_k(x_{k+1}, s_{k+1}) = f_k(x_{k+1}, s_{k+1}|z_0 \ldots z_{k+1}, r_0 \ldots r_{k+1}) \\ \overline{f}_k(z_{k+1}, r_{k+1}) = f_k(z_{k+1}, r_{k+1}|z_0 \ldots z_k, r_0 \ldots r_k) \quad (12)$$

Где символом «$\overline{\square}$» обозначаем математическое ожидание вектора. Опишем $f_k(..)$ из ε^* в следующих уравнениях:

$$f_k(x_{k+1}|s_{k+1}, x_k, s_k) = \\ = (2\pi)^{-n_x} \iint_{-\infty}^{\infty} \exp\{i\omega^T[\varphi(s_{k+1}, x_k, s_k, \xi_k) - x_{k+1}]\} d\Phi(\xi_k) d\omega \quad (13)$$

$$f_k(z_{k+1}|x_{k+1}, s_{k+1}, \eta_k)=$$
$$=(2\pi)^{-n_z}\iint_{-\infty}^{\infty}\exp\{i\omega^T[\psi(x_{k+1}, s_{k+1}, r_{k+1}, \eta_k, \zeta_k)-z_{k+1}]\}d\Phi(\zeta|\xi_k)d\omega \quad (14)$$

Где можно описать функцию Φ следующими уравнениями:

$$\Phi(\zeta_k|\xi_k)=\Phi(\zeta_k, \xi_k)\Phi^{-1}(\xi_k), \Phi(\xi_k)=[\Phi(\xi_k, \zeta_k)]_{\xi_k}=\infty \quad (15)$$

После того как получили все ключевые уравнения можно теперь представить метод следующим образом на рисунке 2.

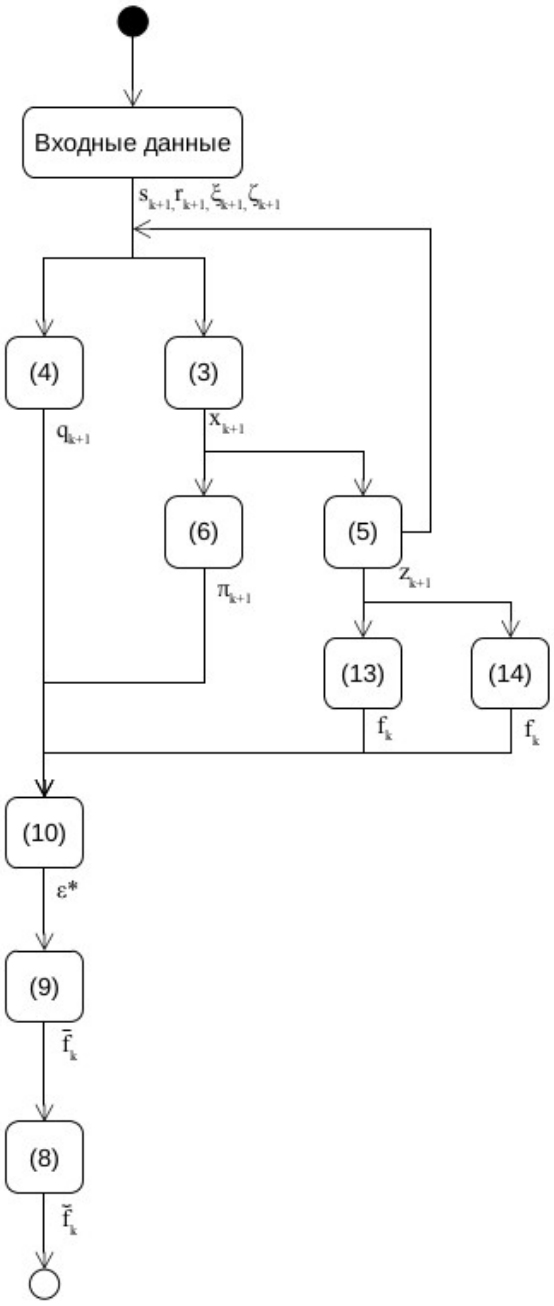

Рисунок 2 — Схема Байесовского фильтра

Исходя из рисунка 2 можно заметить следующее:

1. Сам метод не работает в режиме прогнозировании данных, за счет того что скачкообразное возмущения и помехи приводятся к независимым переменным от времени, а также что на каждый момент времени требуется состояние системы.

2. Результатом метода на каждом момент времени является апостериорная плотность вероятности вектора x_{k+1} .

3. Система подразумевает, что ей требуется для работы нахождение вектора возмущений, действующих на объект, а также нахождение вектора помех, действующих на объект. Что усложняет для работы в практическом применении.

Исходя из этого, можно сделать вывод о том, что нужно поменять в нем, чтобы он подходил для наших целей:

1. Необходимо избавиться от вектора возмущений, действующих на объект, а также от вектора помех, действующих на объект.

2. Свести систему для нахождения не только апостериорной плотности вероятности вектора, но и еще для нахождения его состояния. Благодаря чему можно будет делать не только соотношение того на сколько верно текущее состояние системы, но и предположения того, в каком состоянии системы находится на данный момент времени система.

2.4 Алгоритм идентификации субъектов на базе Байесовского фильтра

Теперь перейдем к самому алгоритму. Для решения нужно применить метод двухмоментной параметрической аппроксимации, который состоит в аппроксимации неизвестных плотностей вероятностей. За основу возьмем алгоритм «классификатор–идентификатор–фильтр–дисперсиометр», где доработаем слабые части, связанные с высчитыванием помех, а также

интегрировании на всей прямой. Основываясь на предыдущем выводе, сделаем следующее:

1. Уменьшить количество входных переменных упростив и получаться непосредственно данные о наблюдаемом объекте.

2. Система должна будет фильтровать на наличие помех или делать поправки относительно их.

3. Нужно находить прогнозируемые данные, чтобы при наличии соответствия с новыми апостериорными значениями, соответствия класса к объекту подкреплялось.

Поэтому в требуемой системе нужно чтобы находились следующие значения:

— апостериорные значения математического ожидания вектора x_{k+1} ;

— апостериорные значения ковариации вектора x_{k+1} ;

— апостериорные значения плотности вероятности вектора x_{k+1} ;

— прогнозируемые значения математического ожидания вектора x_{k+1} ;

— прогнозируемые значения ковариации вектора x_{k+1} ;

— прогнозируемые значения плотности вероятности вектора x_{k+1} ;

— оптимальная апостериорная оценка состояния структуры;

С помощью них можно организовать минимальную структуру, которая сможет получать и обрабатывать поток значений.

За основу будет взята замкнутая система, которая на каждый момент времени будет выдавать оптимальную апостериорную оценку состояния структуры. Она непосредственно будет показывать найден ли искомый объект или нет. Из явных недостатков этой системы, является понижении точности, так как в отличие от Байесовского фильтра, зависит от значений предыдущего момента времени. И точность определения объекта, обуславливается длительностью его нахождения в потоке данных. Метод описан следующим образом на рисунке 3.

Рисунок 3 — Схема алгоритма классификатор–идентификатор–фильтр–дисперсиометр

Для начала введем новые обозначим, которое мы будем использовать:

— $\hat{x}(s_{k+1})$ — функция которая находит апостериорное математическое ожидание вектора x_{k+1} ;

— $\hat{R}(s_{k+1})$ — функция которая находит апостериорную ковариацию вектора x_{k+1} ;

— $\tilde{p}(s_{k+1})$ — функция которая прогнозируемая на один шаг плотность вероятности вектора x_{k+1} ;

— $\tilde{x}(s_{k+1})$ — функция которая прогнозируемая на один шаг дискретности вперед оценка вектора x_{k+1} ;

— $\tilde{R}(s_{k+1})$ — функция которая прогнозируемая на один шаг вперед оценка ковариацию вектора R_{k+1} ;

— $\hat{s}_k$ — оптимальная апостериорная оценка состояния структуры;

— $\hat{x}_k$ — апостериорное математическое ожидание вектора x_{k+1} при фиксированном $z_0..z_k$;

— $\hat{R}_k$ — апостериорная ковариация вектора x_{k+1} при фиксированном $z_0..z_k$.

Учитывая связь $\hat{p}(s_k)$, $\hat{x}(s_k)$, $\hat{R}(s_k)$ с $\hat{f}(x_k|s_k)$, получаем систему обыкновенных рекуррентных уравнений:

$$\hat{p}(s_{k+1})=\bar{f}^{-1}(z_{k+1},r_{k+1})\sum_{s_k}\hat{p}(s_k)\iint_{-\infty}^{+\infty}\varepsilon(\eta_{k+1},\eta_k)\,dx_k\,dx_{k+1} \tag{16}$$

$$\hat{x}(s_{k+1})=\left[\bar{f}(z_{k+1},r_{k+1})\hat{p}(s_{k+1})\right]^{-1}\sum_{s_k}\hat{p}(s_k)\iint_{-\infty}^{+\infty}x_{k+1}\varepsilon(\eta_{k+1},\eta_k)\,dx_k\,dx_{k+1} \tag{17}$$

$$\hat{R}(s_{k+1})=\left[\bar{f}(z_{k+1},r_{k+1})\hat{p}(s_{k+1})\right]^{-1}\sum_{s_k}\hat{p}(s_k)\iint_{-\infty}^{+\infty}\hat{x}^0(s_{k+1})\hat{x}^{0T}\varepsilon(\eta_{k+1},\eta_k)\,dx_k\,dx_{k+1} \tag{18}$$

Где

$$\begin{gathered}\varepsilon(\eta_{k+1},\eta_k)=f(z_{k+1}|x_{k+1},s_{k+1},r_{k+1},\eta_{k+1})\pi(r_{k+1}|s_{k+1},\eta_k)\times\\ \times f(x_{k+1}|s_{k+1},x_k,s_k)q(s_{k+1}|x_k,s_k)\hat{f}(x_k|s_k)\\ s_{k+1}=\overline{1,n^{|s|}},\ k=0,1,2,\ldots\end{gathered}, \tag{19}$$

С нормировочным коэффициентом

$$\hat{f}(z_{k+1},r_{k+1})=\sum_{s_k+1}\sum_{s_k}\hat{p}(s_k)\iint_{-\infty}^{+\infty}x_{k+1}\varepsilon(\eta_{k+1},\eta_k)\,dx_k\,dx_{k+1} \tag{20}$$

и начальными условиями

$$\hat{p}(s_0)=p(s_0)=\int_{-\infty}^{+\infty} f(x_0,s_0)\,dx_0,$$

$$\hat{x}(s_0)=\bar{x}(s_0)=\int_{-\infty}^{+\infty} x_0 f(x_0|s_0)\,dx_0,$$

$$\hat{R}(s_0)=R(s_0)=\int_{-\infty}^{+\infty} \bar{x}^0(s_0)\,x^{\bar{0}T}(s_0)\,f(x_0|s_0)\,dx_0, \tag{21}$$

$$f(x_0|s_0)=p^{-1}(s_0)\,f(x_0,s_0).$$

Приближенно-оптимальные оценки $\hat{s}_k$, $\hat{x}_k$ и $\hat{R}_k$ определяются по формулам:

$$\hat{s}_k=arg\,max_{s_k}\,\hat{p}(s_k) \tag{22}$$

$$\hat{x}_k=\sum_{s_k}\hat{x}_k(s_k)\,\hat{p}(s_k) \tag{23}$$

$$\hat{R}_k=\sum_{s_k}\left[\hat{R}(s_k)+\hat{x}(s_k)\,\hat{x}^T(s_k)\right]\hat{p}(s_k)-\hat{x}_k\,\hat{x}_k^T \tag{24}$$

Где мы получаем следующее:

— (16) — «классификатор»;

— (22) — «идентификатор»;

— (17), (23) — «фильтр»;

— (18), (24) — «дисперсиометр».

Мы будем рассматривать систему «наблюдение без опаздывания», он является аналогом приближенно-оптимальный алгоритму, который основанн на методе двухмоментном параметрической аппроксимации, примененном к уравнениям:

$$\hat{f}(x_{k+1},s_{k+1})=\frac{\beta^*(\eta_{k+1})}{\bar{f}(z_{k+1},r_{k+1})} \tag{25}$$

$$\tilde{f}(x_{k+1},s_{k+1})=\sum_{s_k}\int_{-\infty}^{+\infty} f(x_{k+1}\vee s_{k+1},x_k,s_k,z_k)\,\gamma^*(s_{k+1},x_k,s_k)\,dx_k \tag{26}$$

$$s_{k+1}=\overline{1,n^{(s)}},\ k=0,1,2,\ldots$$

имеет следующий вид:

$$\hat{p}(s_{k+1})=\left[\bar{f}(z_{k+1},r_{k+1})\right]^{-1}\tilde{p}(s_{k+1})\int_{-\infty}^{+\infty}\beta(\eta_{k+1})dx_{k+1} \tag{27}$$

$$\begin{aligned}\hat{x}(s_{k+1})&=\left[\bar{f}(z_{k+1},r_{k+1})\hat{p}(s_{k+1})\right]^{-1}\tilde{p}(s_{k+1})\int_{-\infty}^{+\infty}x_{k+1}\beta(\eta_{k+1})dx_{k+1}=\\&=\left[\int_{-\infty}^{+\infty}\beta(\eta_{k+1})dx_{k+1}\right]^{-1}\int_{-\infty}^{+\infty}x_{k+1}\beta(\eta_{k+1})dx_{k+1}\end{aligned} \tag{28}$$

$$\begin{aligned}\hat{R}(s_{k+1})&=\left[\bar{f}(z_{k+1},r_{k+1})\hat{p}(s_{k+1})\right]^{-1}\tilde{p}(s_{k+1})\int_{-\infty}^{+\infty}\hat{x}^0(s_{k+1})\hat{x}^{0T}(s_{k+1})\beta(\eta_{k+1})dx_{k+1}=\\&=\left[\int_{-\infty}^{+\infty}\beta(\eta_{k+1})dx_{k+1}\right]^{-1}\int_{-\infty}^{+\infty}\hat{x}^0(s_{k+1})\hat{x}^{0T}(s_{k+1})\beta(\eta_{k+1})dx_{k+1}\end{aligned} \tag{29}$$

Где

$$\bar{f}(z_{k+1},r_{k+1})=\sum_{s_k}\tilde{p}(s_{k+1})\int_{-\infty}^{+\infty}\beta(\eta_{k+1})dx_{k+1} \tag{30}$$

$$\tilde{p}(s_{k+1})=\sum_{s_k}\hat{p}(s_k)\int_{-\infty}^{+\infty}\gamma(s_{k+1},x_k,s_k)dx_k \tag{31}$$

$$\tilde{x}(s_{k+1})=\tilde{p}^{-1}(s_{k+1})\sum_{s_k}\hat{p}(s_k)\int_{-\infty}^{+\infty}\gamma(s_{k+1},x_k,s_k)dx_k\int_{-\infty}^{+\infty}x_{k+1}f(x_{k+1}|s_{k+1},x_k,s_k)dx_{k+1} \tag{32}$$

$$\begin{aligned}\tilde{R}(s_{k+1})&=\tilde{p}^{-1}(s_{k+1})\sum_{s_k}\hat{p}(s_k)\int_{-\infty}^{+\infty}\gamma(s_{k+1},x_k,s_k)dx_k\times\\&\times\int_{-\infty}^{+\infty}\tilde{x}^0(s_{k+1})\tilde{x}^{0T}(s_{k+1})f(x_{k+1}|s_{k+1},x_k,s_k)dx_{k+1}\end{aligned} \tag{33}$$

где

- знак «$\tilde{\square}$» — прогнозируемые на один шаг;
- знак «$\hat{\square}$» — апостериорные характеристики.

$$\hat{x}^0(s_k)=x_k-\hat{x}(s_k),\ \tilde{x}^0(s_k)=x_k-\tilde{x}(s_k),\ \eta_k=\left[x_k^T,s_k^T,z_k^T,r_k^T\right]^T \tag{34}$$

$$\beta(\eta_{k+1}) = f(z_{k+1}|x_{k+1}, s_{k+1})\pi(r_{k+1}|r_k, x_{k+1}, s_{k+1})\tilde{f}(x_{k+1}|s_{k+1})$$

$$\gamma(s_{k+1}, x_k, s_k) = q(s_{k+1}|x_k, s_k)\tilde{f}(x_k|s_k)$$

Рассмотрим как это все выглядит на рисунке 4:

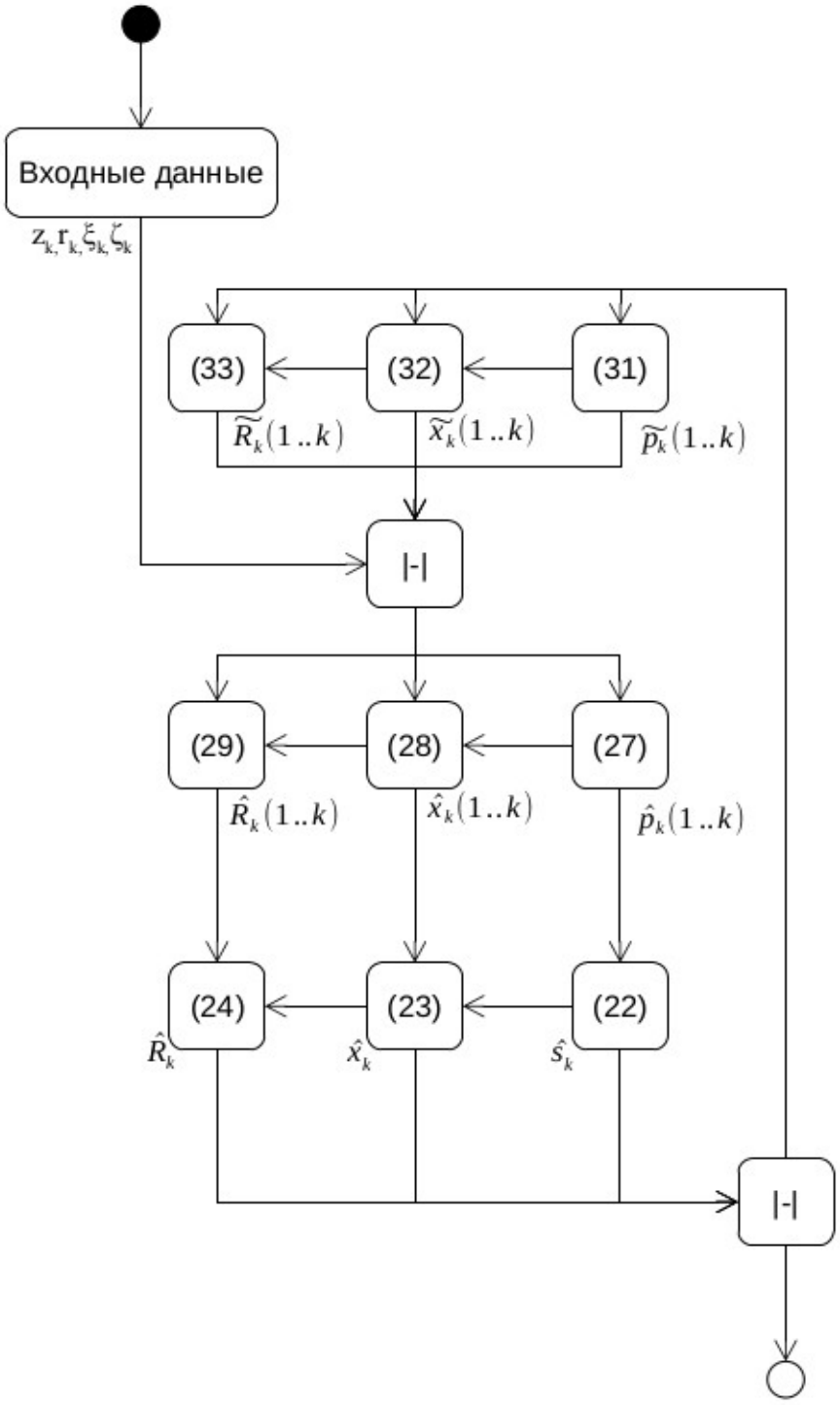

Рисунок 4 — Алгоритм классификатор-идентификатор-фильтр-дисперсиометр

На рисунке 4 отображены взаимодействия только главных функций, вспомогательные функции по нахождению например апостериорных распределений не отображены.

Теперь рассмотрим входные параметры, а также места в которых требуется дополнительная настройка в зависимости от исследуемого субъекта.

Нахождение начальных значения нулевого момента времени для апостериорных значений: $\hat{p}(s_0)$, $\hat{x}(s_0)$, $\hat{R}(s_0)$, можно вычислить по формулам (21).

Входные параметры:

z_k — представляет вектор изменения фазовых координат, где он представляет набор значений характеризующих объект, и с помощью которых например будет описывается ситуация, страдает ли человек ожирением или нет. Например возьмем ситуацию с потоком людей, где поступает в какой то момент времени вектор $z_k=(187,75)$, где :

1. Первый параметр, обозначает рост человека.

2. Второй параметр, обозначает вес человека.

r_k — представляет вектор индикации структур, он определяет статусные состояния объектов. Например возьмем пример с охраняемым объектом, на вход поступает вектор $r_k=(1,0)$, где :

1. Первый параметр, обозначает открытость охраняемого объекта.

2. Второй параметр, обозначает, что сейчас время активного хождения на работу или с неё.

Вспомогательные функции:

Функции (25) и (26) в отличие от прошлого будут задаваться с точностью до параметра. Где (25) является апостериорным распределение или в (26) прогнозируемым распределением при фиксированном наблюдением $z_0..z_k$ и $r_0..r_k$. То есть ожидаемые характеристики фазовых координат при всех состояний и всех индикаторов.

Функции (13) и (14) вычисляют соответствие фазовых координат и индикаторов соответственно, с учетом векторов возмущении ξ_k и вектора помех ζ_k .

Функция (35):

$$\pi(r_{k+1}|r_k, x_{k+1}, s_{k+1}) \ , \tag{35}$$

это условная вероятность перехода индикатора структуры из состояния r_k в состояние r_{k+1} при фиксированных x_{k+1} , s_{k+1} . Очень полезное применение для субъектов, у которых можно высчитывать индикаторы, например наличие очков у человека. Потому что можно делать сильное отсеивание найденных классов.

Функция (36):

$$q(s_{k+1}|x_k, s_k) \ , \tag{36}$$

это условная вероятность перехода структуры из состояния s_k в состояние s_{k+1} при фиксированном x_k . Очень важная функция, так как с помощью её можно настраивать вес значений индикаторов.

Метод представляет хорошую взаимосвязь с точки зрение переменных, но представляет ухудшение точности в связи с тем, что механизм был изменен с наблюдателя, где не производилась смена статуса, а только подтверждение того что новый статус подходил или нет для текущего момента времени. В текущей модели производится предварительный расчет для функций (13), (14), (25), (26), (35), (36). Что сильно упрощает работу с методом и позволяет его использовать с минимальными поправками, определив только с тем, какой класс объектов требуется находить.

Вывод к главе 2

1. Для разработки систем идентификации требуется полное исследование характеристик исследуемого субъекта.

2. После анализа биометрических идентификаторов, в ходе которого было выявлено, что большинство не используется в повседневной жизни.

3. Был произведен расчет биометрических идентификаторов на оценку полезности. Результат которого показал, что даже не самые качественные признаки будут очень полезны. Зачастую выигрывают те признаки, которые нельзя или очень трудно подделать.

4. Был рассмотрен Байесовский фильтр, в ходе которого было выявлено, что он может иметь большое количество модернизаций. И что его структура на основе «функции правдоподобия» может давать хороший процент соответствия.

Глава 3 Построение и реализация компьютерной модели

3.1 Перевод модели в практически применимую

На данный момент есть проблемы, связанные с текущей моделью. Основные проблемы модели связаны с тем, что подразумевается просчет всевозможных вариантов с учетом различных входных данных, из-за чего количество подготавливаемых к началу работы данных стремится к бесконечности. Поэтому для начала представим упрощённую схему текущей модели, где опустим некоторые моменты, с целью получения возможности реализовать эту модель программно. Далее такая модель будет называться — практическая модель. Где будем заменять формулы с абсолютных значений по всему числовому пространству к отрезкам или числительным методам.

Коррекция — классификатор, где оригинал формулы (27).

Так как модель не получится интегрировать при практических применений, то ее нужно упрощать. Переводя интегралы в наборы сумм и условий:

— Уравнение $f\left(z_{k+1}|x_{k+1},s_{k+1}\right)$ — условная плотность вероятности вектора z_{k+1} при фиксированном s_{k+1} и x_{k+1}

$$f\left(z_{k+1}|x_{k+1},s_{k+1}\right)=\begin{Bmatrix} 0,s_{k+1}=0,z_{k+1}\neq 0 \\ \sum 1-\frac{\left|x_{k+1}-z_{k+1}\right|}{x_k},s_{k+1}=1,z_{k+1}\neq 0 \\ \sum \frac{\left|x_{k+1}-z_{k+1}\right|}{x_k},s_{k+1}=2,z_{k+1}\neq 0 \\ 1,s_{k+1}=0,z_{k+1}\approx 0 \\ 0,s_{k+1}=1,z_{k+1}\approx 0 \\ 0,s_{k+1}=2,z_{k+1}\approx 0 \end{Bmatrix}, \quad (37)$$

— Уравнение $\tilde{f}\left(x_{k+1}|s_{k+1}\right)$ — прогнозируемое распределение вектора состояния $\left(x_{k+1}, s_{k+1}\right)$ при фиксированном наблюдении $\left(z\frac{\square}{0,k}, s\frac{\square}{0,k}\right)$

$$\tilde{f}\left(x_{k+1}|s_{k+1}\right)=\left\{\begin{array}{c} 0, s_{k+1}=0, x_{k+1}\neq 0 \\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}, s_{k+1}=1, x_{k+1}\neq 0 \\ \sum \frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}, s_{k+1}=2, x_{k+1}\neq 0 \\ 1, s_{k+1}=0, x_{k+1}\approx 0 \\ 0, s_{k+1}=1, x_{k+1}\approx 0 \\ 0, s_{k+1}=2, x_{k+1}\approx 0 \end{array}\right\}, \tag{38}$$

— Уравнение $\pi\left(r_{k+1}|r_k, x_{k+1}, s_{k+1}\right)$ — условная вероятность перехода структуры из состояния r_k в состояние r_{k+1} при фиксированных s_{k+1}, x_{k+1}.

$$\pi\left(r_{k+1}|r_k, x_{k+1}, s_{k+1}\right)=\left\{\begin{array}{c} 1, s_{k+1}=0, x_{k+1}\approx 0 \\ 0, s_{k+1}=1, x_{k+1}\approx 0 \\ 0, s_{k+1}=2, x_{k+1}\approx 0 \\ 0, s_{k+1}=0, x_{k+1}\neq 0 \\ \sum 1-\frac{\left|r_{k+1}-r_k\right|}{x_k}, s_{k+1}=1, x_{k+1}\neq 0 \\ \sum \frac{\left|r_{k+1}-r_k\right|}{x_k}, s_{k+1}=2, x_{k+1}\neq 0 \end{array}\right\}, \tag{39}$$

— Уравнение $\bar{f}\left(z_{k+1}, r_{k+1}\right)$ — нормированный коэффициент

$$\bar{f}\left(z_{k+1}, r_{k+1}\right)=\sum_{s_{k+1}} \tilde{p}\left(s_{k+1}\right) \int_{-\infty}^{\infty} \beta\left(\eta_{k+1}\right) dx_{k+1}, \tag{40}$$

приведем это уравнение к практическому виду

$$\bar{f}\left(z_{k+1}, r_{k+1}\right)=\sum_{s_{k+1}} \tilde{p}\left(s_{k+1}\right) \sum_{s_{k+1}} \beta\left(\eta_{k+1}\right), \tag{41}$$

в итоге мы получаем следующее:

$$\int_{-\infty}^{+\infty} \beta(\eta_{k+1})\, dx_{k+1} \to \sum_{s_k} \beta(\eta_{k+1}), \tag{42}$$

$$\hat{p}(s_{k+1}) = \left[\bar{f}(z_{k+1}, r_{k+1})\right]^{-1} \sum_{s_k} \beta(\eta_{k+1}). \tag{43}$$

Также можно описать сам $\beta(\eta_{k+1})$ из чего мы получаем следующее соответствие:

$$\beta(\eta_{k+1}) = \begin{Bmatrix} 0, s_{k+1}=0, z_{k+1}\neq 0 \\ \sum 1 - \frac{|x_{k+1} - z_{k+1}|}{x_k}, s_{k+1}=1, z_{k+1}\neq 0 \\ \sum \frac{|x_{k+1} - z_{k+1}|}{x_k}, s_{k+1}=2, z_{k+1}\neq 0 \\ 1, s_{k+1}=0, z_{k+1}\approx 0 \\ 0, s_{k+1}=1, z_{k+1}\approx 0 \\ 0, s_{k+1}=2, z_{k+1}\approx 0 \end{Bmatrix} \times$$

$$\times \begin{Bmatrix} 1, s_{k+1}=0, x_{k+1}\approx 0 \\ 0, s_{k+1}=1, x_{k+1}\approx 0 \\ 0, s_{k+1}=2, x_{k+1}\approx 0 \\ 0, s_{k+1}=0, x_{k+1}\neq 0 \\ \sum 1 - \frac{|r_{k+1} - r_k|}{x_k}, s_{k+1}=1, x_{k+1}\neq 0 \\ \sum \frac{|r_{k+1} - r_k|}{x_k}, s_{k+1}=2, x_{k+1}\neq 0 \end{Bmatrix} \times \tag{44}$$

$$\times \begin{Bmatrix} 0, s_{k+1}=0, x_{k+1}\neq 0 \\ \sum 1 - \frac{|x_{k+1} - \hat{x}_{k+1}|}{x_k}, s_{k+1}=1, x_{k+1}\neq 0 \\ \sum \frac{|x_{k+1} - \hat{x}_{k+1}|}{x_k}, s_{k+1}=2, x_{k+1}\neq 0 \\ 1, s_{k+1}=0, x_{k+1}\approx 0 \\ 0, s_{k+1}=1, x_{k+1}\approx 0 \\ 0, s_{k+1}=2, x_{k+1}\approx 0 \end{Bmatrix}$$

Подставляя в (41) получаем:

$$\overline{f}\left(z_{k+1},r_{k+1}\right)=\sum_{s_{k+1}}\widetilde{p}\left(s_{k+1}\right)\sum_{s_{k+1}}\left\{\begin{array}{c}0,s_{k+1}=0,z_{k+1}\neq 0\\ \sum 1-\frac{\left|x_{k+1}-z_{k+1}\right|}{x_k},s_{k+1}=1,z_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-z_{k+1}\right|}{x_k},s_{k+1}=2,z_{k+1}\neq 0\\ 1,s_{k+1}=0,z_{k+1}\approx 0\\ 0,s_{k+1}=1,z_{k+1}\approx 0\\ 0,s_{k+1}=2,z_{k+1}\approx 0\end{array}\right\}\times$$

$$\times\left\{\begin{array}{c}1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\\ 0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{\left|r_{k+1}-r_k\right|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|r_{k+1}-r_k\right|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\end{array}\right\}\times \qquad . \qquad (45)$$

$$\times\left\{\begin{array}{c}0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\\ 1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\end{array}\right\}$$

Теперь следует добавить в уравнение (43) следующее:

— функцию нахождения нормированного коэффициента (45):

— функцию нахождение бэты (44).

И получаем следующее:

$$\hat{p}(s_{k+1})=[\sum_{s_{k+1}}\tilde{p}(s_{k+1})\sum_{s_{k+1}}[\begin{Bmatrix}0,s_{k+1}=0,z_{k+1}\neq 0\\ \sum 1-\frac{|x_{k+1}-z_{k+1}|}{x_k},s_{k+1}=1,z_{k+1}\neq 0\\ \sum\frac{|x_{k+1}-z_{k+1}|}{x_k},s_{k+1}=2,z_{k+1}\neq 0\\ 1,s_{k+1}=0,z_{k+1}\approx 0\\ 0,s_{k+1}=1,z_{k+1}\approx 0\\ 0,s_{k+1}=2,z_{k+1}\approx 0\end{Bmatrix}\times$$

$$\times\begin{Bmatrix}1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\\ 0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{|r_{k+1}-r_k|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{|r_{k+1}-\hat{r}_k|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\end{Bmatrix}\begin{Bmatrix}0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{|x_{k+1}-\hat{x}_{k+1}|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{|x_{k+1}-\hat{x}_{k+1}|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\\ 1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\end{Bmatrix}]]^{-1}\times$$

$$\times\sum_{s_k}[\begin{Bmatrix}0,s_{k+1}=0,z_{k+1}\neq 0\\ \sum 1-\frac{|x_{k+1}-z_{k+1}|}{x_k},s_{k+1}=1,z_{k+1}\neq 0\\ \sum\frac{|x_{k+1}-z_{k+1}|}{x_k},s_{k+1}=2,z_{k+1}\neq 0\\ 1,s_{k+1}=0,z_{k+1}\approx 0\\ 0,s_{k+1}=1,z_{k+1}\approx 0\\ 0,s_{k+1}=2,z_{k+1}\approx 0\end{Bmatrix}\begin{Bmatrix}1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\\ 0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{|r_{k+1}-r_k|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{|r_{k+1}-r_k|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\end{Bmatrix}\times$$

$$\times\begin{Bmatrix}0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{|x_{k+1}-\hat{x}_{k+1}|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{|x_{k+1}-\hat{x}_{k+1}|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\\ 1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\end{Bmatrix}] \qquad (46)$$

Коррекция — фильтр, где оригинал формулы (28). Исходя из предыдущего получаем следующее:

$$\hat{x}(s_{k+1})=\left[\bar{f}(z_{k+1},r_{k+1})\hat{p}(s_{k+1})\right]^{-1}\tilde{p}(s_{k+1})\sum_{s_k}x_{k+1}\beta(\eta_{k+1}), \qquad (47)$$

Где x_{k+1} это вектор значений, и на выходе $\hat{x}(s_{k+1})$ тоже будет вектор

значений, поэтому это будет выглядеть следующим образом:

$$\hat{x}(s_{k+1})=\left[\bar{f}(z_{k+1},r_{k+1})\,\hat{p}(s_{k+1})\right]^{-1}\tilde{p}(s_{k+1})\begin{pmatrix}x_{k+1,0}\\x_{k+1,1}\\\dots\\x_{k+1,n-1}\\x_{k+1,n}\end{pmatrix}\sum_{s_k}\beta(\eta_{k+1}), \tag{48}$$

После подстановки функции нахождения нормированного коэффициента (45) и нахождение бэты (44) в уравнение (48)

$$\hat{x}(s_{k+1})=[\sum_{s_{k+1}}\tilde{p}(s_{k+1})\sum_{s_{k+1}}\begin{Bmatrix}0,s_{k+1}=0,z_{k+1}\neq 0\\\sum 1-\frac{|x_{k+1}-z_{k+1}|}{x_k},s_{k+1}=1,z_{k+1}\neq 0\\\sum\frac{|x_{k+1}-z_{k+1}|}{x_k},s_{k+1}=2,z_{k+1}\neq 0\\1,s_{k+1}=0,z_{k+1}\approx 0\\0,s_{k+1}=1,z_{k+1}\approx 0\\0,s_{k+1}=2,z_{k+1}\approx 0\end{Bmatrix}\times$$

$$\times\begin{Bmatrix}1,s_{k+1}=0,x_{k+1}\approx 0\\0,s_{k+1}=1,x_{k+1}\approx 0\\0,s_{k+1}=2,x_{k+1}\approx 0\\0,s_{k+1}=0,x_{k+1}\neq 0\\\sum 1-\frac{|r_{k+1}-r_k|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\\sum\frac{|r_{k+1}-r_k|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\end{Bmatrix}\begin{Bmatrix}0,s_{k+1}=0,x_{k+1}\neq 0\\\sum 1-\frac{|x_{k+1}-\hat{x}_{k+1}|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\\sum 1-\frac{|x_{k+1}-\hat{x}_{k+1}|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\\1,s_{k+1}=0,x_{k+1}\approx 0\\0,s_{k+1}=1,x_{k+1}\approx 0\\0,s_{k+1}=2,x_{k+1}\approx 0\end{Bmatrix}\times \tag{49}$$

$$\times\hat{p}(s_{k+1})]^{-1}\tilde{p}(s_{k+1})\begin{pmatrix}x_{k+1,0}\\x_{k+1,1}\\\dots\\x_{k+1,n-1}\\x_{k+1,n}\end{pmatrix}\sum_{s_k}\begin{Bmatrix}0,s_{k+1}=0,z_{k+1}\neq 0\\\sum 1-|x_{k+1}-z_{k+1}|,s_{k+1}=1,z_{k+1}\neq 0\\\sum|x_{k+1}-z_{k+1}|,s_{k+1}=2,z_{k+1}\neq 0\\1,s_{k+1}=0,z_{k+1}\approx 0\\0,s_{k+1}=1,z_{k+1}\approx 0\\0,s_{k+1}=2,z_{k+1}\approx 0\end{Bmatrix}\times$$

$$\times\begin{Bmatrix}1,s_{k+1}=0,x_{k+1}\approx 0\\0,s_{k+1}=1,x_{k+1}\approx 0\\0,s_{k+1}=2,x_{k+1}\approx 0\\0,s_{k+1}=0,x_{k+1}\neq 0\\\sum 1-|r_{k+1}-r_k|,s_{k+1}=1,x_{k+1}\neq 0\\\sum|r_{k+1}-r_k|,s_{k+1}=2,x_{k+1}\neq 0\end{Bmatrix}\begin{Bmatrix}0,s_{k+1}=0,x_{k+1}\neq 0\\\sum 1-|x_{k+1}-\hat{x}_{k+1}|,s_{k+1}=1,x_{k+1}\neq 0\\\sum|x_{k+1}-\hat{x}_{k+1}|,s_{k+1}=2,x_{k+1}\neq 0\\1,s_{k+1}=0,x_{k+1}\approx 0\\0,s_{k+1}=1,x_{k+1}\approx 0\\0,s_{k+1}=2,x_{k+1}\approx 0\end{Bmatrix}$$

Коррекция — дисперсиометр, где оригинал формулы (29). Исходя из предыдущего получаем следующее:

$$\hat{R}(s_{k+1})=\left[\bar{f}(z_{k+1},r_{k+1})\hat{p}(s_{k+1})\right]^{-1}\tilde{p}(s_{k+1})\sum_{s_k}\hat{x}^0(s_{k+1})\hat{x}^{0T}(s_{k+1})\beta(\eta_{k+1}), \quad (50)$$

где

$$\hat{x}^0(s_{k+1})\hat{x}^{0T}(s_{k+1})=x_k-\hat{x}(s_k), \quad (51)$$

и получим следующее:

$$\hat{R}(s_{k+1})=\left[\bar{f}(z_{k+1},r_{k+1})\hat{p}(s_{k+1})\right]^{-1}\tilde{p}(s_{k+1})\sum_{s_k}\left|x_{k+1}-\hat{x}(s_{k+1})\right|\beta(\eta_{k+1}), \quad (52)$$

как и в прошлом уравнении (28), x_k это вектор значений.

На выходе $\hat{R}(s_k)$ тоже будет вектор значений, поэтому это будет выглядеть следующим образом:

$$\hat{R}(s_{k+1})=\left[\bar{f}(z_{k+1},r_{k+1})\hat{p}(s_{k+1})\right]^{-1}\tilde{p}(s_{k+1})\begin{pmatrix}\left|x_{k+1}-\hat{x}(s_{k+1})\right|_0\\ \left|x_{k+1}-\hat{x}(s_{k+1})\right|_1\\ \dots\\ \left|x_{k+1}-\hat{x}(s_{k+1})\right|_{n-1}\\ \left|x_{k+1}-\hat{x}(s_{k+1})\right|_n\end{pmatrix}\sum_{s_k}\beta(\eta_{k+1}), \quad (53)$$

Теперь следует добавить в уравнение (53) следующее:

1. функцию нахождения нормированного коэффициента (45):
2. функцию нахождение бэты (44).

И получаем следующее:

$$\hat{R}\left(s_{k+1}\right)=\left[\sum_{s_{k+1}}\tilde{p}\left(s_{k+1}\right)\sum_{s_{k+1}}\begin{Bmatrix}0,s_{k+1}=0,z_{k+1}\neq 0\\ \sum 1-\frac{\left|x_{k+1}-z_{k+1}\right|}{x_k},s_{k+1}=1,z_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-z_{k+1}\right|}{x_k},s_{k+1}=2,z_{k+1}\neq 0\\ 1,s_{k+1}=0,z_{k+1}\approx 0\\ 0,s_{k+1}=1,z_{k+1}\approx 0\\ 0,s_{k+1}=2,z_{k+1}\approx 0\end{Bmatrix}\times\right.$$

$$\times\begin{Bmatrix}1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\\ 0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{\left|r_{k+1}-r_k\right|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|r_{k+1}-r_k\right|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\end{Bmatrix}\begin{Bmatrix}0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\\ 1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\end{Bmatrix}\Big]^{-1}\times$$

$$\times\tilde{p}\left(s_{k+1}\right)\begin{pmatrix}\left|x_{k+1}-\hat{x}\left(s_{k+1}\right)\right|_0\\ \left|x_{k+1}-\hat{x}\left(s_{k+1}\right)\right|_1\\ \ldots\\ \left|x_{k+1}-\hat{x}\left(s_{k+1}\right)\right|_{n-1}\\ \left|x_{k+1}-\hat{x}\left(s_{k+1}\right)\right|_n\end{pmatrix}\sum_{s_k}\begin{Bmatrix}0,s_{k+1}=0,z_{k+1}\neq 0\\ \sum 1-\frac{\left|x_{k+1}-z_{k+1}\right|}{x_k},s_{k+1}=1,z_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-z_{k+1}\right|}{x_k},s_{k+1}=2,z_{k+1}\neq 0\\ 1,s_{k+1}=0,z_{k+1}\approx 0\\ 0,s_{k+1}=1,z_{k+1}\approx 0\\ 0,s_{k+1}=2,z_{k+1}\approx 0\end{Bmatrix}\times$$

$$\times\begin{Bmatrix}1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\\ 0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{\left|r_{k+1}-r_k\right|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|r_{k+1}-r_k\right|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\end{Bmatrix}\begin{Bmatrix}0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\\ 1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\end{Bmatrix}. \quad (54)$$

Прогноз — классификатор, где оригинал формулы (31).

Уравнение $\tilde{f}\left(x_k|s_k\right)$ — прогнозируемая на один шаг дискретности вперед плотность вероятности вектора x_k, а $q\left(s_{k+1}|x_k,s_k\right)$ — условная вероятность перехода структуры из состояния s_k в состояние s_{k+1} при фиксированном x_k.

Уравнение $q\left(s_{k+1}|x_k,s_k\right)$ заменим на простое уравнение, состоящий из условий:

$$q\left(s_{k+1}|x_k,s_k,\eta_{k+1}\right)=\begin{matrix}0,s_k=0,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=0,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=0,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=0,s_{k+1}=2,z_{k+1}\approx 0\\ 0,s_k=1,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=1,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=1,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=1,s_{k+1}=2,z_{k+1}\approx 0\\ 0,s_k=2,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=2,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=2,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=2,s_{k+1}=2,z_{k+1}\approx 0\end{matrix}, \quad (55)$$

В итоге мы получаем следующее:

$$\gamma\left(s_{k+1},x_k,s_k,\eta_{k+1}\right)=q\left(s_{k+1}|x_k,s_k,\eta_{k+1}\right)\tilde{f}\left(x_k|s_k\right), \quad (56)$$

где

$$\int_{-\infty}^{+\infty}\gamma\left(s_{k+1},x_k,s_k\right)dx_k\to\sum_{s_k}q\left(s_{k+1}|x_k,s_k,\eta_{k+1}\right)\tilde{f}\left(x_k|s_k\right), \quad (57)$$

Где введя ее в уравнение (31) мы получаем следующее:

$$\tilde{p}(s_{k+1})=\sum_{s_k}\hat{p}(s_k)\sum_{s_k}q(s_{k+1}|x_k,s_k,\eta_{k+1})\tilde{f}(x_k|s_k), \tag{58}$$

После подстановки функции прогнозируемое распределение вектора состояния (38) и функции нахождения условной вероятности перехода структуры (55) в уравнение (58)

(59)

$$\tilde{p}(s_{k+1})=\sum_{s_k}\hat{p}(s_k)\sum_{s_k}\begin{Bmatrix}0,s_k=0,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=0,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=0,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=0,s_{k+1}=2,z_{k+1}\approx 0\\ 0,s_k=1,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=1,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=1,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=1,s_{k+1}=2,z_{k+1}\approx 0\\ 0,s_k=2,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=2,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=2,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=2,s_{k+1}=2,z_{k+1}\approx 0\end{Bmatrix}\times$$

$$\times\begin{Bmatrix}0,s_{k+1}=0,x_{k+1}\neq 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_{k+1}=2,x_{k+1}\neq 0\\ 1,s_{k+1}=0,x_{k+1}\approx 0\\ 0,s_{k+1}=1,x_{k+1}\approx 0\\ 0,s_{k+1}=2,x_{k+1}\approx 0\end{Bmatrix}.$$

Прогноз — фильтр, где оригинал формулы (32). Исходя из предыдущего получаем следующее:

$$\tilde{x}(s_{k+1}) = \tilde{p}^{-1}(s_{k+1}) \sum_{s_k} \hat{p}(s_k) \sum_{s_k} q(s_{k+1}|x_k, s_k, \eta_{k+1}) \tilde{f}(x_k|s_k) \times \\ \times \int_{-\infty}^{+\infty} x_{k+1} f(x_{k+1}|s_{k+1}, x_k, s_k) dx_{k+1}, \quad (60)$$

где $f(x_{k+1}|s_{k+1}, x_k, s_k)$ — распределение вероятности вектора состояний (x_{k+1}, s_{k+1}) при (x_k, s_k).

$$\int_{-\infty}^{+\infty} f(x_{k+1}|s_{k+1}, x_k, s_k) dx_k \to \sum_{s_k} f(x_{k+1}|s_{k+1}, x_k, s_k), \quad (61)$$

Следовательно он будет выглядеть следующим образом:

$$\tilde{x}(s_{k+1}) = \tilde{p}^{-1}(s_{k+1}) \sum_{s_k} \hat{p}(s_k) \sum_{s_k} q(s_{k+1}|x_k, s_k, \eta_{k+1}) \tilde{f}(x_k|s_k) \times \\ \times \sum_{s_k} f(x_{k+1}|s_{k+1}, x_k, s_k), \quad (62)$$

результатом этого уравнения является вектор значений, следовательно дополняем его:

$$\tilde{x}(s_{k+1}) = \tilde{p}^{-1}(s_{k+1}) \sum_{s_k} \hat{p}(s_k) \sum_{s_k} q(s_{k+1}|x_k, s_k, \eta_{k+1}) \tilde{f}(x_k|s_k) \times \\ \times \sum_{s_k} x_{k+1} f(x_{k+1}|s_{k+1}, x_k, s_k), \quad (63)$$

как и в прошлом уравнении (28), x_k это вектор значений, и на выходе $\tilde{x}(s_k)$ тоже будет вектор значений, поэтому это будет выглядеть следующим образом:

$$\tilde{x}(s_{k+1})=\tilde{p}^{-1}(s_{k+1})\sum_{s_k}\hat{p}(s_k)\sum_{s_k}q(s_{k+1}|x_k,s_k,\eta_{k+1})\tilde{f}(x_k|s_k)\times$$
$$\times\begin{pmatrix}x_{k+1,0}\\x_{k+1,1}\\\dots\\x_{k+1,n-1}\\x_{k+1,n}\end{pmatrix}\sum_{s_k}f(x_{k+1}|s_{k+1},x_k,s_k) \quad . \qquad (64)$$

Где распределение вероятности вектора состояний (61) является функцией, которая задается формулой. Но в связи с тем что:

— имеется несколько биометрических идентификаторов;

— имеется несколько состояний структуры.

Мы сделаем линейные уравнения, задав их следующим образом:

$$f(x_{k+1}|s_{k+1},x_k,s_k)=\begin{cases}1, s_k=0, s_{k+1}=0, x_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}, s_k=0, s_{k+1}=1, x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}, s_k=0, s_{k+1}=2, x_{k+1}\neq 0\\ 1, s_k=1, s_{k+1}=0, x_{k+1}=0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}, s_k=1, s_{k+1}=1, x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}, s_k=1, s_{k+1}=2, x_{k+1}\neq 0\\ 1, s_k=2, s_{k+1}=0, x_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}, s_k=2, s_{k+1}=1, x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}, s_k=2, s_{k+1}=2, x_{k+1}\neq 0\\ 0, \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}>1\end{cases}, \qquad (65)$$

Подставив в (64) получаем следующее:

$$\widetilde{x}\left(s_{k+1}\right)=\widetilde{p}^{-1}\left(s_{k+1}\right)\sum_{s_k}\hat{p}\left(s_k\right)\sum_{s_k}q\left(s_{k+1}|x_k,s_k,\eta_{k+1}\right)\tilde{f}\left(x_k|s_k\right)\times$$

$$\times\begin{pmatrix}x_{k+1,0}\\x_{k+1,1}\\\cdots\\x_{k+1,n-1}\\x_{k+1,n}\end{pmatrix}\sum_{s_k}\begin{matrix}1,s_k=0,s_{k+1}=0,x_{k+1}\approx 0\\\sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=1,x_{k+1}\neq 0\\\sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=2,x_{k+1}\neq 0\\1,s_k=1,s_{k+1}=0,x_{k+1}=0\\\sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=1,x_{k+1}\neq 0\\\sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=2,x_{k+1}\neq 0\\1,s_k=2,s_{k+1}=0,x_{k+1}\approx 0\\\sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=1,x_{k+1}\neq 0\\\sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=2,x_{k+1}\neq 0\\0,\sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k}>1\end{matrix}, \quad (66)$$

Прогноз — дисперсиометр, где оригинал формулы (33). Исходя из предыдущего получаем следующее:

$$\begin{matrix}\widetilde{R}\left(s_{k+1}\right)=\widetilde{p}^{-1}\left(s_{k+1}\right)\sum_{s_k}\hat{p}\left(s_k\right)\sum_{s_k}q\left(s_{k+1}|x_k,s_k,\eta_{k+1}\right)\tilde{f}\left(x_k|s_k\right)\times\\\times\sum_{s_k}\tilde{x}^0\left(s_{k+1}\right)\tilde{x}^{0T}\left(s_{k+1}\right)f\left(x_{k+1}|s_{k+1},x_k,s_k\right)dx_{k+1}\end{matrix}, \quad (67)$$

где

$$\tilde{x}^0\left(s_{k+1}\right)\tilde{x}^{0T}\left(s_{k+1}\right)=x_k-\tilde{x}\left(s_k\right), \quad (68)$$

и получим следующее:

$$\widetilde{R}(s_{k+1}) = \widetilde{p}^{-1}(s_{k+1}) \sum_{s_k} \hat{p}(s_k) \int_{-\infty}^{+\infty} \gamma(s_{k+1}, x_k, s_k) dx_k \times \\ \times \int_{-\infty}^{+\infty} \widetilde{x}^0(s_{k+1}) \widetilde{x}^{0T}(s_{k+1}) f(x_{k+1} | s_{k+1}, x_k, s_k) dx_{k+1}, \quad (69)$$

как и в прошлом уравнении (27), x_k это вектор значений, и на выходе $\widetilde{R}(s_k)$ тоже будет вектор значений, поэтому это будет выглядеть следующим образом:

$$\widetilde{R}(s_{k+1}) = \widetilde{p}^{-1}(s_{k+1}) \sum_{s_k} \hat{p}(s_k) \sum_{s_k} q(s_{k+1} | x_k, s_k, \eta_{k+1}) \widetilde{f}(x_k | s_k) \times \\ \times \begin{pmatrix} (x_{k+1} - \widetilde{x}(s_{k+1}))_0 \\ (x_{k+1} - \widetilde{x}(s_{k+1}))_1 \\ \dots \\ (x_{k+1} - \widetilde{x}(s_{k+1}))_{n-1} \\ (x_{k+1} - \widetilde{x}(s_{k+1}))_n \end{pmatrix} \sum_{s_k} f(x_{k+1} | s_{k+1}, x_k, s_k), \quad (70)$$

Где можно заметить

И подставим сюда:

— функцию вычисления условной вероятности перехода структуры из состояния s_k в состояние s_{k+1} при фиксированном x_k (55);

— функцию вычисления прогнозируемого распределение вектора состояния (x_{k+1}, s_{k+1}) при фиксированном наблюдении $\left(z \frac{\square}{0,k}, s \frac{\square}{0,k} \right)$ (38).

— функцию вычисления условной плотность вероятности вектора x_{k+1} при фиксированном s_{k+1}, x_k и s_k (65)

Где получим следующее:

$$\widetilde{R}\left(s_{k+1}\right)=\widetilde{p}^{-1}\left(s_{k+1}\right)\sum_{s_k}\hat{p}\left(s_k\right)\sum_{s_k}\begin{matrix}0,s_k=0,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=0,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=0,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=0,s_{k+1}=2,z_{k+1}\approx 0\\ 0,s_k=1,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=1,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=1,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=1,s_{k+1}=2,z_{k+1}\approx 0\\ 0,s_k=2,s_{k+1}=0,z_{k+1}\neq 0\\ 1,s_k=2,s_{k+1}=0,z_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=1,z_{k+1}\neq 0\\ 0,s_k=2,s_{k+1}=1,z_{k+1}\approx 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=2,z_{k+1}\neq 0\\ 0,s_k=2,s_{k+1}=2,z_{k+1}\approx 0\end{matrix}\times$$

$$\times\begin{Bmatrix}\left(x_{k+1}-\widetilde{x}\left(s_{k+1}\right)\right)_0\\ \left(x_{k+1}-\widetilde{x}\left(s_{k+1}\right)\right)_1\\ \cdots\\ \left(x_{k+1}-\widetilde{x}\left(s_{k+1}\right)\right)_{n-1}\\ \left(x_{k+1}-\widetilde{x}\left(s_{k+1}\right)\right)_n\end{Bmatrix}\sum_{s_k}\begin{Bmatrix}1,s_k=0,s_{k+1}=0,x_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=0,s_{k+1}=2,x_{k+1}\neq 0\\ 1,s_k=1,s_{k+1}=0,x_{k+1}=0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=1,s_{k+1}=1,x_{k+1}\neq 0\\ 1,s_k=2,s_{k+1}=0,x_{k+1}\approx 0\\ \sum 1-\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=1,x_{k+1}\neq 0\\ \sum\frac{\left|x_{k+1}-\hat{x}_{k+1}\right|}{x_k},s_k=2,s_{k+1}=2,x_{k+1}\neq 0\end{Bmatrix}, \quad (71)$$

Теперь рассмотрим идентификатор структуры описанный в формуле (22). В данной ситуации мы ничего не меняем, требуется просто найти в векторе максимальное значение, а также само *s* при котором оно было найдено.

В уравнение нахождения апостериорного вектора математического ожидания оригинал формулы в (23)

Оригинал формулы:

$$\hat{x}_k = \sum_{s_k} \hat{x}_k(s_k)\,\hat{p}(s_k), \tag{72}$$

тут требуется перевести уравнение для вывода, так как на выходе будет вектор значений:

$$\hat{x}_k = \sum_{s_k} \begin{pmatrix} (\hat{x}_k(s_k))_0 \\ (\hat{x}_k(s_k))_1 \\ \dots \\ (\hat{x}_k(s_k))_{n-1} \\ (\hat{x}_k(s_k))_n \end{pmatrix} \hat{p}(s_k), \tag{73}$$

А в уравнение нахождения апостериорного вектора ковариационной матрицы оригиналом является (24) , где

$$\hat{x}(s_k)\hat{x}^T(s_k) \to x_k - \hat{x}(s_k), \tag{74}$$

и расписываем под вектор значений и получаем

$$\hat{R}_k = \sum_{s_k} \lfloor \hat{R}(s_k) + \begin{pmatrix} (\hat{x}_k(s_k))_0 \\ (\hat{x}_k(s_k))_1 \\ \dots \\ (\hat{x}_k(s_k))_{n-1} \\ (\hat{x}_k(s_k))_n \end{pmatrix} \rfloor \hat{p}(s_k) - \begin{pmatrix} x_{k,0} \\ x_{k,1} \\ \dots \\ x_{k,n-1} \\ x_{k,n} \end{pmatrix}. \tag{75}$$

После того как рассмотрели все формулы, и упростили все до элементарных переменных. Рассмотрим на рисунке 5.

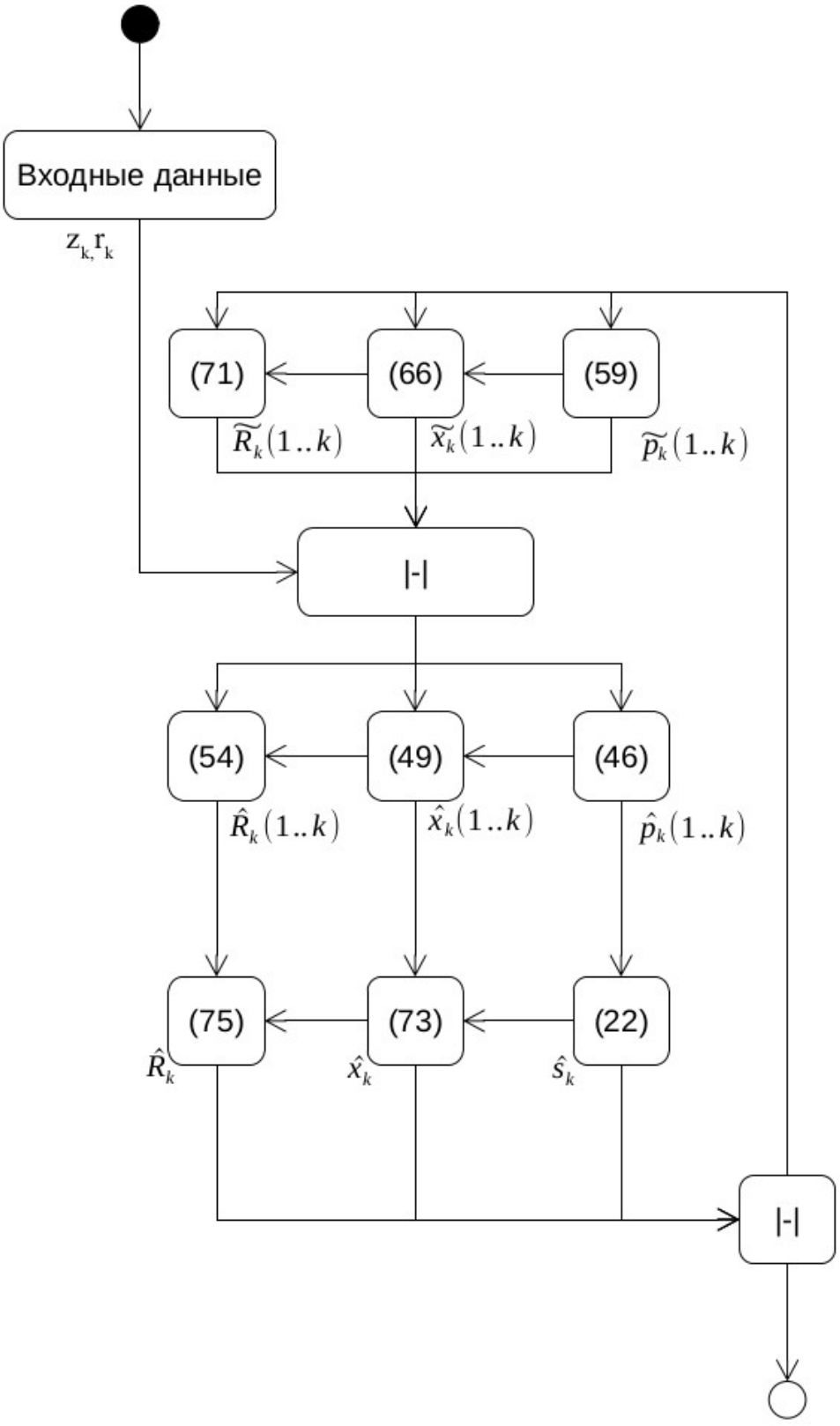

Рисунок 5 — Расписанный алгоритм классификатор–идентификатор–фильтр–дисперсиометр

Где можно заметить, что уже нет в качестве входных переменных вектор

возмущений ξ_k и вектор помех ζ_k .

3.2 Генерация входных данных

Биометрические идентификаторы были рассмотрены в 2.2 под главе. Основываясь на этом, опишем десятерых выдуманных человека, которых будем использовать в дальнейшем в таблице 6.

Таблица 6 — Описание тестовых данных

Человек	Цвет волос (L)	Цвет волос (A)	Цвет волос (B)	Рост	Вес	Наличие очков
Petya	53	8	54	170	85	false
Vasya	58	10	60	165	70	false
Masha	30	10	70	150	60	true
Maksim	25	45	52	170	68	false
Dasha	64	10	50	165	67	true
Anna	60	65	54	171	85	false
Lena	48	75	25	152	68	false
Misha	35	5	25	180	60	true
Kostya	10	45	75	192	67	false
Natasha	25	65	95	205	50	true

Сами значение будут подаваться в поток значений с применением нормального распределения, чтобы приблизить к реальным данным.

Вывод к главе 3

1. Методы в основном разрабатываются с точки зрения теоретического использования, где они подразумевают почти 100% точность. Но из-за того, что невозможно например пройтись по всей прямой, приходится вводить ряд ограничений, которые понижают точность алгоритма.

2. Хорошо разработанные модификации к алгоритмам, позволяют их использовать даже там, где они не подразумевались. Например, превратить алгоритм, которая анализировал весь промежуток времени и давала результат в конце, в плавающее окно, которое анализирует текущее состояние.

Глава 4 Корректировка математической модели и вычислительный эксперимент

4.1 Реализация заполнения данных для тестовой работы

Сама программа состоит из модуля генерации данных и модуля анализа. И выглядит следующим образом на рисунке 6.

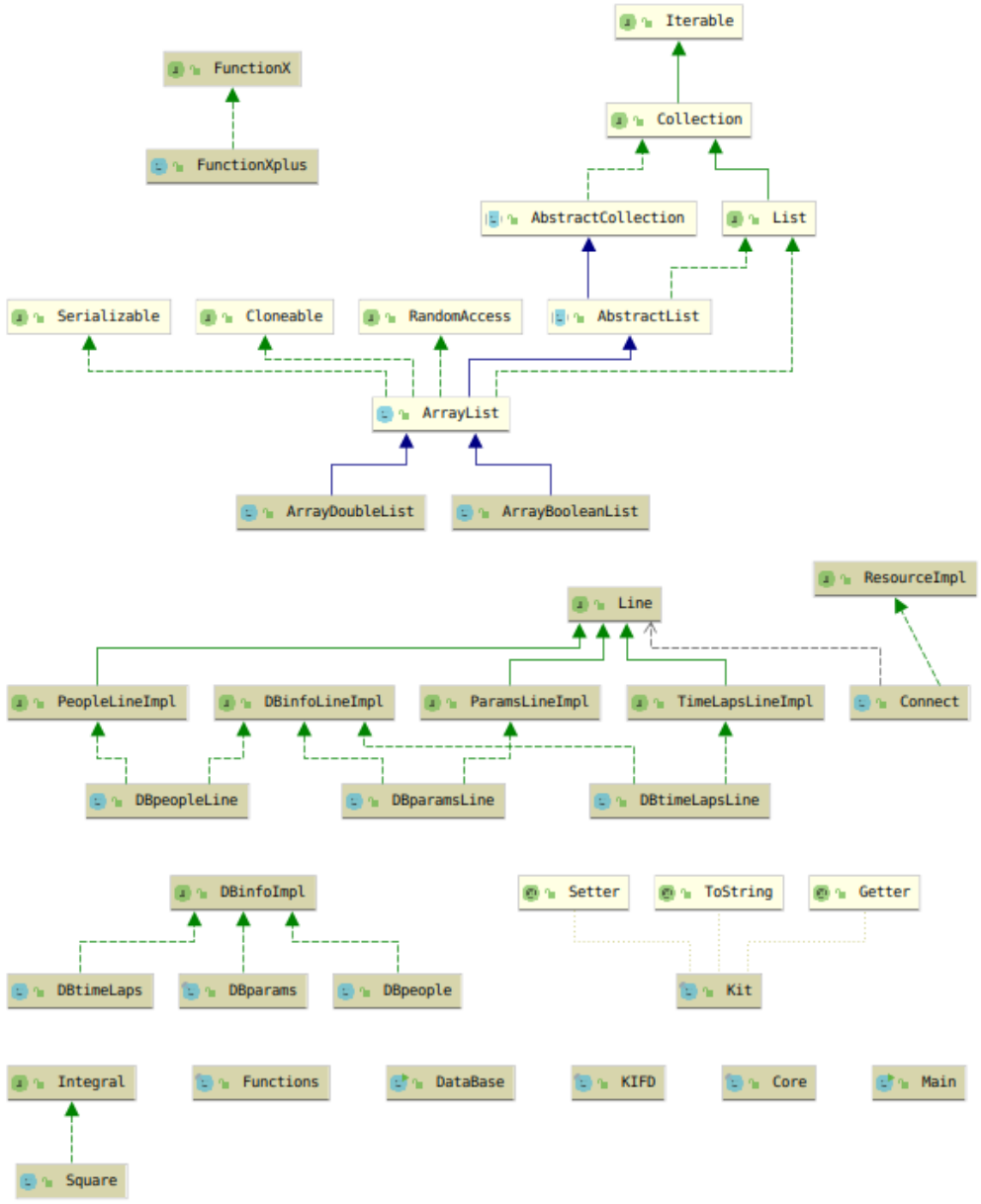

Рисунок 6 — Диаграмма зависимостей классов

Программа является гибкой и для быстрой работы с данными она

использует SQLite. Объявление характеристик объекта показан на рисунке 7.

```
paramsDB.connect();
paramsDB.createMainTable();
new DBparamsLine(paramsDB, name: "hair_l", expectedValue: 64, dispersion: 2).insert();
new DBparamsLine(paramsDB, name: "hair_a", expectedValue: 10, dispersion: 2).insert();
new DBparamsLine(paramsDB, name: "hair_b", expectedValue: 50, dispersion: 2).insert();
new DBparamsLine(paramsDB, name: "height", expectedValue: 165, dispersion: 2).insert();
new DBparamsLine(paramsDB, name: "weight", expectedValue: 67, dispersion: 2).insert();
new DBparamsLine(paramsDB, name: "glasses").insert();
paramsDB.sqlSelectAllPrint();

DBparams<DBparamsLine> params = new DBparams<~>();
params.setResult(paramsDB.sqlGetAll(),paramsDB);
```

Рисунок 7 — Объявление свойств объекта

Где вводится;

— name, название объекта;

— expectedValue, оригинальное значение искомого объекта;

— dispersion, дисперсия значения.

После чего создается таблица «params» и заполняется этими значениям. Пример данных показан в таблице 7.

Таблица 7 — Содержимое таблицы «params»

id	name	expectedValue	dispersion	state
1	hair_l	64	2	0
2	hair_a	10	2	0
3	hair_b	50	2	0
4	height	165	2	0
5	weight	67	2	0
6	glasses	0	0	1

Где в последней колонке можно заметить флаг, который определяет значение будет относиться к вектору фазовых координат x_k или к индикации структуры r_k .

Где после этого идет создание еще двух таблиц, которые столбцы которых основываются на содержимом таблицы «params», это таблицы:

— «people» — информация о других объектов, эта таблица требуется для хранения тестовых данных;

— «timelaps» — таблица которая хранит строки как отдельные моменты времени.

Для начала требуется заполнить таблицу «people», так как на базе нее будет производиться генерация тестовых данных в таблицу timelaps. Делается это следующим образом, показанным на рисунке 9.

```
peopleDB.connect();
peopleDB.createMainTable();
new DBpeopleLine(peopleDB, params,  name: "Petya",  probability: 0.2,
        ...values: "53", "8", "54", "170", "85", "false").insert();
new DBpeopleLine(peopleDB, params,  name: "Vasya",  probability: 0.2,
        ...values: "58", "10", "60", "165", "70", "false").insert();
new DBpeopleLine(peopleDB, params,  name: "Masha",  probability: 0.2,
        ...values: "30", "10", "70", "150", "60", "true").insert();
new DBpeopleLine(peopleDB, params,  name: "Maksim",  probability: 0.2,
        ...values: "25", "45", "52", "170", "68", "false").insert();
new DBpeopleLine(peopleDB, params,  name: "Dasha",  probability: 0.2,
        ...values: "64", "10", "50", "165", "67", "true").insert();
new DBpeopleLine(peopleDB, params,  name: "Anna",  probability: 0.2,
        ...values: "60", "65", "54", "171", "85", "false").insert();
new DBpeopleLine(peopleDB, params,  name: "Lena",  probability: 0.2,
        ...values: "48", "75", "25", "152", "68", "false").insert();
new DBpeopleLine(peopleDB, params,  name: "Misha",  probability: 0.2,
        ...values: "35", "5", "25", "180", "60", "true").insert();
new DBpeopleLine(peopleDB, params,  name: "Kostya",  probability: 0.2,
        ...values: "10", "45", "75", "192", "67", "false").insert();
new DBpeopleLine(peopleDB, params,  name: "Natasha",  probability: 0.2,
        ...values: "25", "65", "95", "205", "50", "true").insert();
peopleDB.sqlSelectAllPrint();
```

Рисунок 9 — Объявление свойств объекта

Листинг кода представлен в приложении А. После чего создается таблица «people» и заполняется этими значениям. Пример данных показан в таблице 8.

Таблица 8 — Содержимое таблицы «people»

id	name	hair_l	hair_a	hair_b	height	weight	glasses	probability
1	Petya	53	8	54	170	85	0	0.2
2	Vasya	58	10	60	165	70	0	0.2
3	Masha	30	10	70	150	60	1	0.2
4	Maksim	25	45	52	170	68	0	0.2
5	Dasha	64	10	50	65	67	1	0.2
6	Anna	60	65	54	171	85	0	0.2
7	Lena	48	75	25	152	68	0	0.2
8	Misha	35	5	25	180	60	1	0.2
9	Kostya	10	45	75	192	67	0	0.2
10	Natasha	25	65	95	205	50	1	0.2

Где колонки, которые создаются при любых ситуациях:

— id;

— name;

— probability.

Остальные колонки, которые создаются на основе содержимого таблицы «params»:

— hair_l;

— hair_a;

— hair_b;

— height;

— weight;

— glasses.

После чего запускается простой механизм заполнения показанный на рисунке 10.

```
int time = 1;
for (int z = 0; z < 10; z++) {
    for (DBpeopleLine peopleLine : new DBpeople<DBpeopleLine>(params)
            .getList(peopleDB.sqlGetAll(), peopleDB)) {
        if (Math.random() > 0.7) {
            for (int j = 0; j < 5 + (Math.random() * 1000); j++) {
                Map<String, Boolean> booleans = new HashMap<>();
                Map<String, Double> numbers = new HashMap<>();
                for (DBparamsLine param : params.getList()) {
                    if (param.isState()) {
                        booleans.put(param.getName(),
                                peopleLine
                                .getBooleans().get(param.getName()));
                    } else {
                        numbers.put(param.getName(),
                                peopleLine
                                .getNumbers().get(param.getName()) + (new Random()
                                .nextGaussian() * 2));
                    }
                }
                new DBtimeLapsLine(timeLaps,
                        params,
                        time: time++ * 10,
                        who: "[" + peopleLine.getId() + "] " + peopleLine.getName(),
                        numbers,
                        booleans,
                        info: null
                ).insert();
            }
        }
    }
}
timeLaps.sqlSelectAllPrint();
```

Рисунок 10 — Создание промежуточных данных.

После чего создается таблица «timelaps» и заполняется значениями, где все численные значение с помощью нормального распределения.

Пример данных показан на рисунке 11.

id	time	who	hair_l	hair_a	hair_b	height	weight	glasses	info
1	10	[3] Masha	28.9717900…	9.07825738…	70.4196653…	150.997669…	58.6158853…	☑	null
2	20	[3] Masha	29.9969429…	8.88164429…	67.0688524…	145.953458…	59.2555099…	☑	null
3	30	[3] Masha	31.8819505…	8.09471885…	70.9900325…	156.865016…	64.8027904…	☑	null
4	40	[3] Masha	34.0400519…	8.65954630…	70.7060260…	149.649144…	59.7519431…	☑	null
5	50	[3] Masha	28.8613496…	12.1486970…	72.3448718…	145.368322…	58.1893239…	☑	null
6	60	[3] Masha	27.9855495…	11.9945120…	72.8194449…	152.229696…	61.5389647…	☑	null
7	70	[3] Masha	31.5483688…	12.7526928…	68.3706248…	150.425349…	60.1989205…	☑	null
8	80	[3] Masha	30.3833846…	9.26121922…	70.4925284…	151.655385…	59.3554254…	☑	null
9	90	[3] Masha	29.6787914…	10.3656881…	71.5697362…	150.624896…	59.9198049…	☑	null
10	100	[3] Masha	31.4172966…	6.32013807…	69.8710494…	149.060331…	59.7885246…	☑	null
11	110	[3] Masha	29.0916614…	6.65007448…	69.3273494…	148.723874…	60.2628074…	☑	null
12	120	[3] Masha	29.1280520…	11.1722890…	75.3706416…	151.242676…	60.7884532…	☑	null
13	130	[3] Masha	28.0149509…	10.9557758…	67.1773385…	151.390948…	61.1059619…	☑	null
14	140	[3] Masha	30.3832683…	14.2308362…	70.9681035…	145.748713…	61.0922987…	☑	null
15	150	[3] Masha	29.0889311…	11.7430476…	67.5364525…	148.594774…	60.1944194…	☑	null
16	160	[3] Masha	29.9270567…	11.5049560…	70.3263776…	150.902174…	60.4096647…	☑	null
17	170	[3] Masha	29.6084435…	7.60404313…	69.0029018…	150.763587…	56.9418319…	☑	null
18	180	[3] Masha	27.8576176…	9.29916402…	70.3080983…	149.691016…	58.4391131…	☑	null
19	190	[3] Masha	30.8626652…	9.41852795…	66.0229974…	150.234684…	64.4068291…	☑	null
20	200	[3] Masha	31.7032315…	9.13013569…	70.0158986…	148.285868…	60.3876922…	☑	null
21	210	[3] Masha	32.2785692…	11.9436023…	68.3187190…	153.107780…	62.5495118…	☑	null
22	220	[3] Masha	31.0711318…	7.36489866…	71.5945826…	150.944444…	54.2815356…	☑	null
23	230	[3] Masha	31.3689711…	6.53293605…	71.1238735…	150.582113…	58.0099801…	☑	null
24	240	[3] Masha	28.9292675…	9.64978891…	72.2089295…	151.462899…	62.2629061…	☑	null
25	250	[3] Masha	29.1728468…	8.83530283…	68.6130085…	150.638655…	61.3679784…	☑	null
26	260	[3] Masha	29.0244557…	8.66407171…	71.3464921…	148.916635…	59.1382613…	☑	null
27	270	[3] Masha	26.9497705…	10.4692128…	69.1219069…	148.683804…	60.3051836…	☑	null
28	280	[3] Masha	30.6185991…	10.4253887…	70.1491518…	152.429531…	58.0251688…	☑	null
29	290	[3] Masha	29.8449464…	9.44496779…	70.4468918…	149.415657…	59.8601475…	☑	null
30	300	[3] Masha	27.9387152…	7.55388182…	74.0785345…	147.615770…	59.2676697…	☑	null
31	310	[3] Masha	31.7669236…	6.90367003…	72.5531204…	149.936038…	60.4657263…	☑	null
32	320	[3] Masha	33.8522546…	9.39456578…	70.3451829…	146.724389…	60.6308523…	☑	null
33	330	[3] Masha	28.7714445…	11.5810579…	71.8207756…	152.416415…	62.9484969…	☑	null
34	340	[4] Maksim	23.2712946…	44.0945005…	52.2201238…	169.227531…	66.2470551…	☐	null
35	350	[4] Maksim	25.1116111…	47.4703185…	49.0727039…	173.278229…	67.5063200…	☐	null
36	360	[4] Maksim	26.9749232…	42.7374659…	53.0544289…	167.209499…	65.5801729…	☐	null
37	370	[4] Maksim	27.5190829…	44.8461695…	50.5017190…	170.037471…	63.8547537…	☐	null
38	380	[4] Maksim	24.0972183…	42.8219964…	48.4730542…	170.641490…	67.2378366…	☐	null
39	390	[4] Maksim	23.3449102…	43.4125807…	50.7878902…	169.188188…	70.8912317…	☐	null
40	400	[4] Maksim	28.1878467…	48.3425623…	48.2449900…	171.153843…	65.9887326…	☐	null
41	410	[4] Maksim	20.7308453…	44.2501264…	50.6160091…	172.023546…	68.8554148…	☐	null
42	420	[4] Maksim	23.1810419…	46.2238892…	52.9978273…	169.496572…	67.9171313…	☐	null
43	430	[4] Maksim	25.1414827…	45.5851654…	52.7013182…	172.363826…	67.4860910…	☐	null
44	440	[4] Maksim	26.9616325…	47.3034122…	53.1300800…	172.650730…	67.1104712…	☐	null

Рисунок 11 — Содержимое таблицы «timelaps»

Где колонки, которые создаются при любых ситуациях:

— id;

— time, момент времени. Для тестирования было взято, что один момент времени равен 10;

— who, чьи данные находятся на наблюдении. Чтобы можно было потом сравнивать с тем, что мы получим в итоге;

— info, здесь будет записываться информация о том, какое состояние

системы приняло на данный момент времени.

Остальные колонки, которые создаются на основе содержимого таблицы «params»:

— hair_l;

— hair_a;

— hair_b;

— height;

— weight;

— glasses.

Для наглядности можно вывести диаграмму, на которой будет показан, какой объект находится в данный момент времени. Диаграмма показана на рисунке 12.

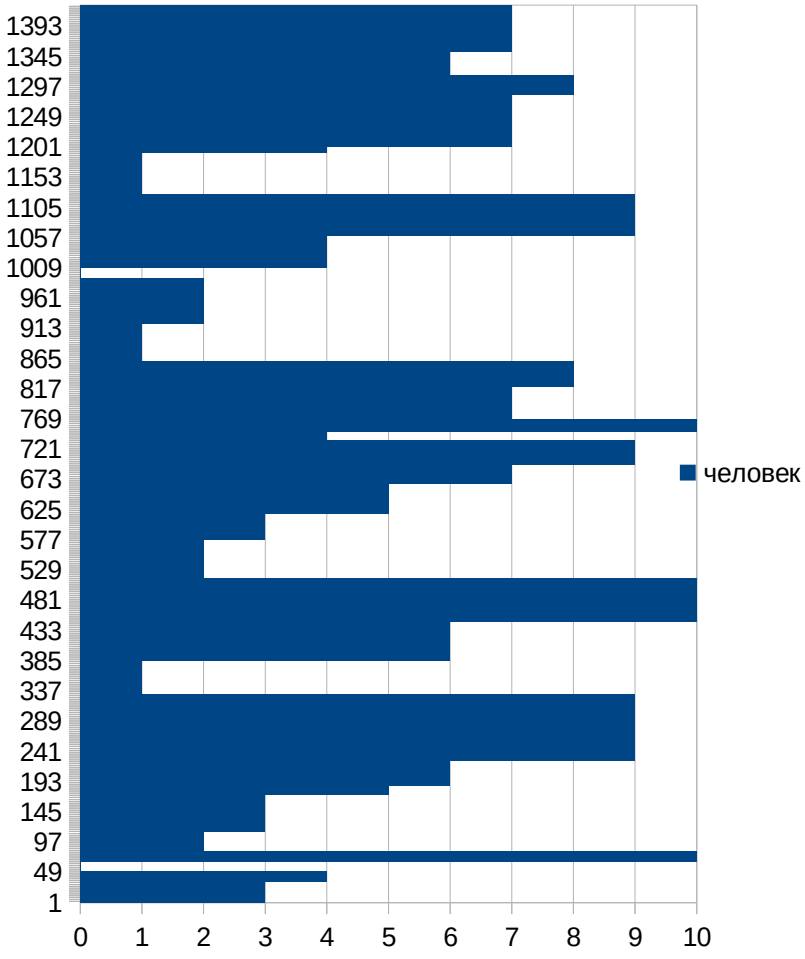

Рисунок 12 — Диаграмма таблицы «timelaps»

Где на шкале оси OX с 1 до 10 показан идентификатор человека, а при 0 записаны небольшие значения, которые символизируют просто шум.

В качестве искомого объекта, возьмем человека с идентификатором 4. Его свойства для анализа были сохранены в таблице «params».

Во входные данные будет передаваться марковская цепь для смены состояния системы. Она была описана с расчетом того, что смена состояния не зависят друг от друга, то есть равно вероятна. Она показана в таблице 9.

Таблица 9 — Марковская цепь смены состояния системы

id	0	1	2
0	0.3333	0.3333	0.3333
1	0.3333	0.3333	0.3333
2	0.3333	0.3333	0.3333

Где у нас следующие состояния:

— 0, система переходит в это состояние, когда на анализ попадает шум;

— 1, система переходит в это состояние, когда она выявляет совпадение с объектом, который ей надо найти;

— 2, система переходит в это состояние, когда она выявляет объект, но он не совпадает с искомым.

Также будут введены следующие ситуации, чтобы оценить качество системы:

1. Перед потоком значений искомого объекта, должен находиться поток шума.

2. Перед потоком значений искомого объекта, должен находиться поток другого объекта.

3. После потока значений искомого объекта, должен находиться поток шума

4. .После потока значений искомого объекта, должен находиться поток

другого объекта.

Благодаря чему будет видно, как быстро система реагирует на смены состояний.

4.2 Результаты и корректировка вычислений

После работы программы данные о том, какие состояния системы были записывались в колонку «info» таблицы «timelaps».

Для простоты работы с данными они были выкачены и сохранены в «.ods» формат. Результат можно увидеть на рисунке 13.

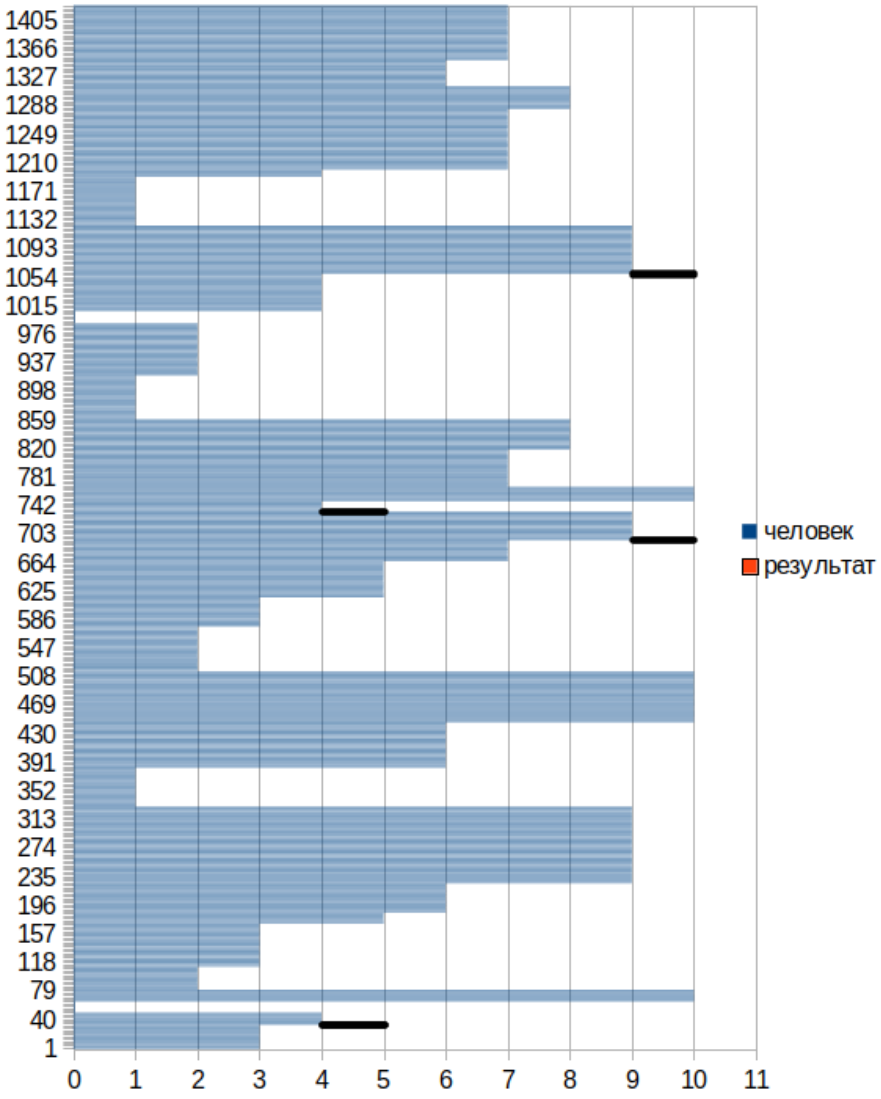

Рисунок 13 — Диаграмма таблицы «timelaps» после анализа

Где черными линиями были обозначены места не соответствий состояний системы. Их было четыре штуки. Выведем их для наглядности на рисунках 14, 15, 16 и 17.

time	who	hair_l	hair_a	hair_b	height	weight	glasses	info	человек	Правдивость
310	[3] Masha	31.76	6.903	72.55	149.9	60.46	1	2	3	ИСТИНА
320	[3] Masha	33.85	9.394	70.34	146.7	60.63	1	2	3	ИСТИНА
330	[3] Masha	28.77	11.58	71.82	152.4	62.94	1	2	3	ИСТИНА
340	[4] Maksim	23.27	44.09	52.22	169.2	66.24	0	2	4	ЛОЖЬ
350	[4] Maksim	25.11	47.47	49.07	173.2	67.50	0	1	4	ИСТИНА
360	[4] Maksim	26.97	42.73	53.05	167.2	65.58	0	1	4	ИСТИНА
370	[4] Maksim	27.51	44.84	50.50	170.0	63.85	0	1	4	ИСТИНА

Рисунок 14 — Первое не соответствие состояние системы

time	who	hair_l	hair_a	hair_b	height	weight	glasses	info	человек	Правдивость
6930	[7] Lena	48.95	75.07	25.82	148.2	66.48	0	2	7	ИСТИНА
6940	[7] Lena	48.00	73.98	23.81	153.0	66.92	0	2	7	ИСТИНА
6950	[7] Lena	48.79	76.86	27.39	152.5	66.63	0	2	7	ИСТИНА
6960	[9] Kostya	11.59	46.48	74.94	192.4	64.97	0	1	9	ЛОЖЬ
6970	[9] Kostya	11.13	43.77	76.13	189.2	69.45	0	2	9	ИСТИНА
6980	[9] Kostya	10.78	47.40	76.59	187.3	70.66	0	2	9	ИСТИНА
6990	[9] Kostya	8.837	43.03	74.71	190.0	66.25	0	2	9	ИСТИНА

Рисунок 15 — Второе не соответствие состояние системы

time	who	hair_l	hair_a	hair_b	height	weight	glasses	info	человек	Правдивость
7320	[9] Kostya	7.500	43.60	74.06	194.4	69.94	0	2	9	ИСТИНА
7330	[9] Kostya	14.56	46.06	78.14	191.5	64.54	0	2	9	ИСТИНА
7340	[9] Kostya	10.64	47.81	75.51	194.4	68.25	0	2	9	ИСТИНА
7350	[4] Maksim	21.75	45.04	49.81	169.3	68.47	0	2	4	ЛОЖЬ
7360	[4] Maksim	22.35	38.88	52.23	168.3	68.58	0	1	4	ИСТИНА
7370	[4] Maksim	26.94	42.99	49.22	171.2	68.34	0	1	4	ИСТИНА
7380	[4] Maksim	23.38	44.70	53.09	170.9	69.31	0	1	4	ИСТИНА

Рисунок 16 — Третье не соответствие состояние системы

time	who	hair_l	hair_a	hair_b	height	weight	glasses	info	человек	Правдивость
10570	[4] Maksim	24.50	44.00	49.44	168.6	63.96	0	1	4	ИСТИНА
10580	[4] Maksim	25.90	47.62	52.88	168.1	64.55	0	1	4	ИСТИНА
10590	[4] Maksim	25.88	41.06	56.67	170.7	69.28	0	1	4	ИСТИНА
10600	[9] Kostya	7.659	45.74	75.87	191.6	69.42	0	1	9	ЛОЖЬ
10610	[9] Kostya	8.237	42.84	72.56	190.1	66.83	0	2	9	ИСТИНА
10620	[9] Kostya	11.24	46.30	78.16	192.9	64.77	0	2	9	ИСТИНА
10630	[9] Kostya	14.20	46.59	78.63	191.2	68.98	0	2	9	ИСТИНА

Рисунок 17 — Четвертое не соответствие состояние системы

Разберем почему возникли эти ошибки по подробнее:

1. На рисунке 14 виден стык потока данных от объекта «Masha» переходящий в наш искомый объект «Maksim». Это произошло потому что значения: «hair_l», «height» и «weight» имели высокую схожесть со значениями нашего искомого объекта, но не достаточную схожесть, так как начиная со второго момента времени, система правильно определила состояние.

2. На рисунке 15 виден стык потока данных от объекта «Lena» переходящий в объект «Kostya». Это произошло потому что значения: «hair_a» и «weight» у объекта «Kostya» имели высокую схожесть со значениями нашего искомого объекта, а также значения: «height» и «weight» у объекта «Lena» имели высокую схожесть со значениями нашего искомого объекта. Что в сумме подкрепило у системы гипотезу, что перед ней искомый объект. Но начиная со второго момента времени, система правильно определила состояние.

3. На рисунке 16 виден стык потока данных от объекта «Kostya» переходящий в наш искомый объект «Maksim». Это произошло потому что значения: «hair_l», «hair_a» и «weight» имели высокую схожесть со значениями нашего искомого объекта, но не достаточную схожесть, так как начиная со второго момента времени, система правильно определила состояние.

4. На рисунке 17 виден стык потока данных от нашего искомого объекта «Maksim», который переходит в объект «Kostya». Это произошло потому что значения: «hair_l», «hair_a» и «weight» имели высокую схожесть со значениями нашего искомого объекта, но не достаточную схожесть, так как начиная со второго момента времени, система правильно определила состояние.

Система в показала очень высокую точность не смотря на эти 4 ошибки. Эти ошибки можно было избежать введя дополнительные биометрические идентификаторы, чтобы уменьшить шанс совпадения.

Вывод к главе 4

1. Используя несколько биометрических идентификаторов с низкой уникальностью, можно в совокупности получить хорошую точность в небольших группах.

2. Существует множество технологий с помощью которых упрощается разработка. Так например для работы с данными использовалась СУБД SQLite, которая позволяла удобно добавлять и изменять данные.

Заключение

Во время выполнение магистерской работы был произведен анализ существующих методов идентификаций. Была выделена актуальность исследуемой темы, определены объект, предмет исследования, цели и задачи работы.

В первой главе данной работы был проведен обзор классификации алгоритмом, в результате чего было принято решение об создании алгоритма на основе Байесовской фильтрации.

В последующих главах уже анализировалось качество биометрических идентификаторов, где была разработана формула качества биометрических идентификаторов под конкретные цели: качество, стоимость и массовость. Также разрабатывалась модель алгоритма с помощью, которого можно было заниматься идентификацией любых субъектов. Которые можно описать численно или параметрически.

В ходе разработки, все вычисления и алгоритмы были рассмотрены непосредственно в работе.

Подводя итоги, было сделано следующее:

1. Подробно были изучены свойства исследуемого субъекта, представленные на рисунке 1. Где были выявлены наиболее полезные признаки под разрабатываемую систему.

2. Модель которая была представлена на рисунке 2 дополнена и сведена после исследования и анализа к практическому применению.

3. Реализована программа, которая может идентифицировать любые субъекты состоящие из множеств описанных в формуле 1.

4. Был получен результат выполнения программы, где после её анализа были разобраны её недостатки, а также выяснили что используя несколько биометрических идентификаторов с низкой уникальностью, можно в совокупности получить хорошую точность в небольших группах.

Список используемой литературы

1. Mikhailov M.A., Volevodz A.G., Sidorenko E.L. Improving the System of Fingerprint Registration // International Scientific and Practical Conference in the State Duma “Improving the System of Fingerprint Registration”. 2016. С. 368-378.

2. Kubitovich S.N. DNK kak nositel informatsii neogranichennogo kruga lits // Vestnik ekonomicheskoy bezopasnosti. 2017. С. 184—188.

3. Edson, A. M. AAMVA Standards Working Group. AAMVA Standard for the Driver License // Identification Card 2000. Technical Report AAMVA DL/ID-2000, The American Association of Motor Vehicles Administrators. 2000. С. 1-7.

4. Jain, A. K. Biometrics: Personal Identification in networked Society. // Kluwer Academic Publishers, Boston, MA. 1999. С. 1-9.

5. Anwar, A. S. Human Ear Recognition Using Geometrical Features Extraction. // International Conference on Communication, Management and Information Technology. 2015. С. 531-537.

6. Исаев, А. Л. Распознование лиц по изображению // Символ науки. 2017. С. 70-76.

7. Карим, О. Построение алгоритма распознавания отпечатков пальцев для систем контроля доступа // Автоматизированные системы управления и приборы автоматики. 2008. С. 116-120.

8. Носов, А. В. Алгоритм распознования жестов рук на основе скелетной модели руки // Сибирский журнал науки и технологий. 2014. С. 62-67.

9. Минакова, Н.Н. Распознавание личности по радужной оболочке глаза с учетом физиологических изменений // Известия Алтайского государственного университета. 2009. С. 129-130.

10. Баранов, Р.П. Идентификация человека по его личной подписи в системах электронного документооборота // Сибирский журнал науки и технологий. 2014. С. 38-43.

11. Меньшаков, П.А. Методика голосовой идентификации на основе нейронных сетей // Доклады Белорусского государственного университета

информатики и радиоэлектроники. 2017. С. 12-18.

12. Шарипов, Р.Р. Система распознования клавиатурного почерка пользователей на основе полигауссового алгоритма // Вестник Казанского государственного энергетического университета. 2016. С. 1-15.

13. Ворона, В.А. Биометрические технологии идентификации в системах контроля и управления доступом // Computational nanotechnology. 2016. С. 224-241.

14. Баша, Н.С. Система выделения подкожного кровеносного рисунка по термографическим изображениям // Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия «Естественные науки». 2012. С. 98-106.

15. Григорьева, М.А. Динамика роста, веса и индекс массы тела реальных поколений Россиян 18-10-1995 годов в возрасте 18-34 лет // Вестник экономики, права и социологии. 2015. С. 198-202.

16. Хомяков, М.Ю. Классификация цвета кожи человека на цветных изображениях// Компьютерная оптика. 2011. С. 1-7.

17. Бовин, Б.Г. Распознавание психологических характеристик личности с помощью признаков почерка // Психопедагогика в правоохранительных органах. 1997. С. 84-85.

18. Борохов, А.Д. Многоосевая классификация татуировок для интегральной оценки проявлений психопатологии личности носителя // Медицинская психология в России. 2018. С. 1-20.

19. Шахтарин, Б.И. Алгоритмы распознавания клеток крови // Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия «Приборостроение» 2015. С. 49-65.

20. Деревцова, С.Н. Инструментальный метод исследования параметров ходьбы людей старших возрастных групп разных соматотипов // Вестник новых медицинских технологий. 2010. С. 181-182.

21. Сесин, Е.М. Системы идентификации личности, основанные на

интеграции нескольких биометрических характеристик человека // Доклады Томского государственного университета систем управления и радиоэлектроники. 2012. С. 175-179.

22. Дятлов, Е.И. Машинное зрение (аналитический обзор) // Математические машины и системы. 2013. С. 1-9.

23. Шахин, Г. Сравнительный анализ библиотек компьютерного зрения // Colloquium-journal. 2019. С 1-3.

24. МВД при помощи камер начнет искать преступников по татуировкам и походке. [Электронный ресурс]: новость / В. Скобелев. URL: https://www.rbc.ru/technology_and_media/24/02/2020/5e4fb5af9a7947cfdfd5e1e3 (дата обращения: 26.02.2020).

25. Большой тур следит за тобой. [Электронный ресурс]: новость / Ю. Тишина. URL: https://www.kommersant.ru/doc/4317311 (дата обращения: 13.04.2020).

26. Попов, В.В. Идентификация личности молекулярно-генетическими методами // Юристъ — Правоведъ. 2018. С 169-174.

27. Чабан, Л.Н. Методы и алгоритмы распознавания образов в автоматизированном дешифрировании данных дистанционного зондирования: учебное пособие. – М.: МИИГАиК, 2016, 94 с.

28. Дж.Ту, Р.Гонсалес. Принципы распознавания образов. – М.: Мир, 1978.

29. Новиков Ф.А. Дискретная математика для программистов. – Спб.: Питер, 2002.

30. Чабан Л.Н. Теория и алгоритмы распознавания образов. М.: МИИГАИК, 2004.

31. Соколова А. И, Конушин А. С. Методы идентификации человека по походке в видео // Труды ИСП РАН. 2019. С 69–82.

32. Тропченко А. А, Тропченко Ф. Ю. Нейросетевые методы идентификации человека по изображению лица // Известия высших учебных

заведений. Приборостроение. 2012. С. 31-36.

33. Семенова М. А, Семёнов В. А. Метод Автоматической Фильтрации При Борьбе Со „Спамом" // Известия высших учебных заведений. Приборостроение. 2009. С. 32-34.

34. Горшков С. А, Кондратенок В. А, Солонар А. С, Крикливый М. В. Сравнительный анализ ошибок измерения калмановского и многогипотезного фильтров с объединением на входе // Доклады Белорусского государственного университета информатики и радиоэлектроники. 2003. С. 1-5.

Приложение А

Листинг кода реализации классов

./src/main/java/ru/binarus/db/storage/DBpeopleLine.java

```
package ru.binarus.db.storage;

import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;
import java.util.StringJoiner;
import ru.binarus.kifd.impl.PeopleLineImpl;
import ru.binarus.db.Connect;

public class DBpeopleLine implements DBinfoLineImpl, PeopleLineImpl {
   private int id ;
   private String  name ;

   private Map<String,Double> numbers;
   private Map<String,Boolean> booleans;

   private double  probability ;
   private DBparams<DBparamsLine> params;
   private Connect connect;

     public DBpeopleLine(Connect connect,  DBparams<DBparamsLine> params, int id, String name,  Map<String, Double> numbers,
      Map<String, Boolean> booleans,  double probability) {
      this.id = id;
      this.name = name;
      this.numbers = numbers;
      this.booleans = booleans;
      this.probability = probability;
      this.params = params;
      this.connect = connect;
   }

      public DBpeopleLine(Connect connect,  DBparams<DBparamsLine> params,String name, Map<String, Double> numbers, Map<String,
      Boolean> booleans, double probability) {
      this.name = name;
      this.numbers = numbers;
      this.booleans = booleans;
      this.probability = probability;
      this.params = params;
      this.connect = connect;
   }

      public DBpeopleLine(Connect connect, DBparams<DBparamsLine> params, String name, double probability,String ... values) {
      DBparamsLine param;
```

```
        int paramsk = 0;
            Map<String,Double> numbers = new HashMap<>();
            Map<String,Boolean> booleans = new HashMap<>();
            for (String value:values) {
               param = params.getList().get(paramsk++);
               if(param.isState()){
                  booleans.put(param.getName(),Boolean.valueOf(value));
               }else{
                  numbers.put(param.getName(),Double.valueOf(value));
               }
            }

            this.name = name;
            this.numbers = numbers;
            this.booleans = booleans;
            this.probability = probability;
            this.params = params;
            this.connect = connect;
         }

         public DBpeopleLine(Connect connect, DBparams params) {
            this.connect = connect;
            this.params = params;
         }

         @Override
         public String getSqlUpdate() {
            StringBuilder update = new StringBuilder();
            for(DBparamsLine param : this.getParams().getList()){
               if(param.isState()){
                                                                        update.append(",
").append(param.getName()).append(this.getBooleans().get(param.getName()));
               }else {
                                                                        update.append(",
").append(param.getName()).append(this.getNumbers().get(param.getName()));
               }
            }
                     return "SET name = '"+this.getName()+"' "+update+" , probability =
"+this.getProbability()+" where id = "+this.getId()+"";
         }

         @Override
         public String getSqlInsert() {
            StringBuilder insertTitle = new StringBuilder();
            StringBuilder insertValue = new StringBuilder();
            for(DBparamsLine param : this.getParams().getList()){
               insertTitle.append(", ").append(param.getName());
               if(param.isState()){
```

```
            insertValue.append(", ").append(this.getBooleans().get(param.getName()));
            }else {
                    insertValue.append(", ").append(this.getNumbers().get(param.getName()));
                }
            }
                return "('name' "+insertTitle+" , probability) VALUES ('"+this.getName()+"'
"+insertValue+" ,"+this.getProbability()+")";
        }

        @Override
        public void update() throws SQLException {
                                                this.getConnect().getStatmt().execute("Update
"+this.getConnect().getConnectInfo().getMainTable() + " " + this.getSqlUpdate());
        }

        @Override
        public void insert() throws SQLException {
                                            this.getConnect().getStatmt().execute("Insert into
"+this.getConnect().getConnectInfo().getMainTable() + " " + this.getSqlInsert());
        }

        public int getId() {
            return id;
        }

        public void setId( int id) {
            this.id = id;
        }

        public String getName() {
            return name;
        }

        public void setName( String name) {
            this.name = name;
        }

        public Map<String, Double> getNumbers() {
            return numbers;
        }

        public void setNumbers( Map<String, Double> numbers) {
            this.numbers = numbers;
        }

        public Map<String, Boolean> getBooleans() {
            return booleans;
        }
```

```
    public void setBooleans( Map<String, Boolean> booleans) {
this.booleans = booleans;
    }

    public double getProbability() {
        return probability;
    }

    public void setProbability( double probability) {
        this.probability = probability;
    }

    public DBparams<DBparamsLine> getParams() {
        return params;
    }

    public void setParams( DBparams<DBparamsLine> params) {
        this.params = params;
    }

    public Connect getConnect() {
        return connect;
    }

    public void setConnect(final Connect connect) {
        this.connect = connect;
    }

    @Override
    public String toString() {
        return new StringJoiner(", ", DBpeopleLine.class.getSimpleName() + "[", "]")
            .add("id=" + id)
            .add("name='" + name + "'")
            .add("numbers=" + numbers)
            .add("booleans=" + booleans)
            .add("probability=" + probability)
            .toString();
    }
}
-----------------
./src/main/java/ru/binarus/db/storage/DBpeople.java
package ru.binarus.db.storage;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

```
import java.util.Map;
import ru.binarus.db.Connect;
public class DBpeople<TypeLine> implements DBinfoImpl {

    private String driver = "org.sqlite.JDBC";
    private String connect = "jdbc:sqlite:core/src/main/resources/sqlLite.db";
    private List<DBpeopleLine> result;
    private String mainTable = "people";
    private DBparams<DBparamsLine> params;

    public DBpeople() {

    }

    @Override
    public void setResult(ResultSet resSet, Connect connect) throws SQLException {
        List<DBpeopleLine> result = new ArrayList<>();
        while(resSet.next())
        {
            int id = resSet.getInt("id");
            String  name = resSet.getString("name");

            Map<String,Double> numbers = new HashMap<>();
            Map<String,Boolean> booleans = new HashMap<>();
            for(DBparamsLine param : this.getParams().getList()){
                if(param.isState()){
                    booleans.put(param.getName(),resSet.getBoolean(param.getName()));
                }else{
                    numbers.put(param.getName(),resSet.getDouble(param.getName()));
                }
            }
            double  probability = resSet.getDouble("probability");
                                                                    result.add(new
DBpeopleLine(connect,this.getParams(),id,name,numbers,booleans,probability));
        }
        this.result = result;
    }

    @Override
    public String getSqlGenerateMainTable() {
            StringBuilder sql = new StringBuilder("CREATE TABLE if not exists '" +
this.getMainTable() + "' ('id' INTEGER PRIMARY KEY AUTOINCREMENT,'name' text");

        for(DBparamsLine param : this.getParams().getList()){
            if(param.isState()){
                sql.append(",'").append(param.getName()).append("' boolean");
            }else{
                sql.append(",'").append(param.getName()).append("' int");
```

```
            }
            }
            sql.append(",'probability' int);");
            return sql.toString();
        }

        public DBpeople(DBparams params) {
            this.params = params;
        }

        @Override
        public List<TypeLine> getResult() {
            return (List<TypeLine>) result;
        }

        @Override
        public String getMainTable() {
            return mainTable;
        }


            public List<DBpeopleLine> getList(ResultSet resSet, Connect connect) throws
SQLException {
            this.setResult(resSet,connect);
            return result;
        }

        public List<DBpeopleLine> getList() {
            return result;
        }

        @Override
        public String getDriver() {
            return driver;
        }

        @Override
        public String getConnect() {
            return connect;
        }

        @Override
        public void printResult() {
            for (DBinfoLineImpl i : this.getList()) {
                System.out.println(i);
            }
        }
```

```
    public void setDriver( String driver) {
        this.driver = driver;
    }

    public void setConnect( String connect) {
        this.connect = connect;
    }

    public void setResult( List<DBpeopleLine> result) {
        this.result = result;
    }

    public void setMainTable( String mainTable) {
        this.mainTable = mainTable;
    }

    public DBparams<DBparamsLine> getParams() {
        return params;
    }

    public void setParams(DBparams<DBparamsLine> params) {
        this.params = params;
    }
}
-----------------
./src/main/java/ru/binarus/db/storage/DBinfoLineImpl.java
package ru.binarus.db.storage;

public interface DBinfoLineImpl {
    String toString();
    String getSqlUpdate();
    String getSqlInsert();
}
-----------------
./src/main/java/ru/binarus/db/storage/DBtimeLaps.java
package ru.binarus.db.storage;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import ru.binarus.db.Connect;

public class DBtimeLaps<TypeLine> implements DBinfoImpl {
    private String driver = "org.sqlite.JDBC";
    private String connect = "jdbc:sqlite:core/src/main/resources/sqlLite.db";
```

```
        private List<DBtimeLapsLine> result;
        private String mainTable = "timelaps";
        private DBparams<DBparamsLine> params;

        public DBtimeLaps() {

        }

        @Override
        public void setResult(ResultSet resSet, Connect connect) throws SQLException {
           List<DBtimeLapsLine> result = new ArrayList<>();
           while(resSet.next())
           {
              int id = resSet.getInt("id");
              int time = resSet.getInt("time");
              String  who = resSet.getString("who");

              Map<String,Double> numbers = new HashMap<>();
              Map<String,Boolean> booleans = new HashMap<>();
              for(DBparamsLine param : this.getParams().getList()){
                 if(param.isState()){
                    booleans.put(param.getName(),resSet.getBoolean(param.getName()));
                 }else{
                    numbers.put(param.getName(),resSet.getDouble(param.getName()));
                 }
              }
              String  info = resSet.getString("info");
                                                                    result.add(new
DBtimeLapsLine(connect,this.getParams(),id,time,who,numbers,booleans,info));
           }
           this.result = result;
        }

        @Override
        public String getSqlGenerateMainTable() {
                 StringBuilder sql = new StringBuilder("CREATE TABLE if not exists '" +
this.getMainTable() + "' ('id' INTEGER PRIMARY KEY AUTOINCREMENT, time int, 'who'
text");

           for(DBparamsLine param : this.getParams().getList()){
              if(param.isState()){
                 sql.append(",'").append(param.getName()).append("' boolean");
              }else{
                 sql.append(",'").append(param.getName()).append("' int");
              }
           }

           sql.append(",'info' text);");
```

```
        return sql.toString();
    }

    @Override
    public List<TypeLine> getResult() {
        return (List<TypeLine>) result;
    }

    public DBtimeLaps( DBparams params) {
        this.params = params;
    }

    public void setDriver( String driver) {
        this.driver = driver;
    }

    public void setConnect( String connect) {
        this.connect = connect;
    }

    public void setResult( List<DBtimeLapsLine> result) {
        this.result = result;
    }

    public void setMainTable( String mainTable) {
        this.mainTable = mainTable;
    }

    public DBparams<DBparamsLine> getParams() {
        return params;
    }

    public void setParams( DBparams<DBparamsLine> params) {
        this.params = params;
    }

     public List<DBtimeLapsLine> getList(ResultSet resSet, Connect connect) throws
SQLException {
        this.setResult(resSet,connect);
        return result;
    }

    @Override
    public String getMainTable() {
        return mainTable;
    }

    public List<DBtimeLapsLine> getList() {
```

```
        return result;
    }

    @Override
    public String getDriver() {
        return driver;
    }

    @Override
    public String getConnect() {
        return connect;
    }

    @Override
    public void printResult() throws SQLException {
        for (DBinfoLineImpl i : this.getList()) {
            System.out.println(i);
        }
    }
}
-----------------
./src/main/java/ru/binarus/db/storage/DBparams.java
package ru.binarus.db.storage;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import ru.binarus.db.Connect;

public final class DBparams<TypeLine> implements DBinfoImpl {

    private String driver = "org.sqlite.JDBC";
    private String connect = "jdbc:sqlite:core/src/main/resources/sqlLite.db";
    private List<DBparamsLine> result;
    private String mainTable = "params";
        private String sqlGenerateMainTable = "CREATE TABLE if not exists
'"+this.getMainTable()+"' ('id' INTEGER PRIMARY KEY AUTOINCREMENT, 'name' text,
'expectedValue' int, 'dispersion' int, 'state' boolean default false);";

    @Override
    public void setResult(ResultSet resSet, Connect connect) throws SQLException {
        List<DBparamsLine> result = new ArrayList<>();
        while(resSet.next())
        {
            int id = resSet.getInt("id");
            String  name = resSet.getString("name");
            double  expectedValue = resSet.getDouble("expectedValue");
```

```
            double  dispersion = resSet.getDouble("dispersion");
            boolean  state = resSet.getBoolean("state");
            result.add(new DBparamsLine(connect,id,name,expectedValue,dispersion,state));
        }
        this.result = result;
    }

    @Override
    public List<TypeLine> getResult() throws SQLException  {
        return (List<TypeLine>) result;
    }

    public List<DBparamsLine> getList() {
        return result;
    }

    @Override
    public String getMainTable() {
        return mainTable;
    }

    @Override
    public String getSqlGenerateMainTable() {
        return this.sqlGenerateMainTable;
    }

        public List<DBparamsLine> getList(ResultSet resSet, Connect connect) throws
SQLException {
        this.setResult(resSet,connect);
        return result;
    }

    @Override
    public String getDriver() {
        return driver;
    }

    @Override
    public String getConnect() {
        return connect;
    }

    @Override
    public void printResult() throws SQLException {
        for (DBinfoLineImpl i : this.getList()) {
            System.out.println(i);
        }
    }
```

```
}
-----------------
./src/main/java/ru/binarus/db/storage/DBtimeLapsLine.java
package ru.binarus.db.storage;

import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;
import java.util.StringJoiner;
import ru.binarus.kifd.impl.TimeLapsLineImpl;
import ru.binarus.db.Connect;

public class DBtimeLapsLine implements DBinfoLineImpl, TimeLapsLineImpl {

   private int id ;
   private double time;
   private String  who ;
   private Map<String,Double> numbers;
   private Map<String,Boolean> booleans;
   private String  info ;
   private DBparams<DBparamsLine> params;
   private Connect connect;

   public DBtimeLapsLine(Connect connect,
      DBparams<DBparamsLine> params, int id,  double time,  String who,  Map<String,
      Double> numbers,  Map<String, Boolean> booleans,  String info) {
      this.id = id;
      this.time = time;
      this.who = who;
      this.numbers = numbers;
      this.booleans = booleans;
      this.info = info;
      this.params = params;
      this.connect = connect;
   }

    public DBtimeLapsLine(Connect connect,  DBparams<DBparamsLine> params, double
time,  String who,  Map<String, Double> numbers,
       Map<String, Boolean> booleans,  String info) {
      this.time = time;
      this.who = who;
      this.numbers = numbers;
      this.booleans = booleans;
      this.info = info;
      this.params = params;
      this.connect = connect;
   }
```

```
            public DBtimeLapsLine(Connect connect,DBparams<DBparamsLine> params,double
time, String who, String info,  String...values) {
            DBparamsLine param;
            int paramsk = 0;
            Map<String,Double> numbers = new HashMap<>();
            Map<String,Boolean> booleans = new HashMap<>();
            for (String value:values) {
               param = params.getList().get(paramsk++);
               if(param.isState()){
                  booleans.put(param.getName(),Boolean.valueOf(value));
               }else{
                  numbers.put(param.getName(),Double.valueOf(value));
               }
            }

            this.time = time;
            this.who = who;
            this.numbers = numbers;
            this.booleans = booleans;
            this.info = info;
            this.params = params;
            this.connect = connect;
         }

         public DBtimeLapsLine(Connect connect, DBparams params) {
            this.params = params;
            this.connect = connect;
         }

         @Override
         public String getSqlUpdate() {
            StringBuilder update = new StringBuilder();
            for(DBparamsLine param : this.getParams().getList()){
               if(param.isState()){
                                                                        update.append(",
").append(param.getName()).append("=").append(this.getBooleans().get(param.getName()));
               }else {
                                                                        update.append(",
").append(param.getName()).append("=").append(this.getNumbers().get(param.getName()));
               }
            }
               return "SET who = '"+this.getWho()+"' "+update+" , time = "+getTime()+", info =
'"+this.getInfo()+"' where id = '"+this.getId()+"' ";
         }

         @Override
         public String getSqlInsert() {
            StringBuilder insertTitle = new StringBuilder();
```

```
            StringBuilder insertValue = new StringBuilder();
            for(DBparamsLine param : this.getParams().getList()){
                insertTitle.append(", ").append(param.getName());
                if(param.isState()){
                    insertValue.append(", ").append(this.getBooleans().get(param.getName()));
                }else {
                    insertValue.append(", ").append(this.getNumbers().get(param.getName()));
                }
            }
                return "('who', 'time' "+insertTitle+" , info) VALUES ('"+this.getWho()+"',
"+this.getTime()+" "+insertValue+" , '"+this.getInfo()+"')";
        }

        @Override
        public void update() throws SQLException {
                                                this.getConnect().getStatmt().execute("Update
"+this.getConnect().getConnectInfo().getMainTable() + " " + this.getSqlUpdate());
        }

        @Override
        public void insert() throws SQLException {
                                                this.getConnect().getStatmt().execute("Insert into
"+this.getConnect().getConnectInfo().getMainTable() + " " + this.getSqlInsert());
        }

        @Override
        public String toString() {
            return new StringJoiner(", ", DBtimeLapsLine.class.getSimpleName() + "[", "]")
                .add("id=" + id)
                .add("time=" + time)
                .add("who='" + who + "'")
                .add("numbers=" + numbers)
                .add("booleans=" + booleans)
                .add("info='" + info + "'")
                .toString();
        }

        public int getId() {
            return id;
        }

        public void setId( int id) {
            this.id = id;
        }

        public String getWho() {
            return who;
        }
```

```
public void setWho( String who) {
   this.who = who;
}

public Map<String, Double> getNumbers() {
   return numbers;
}

public void setNumbers( Map<String, Double> numbers) {
   this.numbers = numbers;
}

public Map<String, Boolean> getBooleans() {
   return booleans;
}

public void setBooleans( Map<String, Boolean> booleans) {
   this.booleans = booleans;
}

public String getInfo() {
   return info;
}

@Override
public void setInfo(String info) {
   this.info = info;
}

public DBparams<DBparamsLine> getParams() {
   return params;
}

public void setParams( DBparams<DBparamsLine> params) {
   this.params = params;
}

public double getTime() {
   return time;
}


public void setTime( double time) {
   this.time = time;
}

public Connect getConnect() {
   return connect;
```

```
    }

    public void setConnect(final Connect connect) {
        this.connect = connect;
    }
}
----------------
./src/main/java/ru/binarus/db/storage/DBparamsLine.java
package ru.binarus.db.storage;

import java.sql.SQLException;
import java.util.StringJoiner;
import ru.binarus.kifd.impl.ParamsLineImpl;
import ru.binarus.db.Connect;

public class DBparamsLine implements DBinfoLineImpl, ParamsLineImpl {
    private int id;
    private String  name;
    private double expectedValue;
    private double dispersion;
    private boolean state = false;
    private Connect connect;

    @Override
    public String getSqlUpdate() {
        return "SET name = '"+this.getName()+"', expectedValue = "+this.getExpectedValue()
+", dispersion = "+this.getDispersion()+", state = "+this.isState()+" where id = "+this.getId()+";";
    }

    @Override
    public String getSqlInsert() {
        return "('name', 'expectedValue', 'dispersion','state') VALUES ('"+this.getName()+"',
"+this.getExpectedValue()+", "+this.getDispersion()+","+this.isState()+");";
    }

    public DBparamsLine(Connect connect, int id,  String name,  double expectedValue,
double dispersion) {
        this.id = id;
        this.name = name;
        this.expectedValue = expectedValue;
        this.dispersion = dispersion;
        this.connect = connect;
    }

    public DBparamsLine(Connect connect,int id, String name) {
        this.id = id;
        this.name = name;
        this.state = true;
```

```
        this.connect = connect;
    }
    public DBparamsLine(Connect connect,String name) {
        this.name = name;
        this.state = true;
        this.connect = connect;
    }

     public DBparamsLine(Connect connect, String name,  double expectedValue,  double
dispersion) {
        this.name = name;
        this.expectedValue = expectedValue;
        this.dispersion = dispersion;
        this.connect = connect;
    }

     public DBparamsLine(Connect connect, int id,  String name,  double expectedValue,
double dispersion
        ,  boolean state) {
        this.id = id;
        this.name = name;
        this.expectedValue = expectedValue;
        this.dispersion = dispersion;
        this.state = state;
        this.connect = connect;
    }

    public int getId() {
        return id;
    }

    public void setId( int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName( String name) {
        this.name = name;
    }

    public double getExpectedValue() {
        return expectedValue;
    }

    public void setExpectedValue( double expectedValue) {
```

```
        this.expectedValue = expectedValue;
    }

    public double getDispersion() {
        return dispersion;
    }

    public void setDispersion( double dispersion) {
        this.dispersion = dispersion;
    }

    public boolean isState() {
        return state;
    }

    public void setState( boolean state) {
        this.state = state;
    }

    @Override
    public void update() throws SQLException {
                                                this.getConnect().getStatmt().execute("Update
"+this.getConnect().getConnectInfo().getMainTable() + " " + this.getSqlUpdate());
    }

    @Override
    public void insert() throws SQLException {
                                            this.getConnect().getStatmt().execute("Insert into
"+this.getConnect().getConnectInfo().getMainTable() + " " + this.getSqlInsert());
    }

    @Override
    public String toString() {
        if(this.isState())
        return new StringJoiner(", ", DBparamsLine.class.getSimpleName() + "[", "]")
            .add("id=" + id)
            .add("name='" + name + "'")
            .toString();
        else
            return new StringJoiner(", ", DBparamsLine.class.getSimpleName() + "[", "]")
                .add("id=" + id)
                .add("name='" + name + "'")
                .add("expectedValue=" + expectedValue)
                .add("dispersion=" + dispersion)
                .toString();
    }

    public Connect getConnect() {
```

```
        return connect;
    }

    public void setConnect(final Connect connect) {
        this.connect = connect;
    }
}
-----------------
./src/main/java/ru/binarus/db/storage/DBinfoImpl.java
package ru.binarus.db.storage;

import java.sql.ResultSet;
import java.sql.SQLException;
import ru.binarus.db.Connect;
import java.util.List;
public interface DBinfoImpl<TypeLine> {
    String getDriver();
    String getConnect();
    String getMainTable();
    String getSqlGenerateMainTable();
    void setResult(ResultSet resSet, Connect connect) throws SQLException;
    List<TypeLine> getResult() throws SQLException;
    void printResult() throws SQLException;
}
-----------------
./src/main/java/ru/binarus/db/DataBase.java
package ru.binarus.db;

import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;
import ru.binarus.db.storage.DBparams;
import ru.binarus.db.storage.DBparamsLine;
import ru.binarus.db.storage.DBpeople;
import ru.binarus.db.storage.DBpeopleLine;
import ru.binarus.db.storage.DBtimeLaps;
import ru.binarus.db.storage.DBtimeLapsLine;

public class DataBase {

    public static void main(String[] args) throws ClassNotFoundException, SQLException {
        //=======================================
        Connect paramsDB = new Connect(new DBparams<DBparamsLine>());
        paramsDB.connect();
        paramsDB.createMainTable();
        new DBparamsLine(paramsDB,"hair_l",64,2).insert();
```

```
new DBparamsLine(paramsDB,"hair_a",10,2).insert();
new DBparamsLine(paramsDB,"hair_b",50,2).insert();
new DBparamsLine(paramsDB,"height",165,2).insert();
new DBparamsLine(paramsDB,"weight",67,2).insert();
new DBparamsLine(paramsDB,"glasses").insert();
paramsDB.sqlSelectAllPrint();

DBparams<DBparamsLine> params = new DBparams<DBparamsLine>();
params.setResult(paramsDB.sqlGetAll(),paramsDB);

//=======================================
Connect peopleDB = new Connect(new DBpeople<DBpeopleLine>(params));
peopleDB.connect();
peopleDB.createMainTable();
new DBpeopleLine(peopleDB, params, "Petya", 0.2,
      "53", "8", "54", "170", "85", "false").insert();
new DBpeopleLine(peopleDB, params, "Vasya", 0.2,
      "58", "10", "60", "165", "70", "false").insert();
new DBpeopleLine(peopleDB, params, "Masha", 0.2,
      "30", "10", "70", "150", "60", "true").insert();
new DBpeopleLine(peopleDB, params, "Maksim", 0.2,
      "25", "45", "52", "170", "68", "false").insert();
new DBpeopleLine(peopleDB, params, "Dasha", 0.2,
      "64", "10", "50", "165", "67", "true").insert();
new DBpeopleLine(peopleDB, params, "Anna", 0.2,
      "60", "65", "54", "171", "85", "false").insert();
new DBpeopleLine(peopleDB, params, "Lena", 0.2,
      "48", "75", "25", "152", "68", "false").insert();
new DBpeopleLine(peopleDB, params, "Misha", 0.2,
      "35", "5", "25", "180", "60", "true").insert();
new DBpeopleLine(peopleDB, params, "Kostya", 0.2,
      "10", "45", "75", "192", "67", "false").insert();
new DBpeopleLine(peopleDB, params, "Natasha", 0.2,
      "25", "65", "95", "205", "50", "true").insert();
peopleDB.sqlSelectAllPrint();


//=======================================
Connect timeLaps = new Connect(new DBtimeLaps<DBtimeLapsLine>(params));
timeLaps.connect();
timeLaps.createMainTable();
int time = 1;
for (int z = 0; z < 10; z++) {
   for (DBpeopleLine peopleLine : new DBpeople<DBpeopleLine>(params)
         .getList(peopleDB.sqlGetAll(), peopleDB)) {
      if (Math.random() > 0.7) {
         for (int j = 0; j < 5 + (Math.random() * 1000); j++) {
            Map<String, Boolean> booleans = new HashMap<>();
```

```
                    Map<String, Double> numbers = new HashMap<>();
                    for (DBparamsLine param : params.getList()) {
                        if (param.isState()) {
                            booleans.put(param.getName(),
                                    peopleLine
                                    .getBooleans().get(param.getName()));
                        } else {
                            numbers.put(param.getName(),
                                    peopleLine
                                    .getNumbers().get(param.getName()) + (new Random()
                                    .nextGaussian() * 2));
                        }
                    }
                    new DBtimeLapsLine(timeLaps,
                            params,
                            time++ * 10,
                            "[" + peopleLine.getId() + "] " + peopleLine.getName(),
                            numbers,
                            booleans,
                            null
                    ).insert();
                }
            }
        }
    }
    timeLaps.sqlSelectAllPrint();

    //=======================================
    paramsDB.closeDB();
    peopleDB.closeDB();
    timeLaps.closeDB();

  }
}
-----------------
./src/main/java/ru/binarus/db/Connect.java
package ru.binarus.db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import ru.binarus.kifd.impl.Line;
import ru.binarus.db.storage.DBinfoImpl;
import ru.binarus.db.storage.DBinfoLineImpl;
import ru.binarus.kifd.impl.ResourceImpl;
```

```
public class Connect<TypeResult extends Line> implements ResourceImpl {
    private Connection conn;
    private DBinfoImpl connectInfo;
    private Statement statmt;

    private String driver;
    private String connect;
    private Integer lineId = 1;

    public Connect(DBinfoImpl connect) {
        this.connectInfo = connect;
        this.driver = connect.getDriver();
        this.connect = connect.getConnect();
    }

    public void connect() throws ClassNotFoundException, SQLException
    {
        conn = null;
        Class.forName(this.driver);
        conn = DriverManager.getConnection(this.connect);
        statmt = conn.createStatement();
        System.out.println("База Подключена: "+this.connect);
    }

    public void sqlExecute(String sql) throws SQLException
    {
        statmt.execute(sql);
    }

    public void sqlSelectPrint(String sql) throws ClassNotFoundException, SQLException
    {
        this.connectInfo.setResult(statmt.executeQuery(sql),this);
        this.connectInfo.printResult();
    }

    public void sqlSelectAllPrint() throws ClassNotFoundException, SQLException
    {
                    this.connectInfo.setResult(statmt.executeQuery("SELECT * FROM
"+this.connectInfo.getMainTable()),this);
        this.connectInfo.printResult();
    }

        public void sqlInsert(DBinfoLineImpl inserClass) throws ClassNotFoundException,
SQLException
    {
                statmt.execute("Insert into "+this.connectInfo.getMainTable() + " " +
inserClass.getSqlInsert());
    }
```

```
            public void sqlUpdate(DBinfoLineImpl inserClass) throws ClassNotFoundException, SQLException
            {
                        statmt.execute("Update  "+this.connectInfo.getMainTable() + " " + inserClass.getSqlUpdate());
            }

            @Override
            public TypeResult getNext() throws SQLException {
                this.getConnectInfo().setResult(statmt.executeQuery(String.format("select * from %s where id = %d " ,this.connectInfo.getMainTable(), this.getLineId())), this );
                if (!this.getConnectInfo().getResult().isEmpty()){
                    this.setLineId(((TypeResult) (this.getConnectInfo().getResult().get(0))).getId() + 1 );
                    return (TypeResult) this.getConnectInfo().getResult().get(0);
                }else{
                    this.getConnectInfo().setResult(statmt.executeQuery(String.format("select * from %s where id > %d " ,this.connectInfo.getMainTable(), this.getLineId())), this );
                    if (!this.getConnectInfo().getResult().isEmpty()){
                        this.setLineId(((TypeResult) (this.getConnectInfo().getResult().get(0))).getId() + 1 );
                        return (TypeResult) this.getConnectInfo().getResult().get(0);
                    }else{
                        return null;
                    }
                }
            };

            @Override
            public boolean hasNext() throws SQLException {
                this.getConnectInfo().setResult(statmt.executeQuery(String.format("select * from %s where id = %d " ,this.connectInfo.getMainTable(), this.getLineId())), this );
                if (!this.getConnectInfo().getResult().isEmpty()){
                    return true;
                }else{
                    this.getConnectInfo().setResult(statmt.executeQuery(String.format("select * from %s where id > %d " ,this.connectInfo.getMainTable(), this.getLineId())), this );
                    if (!this.getConnectInfo().getResult().isEmpty()){
                        return true;
                    }else{
                        return false;
                    }
                }
            }

            public void reset(){
                this.setLineId(1);
            };
```

```
public ResultSet sqlSelect(String sql) throws ClassNotFoundException, SQLException
{
   return statmt.executeQuery(sql);
}

public ResultSet sqlGetAll() throws ClassNotFoundException, SQLException
{
   return statmt.executeQuery("SELECT * FROM "+this.connectInfo.getMainTable());
}

public void createMainTable() throws SQLException {
   this.sqlExecute("DROP TABLE IF EXISTS "+this.connectInfo.getMainTable()+";");
   this.sqlExecute(this.connectInfo.getSqlGenerateMainTable());
}

// --------Закрытие--------
public void closeDB() throws ClassNotFoundException, SQLException
{
   conn.close();
   statmt.close();
}


public Connection getConn() {
   return conn;
}

public void setConn(final Connection conn) {
   this.conn = conn;
}

public DBinfoImpl getConnectInfo() {
   return connectInfo;
}

public void setConnectInfo(final DBinfoImpl connectInfo) {
   this.connectInfo = connectInfo;
}

public Statement getStatmt() {
   return statmt;
}

public void setStatmt(final Statement statmt) {
   this.statmt = statmt;
}

public String getDriver() {
```

```
        return driver;
    }

    public void setDriver(final String driver) {
        this.driver = driver;
    }

    public String getConnect() {
        return connect;
    }

    public void setConnect(final String connect) {
        this.connect = connect;
    }

    public Integer getLineId() {
        return lineId;
    }

    public void setLineId(final Integer lineId) {
        this.lineId = lineId;
    }

}
///home/binar/IdeaProjects/mma-master/core/src/main/resources/sqlLite.db----------
./src/main/java/ru/binarus/Main.java
package ru.binarus;
import java.sql.SQLException;
import ru.binarus.kifd.KIFD;
import ru.binarus.db.Connect;
import ru.binarus.db.storage.DBparams;
import ru.binarus.db.storage.DBparamsLine;
import ru.binarus.db.storage.DBpeople;
import ru.binarus.db.storage.DBpeopleLine;
import ru.binarus.db.storage.DBtimeLaps;
import ru.binarus.db.storage.DBtimeLapsLine;

public class Main {

    public static void main(String[] args) throws SQLException, ClassNotFoundException {

                        Connect<DBparamsLine> paramsDB = new Connect<>(new
DBparams<DBparamsLine>());
        paramsDB.connect();
        DBparams<DBparamsLine> params = new DBparams<DBparamsLine>();
        params.setResult(paramsDB.sqlGetAll(),paramsDB);

        Connect<DBpeopleLine> peopleDB = new Connect<>(new
```

```
DBpeople<DBpeopleLine>(params));
        peopleDB.connect();

                                Connect<DBtimeLapsLine> timeLaps = new Connect<>(new
DBtimeLaps<DBtimeLapsLine>(params));
        timeLaps.connect();

        new KIFD(paramsDB,peopleDB,timeLaps);

        paramsDB.closeDB();
        peopleDB.closeDB();
        timeLaps.closeDB();
//        System.out.println(new Square((x)->(x)));
//        System.out.println("Hello World!");
    }
}
-----------------
./src/main/java/ru/binarus/math/functions/FunctionXplus.java
package ru.binarus.math.functions;

/**
 * .
 *
 * @since 0.1.
 */
public class FunctionXplus implements FunctionX{
    private FunctionX function;

    public FunctionXplus(final FunctionX function) {
        this.function = function;
    }

    @Override
    public double get(final double x) {
        return this.function.get(x);
    }

    public double getAbs(final double x) {
        return Math.abs(this.function.get(x));
    }
}
-----------------
./src/main/java/ru/binarus/math/functions/FunctionX.java
package ru.binarus.math.functions;

/**
 * .
 *
```

```
 * @since 0.1.
 */
public interface FunctionX {
   double get(double x);
}
----------------
./src/main/java/ru/binarus/math/integral/Square.java
package ru.binarus.math.integral;

import ru.binarus.math.functions.FunctionX;
import ru.binarus.math.functions.FunctionXplus;

public class Square implements Integral{
   private double leftBorder;
   private double rightBorder;
   private int clust;
   private double sum = 0;
   private FunctionXplus function;

   public Square(final double leftBorder, final double rightBorder, final int clust,
      final FunctionX function) {
      this.leftBorder = leftBorder;
      this.rightBorder = rightBorder;
      this.clust = clust;
      this.function = new FunctionXplus(function);
   }

   public Square(final FunctionX function) {
      this(-1000,1000,2000,function);
   }
   public Square( final int clust,final FunctionX function) {
      this(-1000,1000,clust,function);
   }

   public double get(){
      double shag = ( this.rightBorder - this.leftBorder ) / this.clust;
      for (int i = 1; i < this.clust - 1; i++){
         this.sum = this.sum + this.function.getAbs( shag*i+this.leftBorder );
      }
                              this.sum = this.sum + (this.function.getAbs(this.leftBorder)
+this.function.getAbs(this.rightBorder))/2;
      this.sum =shag * sum;
      return sum;
   }

   @Override
   public String toString() {
      return String.valueOf(this.get());
```

```
    }
}
-----------------
./src/main/java/ru/binarus/math/integral/Integral.java
package ru.binarus.math.integral;

public interface Integral {
    double get();
}
-----------------
./src/main/java/ru/binarus/math/ArrayBooleanList.java
package ru.binarus.math;

import java.util.ArrayList;

public class ArrayBooleanList extends ArrayList<Boolean> {

}
-----------------
./src/main/java/ru/binarus/math/ArrayDoubleList.java
package ru.binarus.math;

import java.util.ArrayList;
import java.lang.Math;
public class ArrayDoubleList extends ArrayList<Double> {

    public ArrayDoubleList(Integer count_coordinate) {
        super(count_coordinate);
    }

    public void multiplyByNumber(Double number){
        for(int i = 0; i < size();i++){
            set(i,get(i) * number);
        };
    }

    public void multiplyByMatrix(ArrayDoubleList matrix){
        for(int i = 0; i < size();i++){
            set(i,get(i) * matrix.get(i));
        }
    }

    public void power(Double number){
        for(int i = 0; i < size();i++){
            set(i,Math.pow(get(i),number));
        }
```

```
    }


}
----------------
./src/main/java/ru/binarus/kifd/KIFD.java
package ru.binarus.kifd;

import java.sql.SQLException;
import java.util.ArrayList;
import com.oracle.math.Run;
import com.sun.org.apache.xalan.internal.xsltc.dom.KeyIndex;
import ru.binarus.kifd.impl.ResourceImpl;
import ru.binarus.kifd.impl.ParamsLineImpl;
import ru.binarus.kifd.impl.PeopleLineImpl;
import ru.binarus.kifd.impl.TimeLapsLineImpl;
import ru.binarus.kifd.math.Kit;
import ru.binarus.math.ArrayBooleanList;



public final class KIFD {

    public KIFD(ResourceImpl<ParamsLineImpl> params,ResourceImpl<PeopleLineImpl>
people, ResourceImpl<TimeLapsLineImpl> timeLine) throws SQLException {

        Kit kit = new Kit();

        for (ParamsLineImpl i = params.getNext(); null != i ; i = params.getNext()){
            System.out.println(i);
        }

        for (PeopleLineImpl i = people.getNext(); null != i ; i = people.getNext()){
            System.out.println(i);
        }

        for (TimeLapsLineImpl i = timeLine.getNext(); null != i ;  i = timeLine.getNext()){
            kit.setIr(i.getBooleans());
            kit.setIz(i.getNumbers());
            kit.run(new Run(2));
                        i.setInfo(kit.getDS().toString()+";   "+   kit.getDX().toString()+";   "+
kit.getDR().toString());
            i.update();
            kit.downValues();
        }
    }
```

```
}
./src/main/java/ru/binarus/kifd/math/Kit.java
package ru.binarus.kifd.math;

import java.util.ArrayList;
import lombok.*;
import ru.binarus.math.ArrayBooleanList;
import ru.binarus.math.ArrayDoubleList;

/**
 * Данные которые требуются для одной итерации
 */
@Getter
@Setter
@ToString
public final class Kit {

    //Constans
    public final Integer COUNT_STATE = 2;
    public final Integer COUNT_COORDINATE = 1;
    public final Integer COUNT_INDICATE = 0;

    //Base
    private ArrayBooleanList s;
    private ArrayBooleanList s1;
    private Integer k;

    //Input
    private ArrayBooleanList ir;
    private ArrayDoubleList iz;

    //Output
    private Integer os;
    private ArrayDoubleList ox;

    private ArrayDoubleList fp;
    private ArrayDoubleList fp1;

    private ArrayList<ArrayDoubleList> fx;
    private ArrayList<ArrayDoubleList> fx1;

    private ArrayList<ArrayDoubleList> fR;
    private ArrayList<ArrayDoubleList> fR1;


    //Storage
    private Double sumP_a = 0.;
```

```
    public void downValues(){
        s=s1;
        fp=fp1;
        fx=fx1;
        fR=fR1;
    }

}
----------------
./src/main/java/ru/binarus/kifd/math/Functions.java
package ru.binarus.kifd.math;

import ru.binarus.math.ArrayDoubleList;
import ru.binarus.math.ArrayBooleanList;

/**
 * Все основные вспомогательные функции.
 */
public final class Functions {

    public static double b(Kit allValue,Integer State){
        return f(allValue.getIz(),allValue.getOx(),State)
            *pi(allValue.getIr(),allValue.getIr(),allValue.getOx(),State)
            *f_p(allValue.getOx(),State);
    }

    /**
     * y() -
     * @param stateFrom s_(k)
     * @param stateTo s_(k+1)
     * @param fazCoordinate x_(k)
     * @return double
     */
    public static double y(Integer stateTo,ArrayDoubleList fazCoordinate, Integer stateFrom)
{
        return q(stateTo,fazCoordinate,stateFrom)*f_a(fazCoordinate,stateFrom);
    }

    /**
      * q() - Условная вероятность перехода структуры из состояния s_(k) в состояние
s_(k+1) прификсированном x_(k)
     * @param stateFrom s_(k)
     * @param stateTo s_(k+1)
     * @param fazCoordinate x_(k)
     * @return double
     */
    public static double q(Integer stateTo, ArrayDoubleList fazCoordinate, Integer stateFrom)
{
```

```
            if(stateFrom == stateTo){
                return 0.4;
            }else{
                return 0.1;
            }
        }

        /**
         * π() - Условная вероятность перехода индикатора структуры из состояния r_(k) в состояние r_(k+1) при фиксированных s_(k+1) , x_(k+1)
         * @param diffIndicatorFrom r_(k)
         * @param diffIndicatorTo r_(k+1)
         * @param state s_(k+1)
         * @param fazCoordinate x_(k+1)
         * @return double
         */
            public static double pi(ArrayBooleanList diffIndicatorTo, ArrayBooleanList diffIndicatorFrom,ArrayDoubleList fazCoordinate,Integer state){
            if(diffIndicatorFrom == diffIndicatorTo){
                return 0.8;
            }else{
                return 0.2;
            }
        }


        /**
         * f double.
         * @param fazCoordinateTo x_(k+1)
         * @param stateTo s_(k+1)
         * @param fazCoordinateFrom x_(k)
         * @param stateFrom s_(k)
         * @return
         */
            public static double f(ArrayDoubleList fazCoordinateTo,Integer stateTo, ArrayDoubleList fazCoordinateFrom,Integer stateFrom){
            return 1;
        }

        /**
         * f double.
         * @param diffFazCoordinate z_(k)
         * @param fazCoordinate x_(k)
         * @param state s_(k)
         * @return the double
         */
            public static double f(ArrayDoubleList diffFazCoordinate,ArrayDoubleList fazCoordinate,Integer state){
```

```
            return 1;
        }

        /**
         * f'~' - Прогнозируемая на один шаг дискретности вперед плотность вероятности вектора x_(k)
         *
         * @param fazCoordinate x_(k)
         * @param state s_(k)
         * @return double
         */
        public static double f_p(ArrayDoubleList fazCoordinate,Integer state){
            return 1; // Должен задаться
        }

        /**
         * f'^' - Условная плотность вероятности вектора x_(k) при фиксированном s_(k)
         * @param fazCoordinate x_(k)
         * @param state s_(k)
         * @return double
         */
        public static double f_a(ArrayDoubleList fazCoordinate,Integer state){
            return 1;  // Должен задаться
        }
        /**f'-' - Нормировочный коэффициент */
            public static double f_n(ArrayDoubleList diffFazCoordinate, ArrayBooleanList diffIndicator){
            return 0.5;
        }
        /** x'~'0*/
        public static ArrayDoubleList x_c0(Kit allValue,Integer state){
            return diff(allValue.getOx(), allValue.getFx1().get(state));
        }
        /** x'^'0*/
        public static ArrayDoubleList x_p0(Kit allValue,Integer state){
            return diff(allValue.getOx(), allValue.getFx().get(state));
        }
        /**Вычесть из одного листа другой*/
         public static ArrayDoubleList diff(ArrayDoubleList diffFazCoordinate, ArrayDoubleList diffIndicator){
            ArrayDoubleList value = new ArrayDoubleList(diffFazCoordinate.size());
            for(int iter = 0; iter < diffFazCoordinate.size(); iter++){
                value.add(diffFazCoordinate.get(iter) - diffIndicator.get(iter));
            }
            return value;
        }
    }
    ----------------
```

./src/main/java/ru/binarus/kifd/math/Core.java

```
package ru.binarus.kifd.math;
import ru.binarus.math.ArrayDoubleList;
/** Тут основная логика работы программы*/
public final class Core {
   private Kit values;
   /** @param values the values*/
   public Core(Kit values) {
      this.values = values;
   }
   /**p`^`(s_k) - Коррекция Классификатор*/
   private void correlationClassifier(){
      for(int state = 0; state < values.COUNT_STATE; state++) {
         Double result = values.getFp().get(state);
         Double sumIntegral = 0.0;
         result = result * (1 / Functions.f_n(values.getIz(), values.getIr()));
         for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++) {
            sumIntegral += (Functions.b(values,stateTwo));
         }
         result = result * sumIntegral;
         values.getFp1().set(state,result);
      }
   }
   /**x'^'(s_k) - Коррекция Фильтр */
   private void correlationFilter(){
      for(int state = 0; state < values.COUNT_STATE; state++){
         ArrayDoubleList result = new ArrayDoubleList(values.COUNT_COORDINATE);
                              Double sumOnePart = (1 / (Functions.f_n(values.getIz(),
values.getIr())*values.getFp1().get(state)))
                * values.getFp().get(state);
         Double sumIntegral = 0.0;
         for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++) {
            sumIntegral += (Functions.b(values,stateTwo));
         }
            for(int fazCoordinates = 0; fazCoordinates < values.COUNT_COORDINATE;
fazCoordinates++){
            result.add(
                  sumOnePart
                  * (values.getOx().get(fazCoordinates)
                        *sumIntegral)
            );
         }
         values.getFx1().set(state,result);
      }
   }
   /**R'^'(s_k) - Коррекция Дисперсиометр*/
   private void correlationDispersiometer(){
      for(int state = 0; state < values.COUNT_STATE; state++){
```

```
                ArrayDoubleList result = new ArrayDoubleList(values.COUNT_COORDINATE);
        Double sumOnePart = (1 / (Functions.f_n(values.getIz(),
values.getIr())*values.getFp1().get(state)))
                    * values.getFp().get(state);
                Double sumIntegral = 0.0;
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++) {
                   sumIntegral += (Functions.b(values,stateTwo));
                }
                    for(int fazCoordinates = 0; fazCoordinates < values.COUNT_COORDINATE;
fazCoordinates++){ result.add(sumOnePart* ((values.getOx().get(fazCoordinates) -
values.getFx1().get(state).get(fazCoordinates) )
                                *sumIntegral)
                   );
                }
                values.getFR1().set(state,result);
             }
          }
          /**p'~'(s_k) - Прогноз Классификатор*/
          private void prognosisClassifier(){
             for(int state = 0; state < values.COUNT_STATE; state++) {
                Double result = 0.0;
                Double sumOnePart = 0.0;
                Double sumIntegral = 0.0;
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++){
                   sumOnePart = sumOnePart + values.getFp1().get(stateTwo);
                }
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++){
                   sumIntegral = sumIntegral + Functions.y(stateTwo,values.getOx(),state);
                }
                result = sumOnePart * sumIntegral;
                values.getFp().set(state,result);
             }
          }
          /** x'~'(s_k) - Прогноз Фильтр*/
          private void prognosisDispersiometer(){
             for(int state = 0; state < values.COUNT_STATE; state++) {
                ArrayDoubleList result = new ArrayDoubleList(values.COUNT_COORDINATE);
                Double sumOnePart = 0.0;
                Double sumOneIntegral = 0.0;
                Double sumTwoIntegral = 0.0;
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++){
                   sumOnePart = sumOnePart + values.getFp1().get(stateTwo);
                }
                sumOnePart = Math.pow(values.getFp().get(state),-1) * sumOnePart;
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++){
                   sumOneIntegral = sumOneIntegral + Functions.y(stateTwo,values.getOx(),state);
                }
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++){
```

```
        sumTwoIntegral = sumTwoIntegral + Functions.f(values.getOx(),stateTwo,values.getOx(),state);
                }
                    for(int fazCoordinates = 0; fazCoordinates < values.COUNT_COORDINATE; fazCoordinates++){
                    result.add(
                            sumOnePart
                            * sumOneIntegral
                            * values.getOx().get(fazCoordinates)
                            * sumTwoIntegral
                    );
                }
                values.getFx().set(state,result);
            }
        }
        /**R'~'(s_k) - Прогноз Дисперсиометр */
        private void prognosisFilter(){
            for(int state = 0; state < values.COUNT_STATE; state++) {
                ArrayDoubleList result = new ArrayDoubleList(values.COUNT_COORDINATE);
                Double sumOnePart = 0.0;
                Double sumOneIntegral = 0.0;
                Double sumTwoIntegral = 0.0;
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++){
                    sumOnePart = sumOnePart + values.getFp1().get(stateTwo);
                }
                sumOnePart = Math.pow(values.getFp().get(state),-1) * sumOnePart;
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++){
                    sumOneIntegral = sumOneIntegral + Functions.y(stateTwo,values.getOx(),state);
                }
                for(int stateTwo = 0; stateTwo < values.COUNT_STATE; stateTwo++){
                                                    sumTwoIntegral = sumTwoIntegral + Functions.f(values.getOx(),stateTwo,values.getOx(),state);
                }
                    for(int fazCoordinates = 0; fazCoordinates < values.COUNT_COORDINATE; fazCoordinates++){
                    result.add(
                            sumOnePart
                                * sumOneIntegral
                                                        * (values.getOx().get(fazCoordinates) - values.getFx1().get(state).get(fazCoordinates) )
                                * sumTwoIntegral
                    );
                }

                values.getFx().set(state,result);
            }
        }
        /** s'^'_k - Идентификация структуры */
```

```
        private void identificStructurs(){
            Integer bestState = 0;
            Double stateHaveMax = 0.0;
            for(int state = 0; state < values.COUNT_STATE; state++){
                if(stateHaveMax < values.getFp1().get(state)){
                    stateHaveMax = values.getFp1().get(state);
                    bestState = state;
                }
            }
            values.setOs(bestState);
        }

        /**x'^'_k - Новое положение фазовых координат */
        private void newFazovCoordinate(){
            ArrayDoubleList result = values.getOx();
            result.multiplyByNumber(0.0);
            for(int state = 0; state < values.COUNT_STATE; state++){
                for(int fazCoordinates = 0; fazCoordinates < values.getFx1().size(); fazCoordinates+
+){
                                                result.set(fazCoordinates,  result.get(fazCoordinates)+
(values.getFx1().get(state).get(fazCoordinates)*values.getFp1().get(state)));
                }
            }
            values.setOx(result);
        }

        /** R'^'_k - Новая ковариация */
        private void newFazovCoordinateDispers(){
            ArrayDoubleList result = values.getOx();
            result.multiplyByNumber(0.0);
            for(int state = 0; state < values.COUNT_STATE; state++){
                for(int fazCoordinates = 0; fazCoordinates < values.getFx1().size(); fazCoordinates+
+){
                                                result.set(fazCoordinates,result.get(fazCoordinates)  +
(values.getFR1().get(state).get(fazCoordinates)      +      (values.getOx().get(fazCoordinates)      -
values.getFx1().get(state).get(fazCoordinates))) * values.getFp1().get(state) );
                }
            }
            values.setOx(result);
        }
    }

    ----------------

    ./src/main/java/ru/binarus/kifd/impl/ResourceImpl.java
    package ru.binarus.kifd.impl;
    import java.sql.SQLException;
    public interface ResourceImpl<typeResource> {
```

```
    typeResource getNext() throws SQLException;
    boolean hasNext() throws SQLException;
    void reset();
}
-----------------
./src/main/java/ru/binarus/kifd/impl/TimeLapsLineImpl.java
package ru.binarus.kifd.impl;
import java.sql.SQLException;
import java.util.Map;
public interface TimeLapsLineImpl extends Line {
    Map<String,Double> getNumbers();
    Map<String,Boolean> getBooleans();
    double  getTime();
    void setInfo(String info);
    void update() throws SQLException;
    void insert() throws SQLException;
}
-----------------
./src/main/java/ru/binarus/kifd/impl/Line.java
package ru.binarus.kifd.impl;public interface Line {
    int getId();
}
./src/main/java/ru/binarus/kifd/impl/ParamsLineImpl.java
package ru.binarus.kifd.impl;
import java.sql.SQLException;
public interface ParamsLineImpl extends Line {
    String getName();
    double getExpectedValue();
    double getDispersion();
    boolean isState();
    void insert() throws SQLException;
    void update() throws SQLException;
}
./src/main/java/ru/binarus/kifd/impl/PeopleLineImpl.java
package ru.binarus.kifd.impl;
import java.sql.SQLException;
import java.util.Map;
public interface PeopleLineImpl extends Line {
    String getName();
    Map<String,Double> getNumbers();
    Map<String,Boolean> getBooleans();
    double  getProbability();
    void update() throws SQLException;
    void insert() throws SQLException;
}
```