Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Иркутский национальный исследовательский технический университет»

На правах рукописи

Харахинов Владимир Александрович

НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ РЕШЕНИЯ ЗАДАЧ КЛАСТЕРИЗАЦИИ И КЛАССИФИКАЦИИ ДАННЫХ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Специальность 2.3.1 – Системный анализ, управление и
обработка информации, статистика

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
кандидат технических наук,
доцент Сосинская С.С.

Иркутск – 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы. На сегодняшний день интеллектуальный анализ данных является междисциплинарной областью знаний, основу которой заложили следующие научные дисциплины: математическая статистика, искусственный интеллект, машинное обучение, визуализация данных.

В широком смысле современная концепция анализа данных предполагает, что:

- данные могут быть неточными, неполными, противоречивыми, разнородными, косвенными, и при этом иметь гигантские объёмы;
- сами алгоритмы анализа данных могут обладать «элементами интеллекта», в частности, способностью обучаться по прецедентам, то есть делать общие выводы на основе частных наблюдений;
- процессы переработки «сырых» данных в информацию и обратно уже не могут быть выполнены вручную и требуют нетривиальной автоматизации [1].

К основным задачам интеллектуального анализа данных можно отнести следующие:

- задача классификации;
- задача регрессионного анализа;
- задача прогнозирования;
- задача кластеризации;
- задача поиска ассоциативных правил;

Задача классификации является наиболее распространенной задачей анализа данных. Для ее решения используются признаки, которыми описываются объекты исследуемого набора данных и выделяются группы объектов - классы; по этим признакам новый объект можно отнести к тому или иному классу.

Кластеризация является логическим продолжением идеи классификации. Особенность кластеризации заключается в том, что классы объектов изначально

не предопределены. Результатом кластеризации является разбиение объектов на классы.

Приведенные выше задачи анализа данных решаются путем применения самых разнообразных методов анализа данных.

На данном этапе постоянно существует необходимость решения задач классификации и кластеризации в той или иной сфере человеческой деятельности: определения спама, распознавания голоса, кредитоспособности заемщика. Решение этих задач с помощью интеллектуального анализа данных зачастую позволяет повысить уровень автоматизации того или иного процесса, тем самым оказывая положительный эффект на всю систему, к которой относится этот процесс.

Нередко проведение классификации и кластеризации связано с потребностью принимать решения различной сложности. Для снижения вероятности принятия неверных решений применяют СППР (системы поддержки принятия решений), в ряде которых активно используются совместно различные методы интеллектуального анализа данных.

По одной из классификаций [2] методы интеллектуального анализа данных разделяют на две группы: статистические и кибернетические методы.

В литературе, а именно в [1], статистическая группа методов анализа данных представляет собой четыре взаимосвязанных направления:

1. Дескриптивный (описательный) анализ и описание исходных данных.

2. Анализ связей (корреляционный и регрессионный анализ, факторный анализ, дисперсионный анализ).

3. Многомерный статистический анализ (компонентный анализ, дискриминантный анализ, многомерный регрессионный анализ, канонические корреляции и др.).

4. Анализ временных рядов (динамические модели и прогнозирование).

Кибернетические методы анализа данных – это множество методов, объединенных идеей компьютерной математики и использования теории искусственного интеллекта. Эта группа состоит из следующих подгрупп методов:

- искусственные нейронные сети (классификация, кластеризация, прогнозирование);
- эволюционное программирование;
- генетические алгоритмы;
- ассоциативная память (поиск аналогов, прототипов);
- нечеткая логика;
- деревья решений;
- системы обработки экспертных знаний.

На текущий момент кибернетические методы анализа данных являются наиболее распространенными; исследователи в области интеллектуального анализа данных постоянно разрабатывают новые и совершенствуют уже разработанные, проверенные на практике методы. Один из возможных вариантов совершенствования метода – его совместное использование с другими методами для решения одной задачи.

Несмотря на известные достоинства нейронных сетей, в ряде случаев они обладают свойствами, приводящими к нежелательным результатам, например, от обучающей выборки зависит архитектура сети, количество слоев, количество нейронов в каждом слое. Процесс обучения сети может протекать достаточно медленно из-за большой размерности входных данных. Одним из основных способов уменьшения размерности данных является проведение процесса редукции данных. Обучение на ненормализованных данных может снизить качество решения задач классификации и кластеризации объектов. Качество решения может также зависеть от выбора начальной конфигурации сети, начальных весов нейронов, выбора необходимых функций активации.

На основании вышесказанного была определена тема работы: «Нейросетевые технологии решения задач кластеризации и классификации данных в технических системах».

Таким образом, актуальность выбранной темы диссертационной работы обусловлена необходимостью разработки рациональной методики обработки

данных при инициализации параметров нейронных сетей в задачах классификации и кластеризации.

Целью диссертационной работы является повышение качества решения задач классификации и кластеризации в технических системах за счет совместного использования редукции, нормализации анализируемых данных автокодировщиком и настройки параметров слоя Кохонена нейронной сети с применением генетического алгоритма.

Для достижения поставленной цели решались следующие **задачи**:

1. Разработка методики решения задачи кластеризации на основе нейросетевой технологии с применением генетического алгоритма настройки параметров нейронной сети;
2. Редукция и нормализация анализируемых данных на основе нейросетевых технологий с последующей оценкой влияния на качество кластерного анализа;
3. Разработка модели функционирования системы анализа данных с помощью математического аппарата сетей Петри;
4. Разработка алгоритмического и программного обеспечения, нейронной сети для решения задач классификации и кластеризации, а также для принятия управленческих решений по оперативному реагированию на дорожно-транспортные происшествия;
5. Апробация разработанной методики, предложенных методов и СППР в предметных областях: транспорт, машиностроение, биология, сельское хозяйство, банковское дело

Степень разработанности темы. Вопросы классификации и кластерного анализа рассматриваются в работах многих отечественных и зарубежных авторов: С.А. Айвазяна, М.А. Айзермана, М.М. Бонгарда, С. Виежхона, Б. Дюрана, М. Жамбю, Ю.И. Журавлёва, Н.Г. Загоруйко, Л. Кауфмана, И.Д. Манделя, Б.Г. Миркина, М.С. Олдендерфера, В. Рэнда, Р. Триона, М. Штейнбаха и многих других.

Вопросам применения нейронных сетей для решения различных задач анализа данных уделено внимание в работах: А.Н. Горбаня, А. Кофмана, Т.

Кохонена, В. Маккалока, М. Моллера, У. Питтса, Ф. Розенблатта, Ф. Уоссермена, С. Хайкина, Д. Хинтона, Д. Хопфилда.

Вопросами совместного применения различных методов интеллектуального анализа данных занимались следующие авторы: З. Гуо, Н. Кадаба, Д. Келли, Ж. Корбич, К. Сузуки, Л. Миддлтон, Д. Хинтон, Р. Эберхарт.

Объект исследования: технические объекты, характеризуемые множеством признаков состояния.

Предмет исследования: нейросетевые технологии решения задач классификации и кластеризации.

Основные методы исследования: теория сетей Петри, теория информации, теория искусственных нейронных сетей, методы редукции пространства признаков состояния, генетические алгоритмы.

Тематика работы соответствует следующим пунктам паспорта специальности 2.3.1: п. 3 «Разработка критериев и моделей описания и оценки эффективности решения задач системного анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта», п. 5 «Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта», п. 10 «Методы и алгоритмы интеллектуальной поддержки при принятии управленческих решений в технических системах», п. 12 «Визуализация, трансформация и анализ информации на основе компьютерных методов обработки информации».

Научная новизна работы определяется следующими основными результатами, выносимыми на защиту:

1. Впервые разработана методика совместного использования слоя Кохонена и генетического алгоритма с редукцией данных, повышающая качество результатов проводимого кластерного анализа объектов.

2. Предложено использование автокодировщика в качестве эффективного альтернативного способа нормализации анализируемых данных по отношению к общеизвестным способам.

3. Реализован новый подход к редукции данных для задач классификации и кластеризации.

4. Спроектировано и разработано специальное алгоритмическое обеспечение системы анализа, управления, принятия решений и обработки информации, отличающее совместным использованием общеизвестных и предложенных автором методик для классификации и кластеризации технических объектов.

Теоретическая значимость работы заключается в разработке методики совместного использования генетического алгоритма и алгоритма К-средних для настройки параметров самоорганизующегося слоя Кохонена с целью повышения качества результатов кластеризации. Предложено использование автокодировщиков в качестве эффективного альтернативного способа нормализации анализируемых данных.

Практическая значимость состоит в разработке инструментальных средств, позволяющих исследователям проводить обработку и анализ данных с различными методами классификации и кластеризации, а также для принятия управленческих решений по оперативному реагированию на серьезные дорожно-транспортные происшествия (ДТП), существенно затрудняющих пропускную способность автомобильной дороги. Разработанные в диссертации концепция, алгоритмы и методы, а также программные модули могут использоваться при разработке математического и программного обеспечения систем анализа данных в различных отраслях промышленности и иных предметных областях.

Применение результатов. Разработанные методы классификации и кластеризации обработки данных использовались при моделировании транспортных потоков в компании ООО «Центр транспортных технологий»; компанией ООО НПО ССЦ «Ангара» при анализе массивов данных в различных районах Иркутской области; в учебном процессе Института высоких технологий Иркутского национального исследовательского технического университета (ИРНИТУ) при организации учебного курса «Технологии обработки информации».

Результаты исследования подтверждаются наличием соответствующих актов о внедрении.

Достоверность полученных результатов. Достоверность полученных результатов подтверждена совпадением результатов решения задач классификации и кластеризации, которые были получены другими авторами, а также корректностью архитектуры спроектированной системы «Анализ данных».

Апробация работы. Работа выполнялась в ИРНИТУ на кафедре технологии и оборудования машиностроительных производств и вычислительной техники. Основные положения проведенных исследований докладывались на научных семинарах кафедры технологий и оборудования машиностроительных производств и вычислительной техники ИРНИТУ, на Всероссийских молодежных научно-практических конференциях «Винеровские чтения» (г. Иркутск, 2016, 2019), на Международной научной конференции «Applied Physics, Information Technologies and Engineering» (г. Красноярск, APITECH-2019), на VIII Всероссийской научной конференции с международным участием «Информационные технологии интеллектуальной поддержки принятия решений» (г. Уфа, ITIDS 2020), на VIII международном семинаре «Критические инфраструктуры в цифровом мире» (г. Байкальск, IWCI 2021), на XII международной научно-практической конференции «Транспортная инфраструктура Сибирского региона» (г. Иркутск, 2021).

Личный вклад автора. Результаты, составляющие научную новизну и выносимые на защиту, получены лично автором. В остальных работах, полученных совместно другими авторами, автору принадлежат от 40 до 90% полученных научных результатов.

Структура и объем работы. Диссертационная работа состоит из введения, трех глав, заключения, списка литературы из 107 наименований, 5 приложений. Общий объем работы составляет 139 страниц, 41 рисунок и 17 таблиц.

Во введении обоснована актуальность диссертационной работы, на основании чего сформулированы цель и задачи исследования, определены объект,

предмет, методы и средства исследования, научная и практическая значимость работы, изложены научные положения, выносимые на защиту.

В первой главе описаны основные задачи интеллектуального анализа данных и подходы к их решению. Более детально описаны задачи классификации и кластеризации, особое внимание уделяется алгоритму К-средних. Выполнена проверка качества решения задачи кластеризации с помощью индексов Рэнда. Изложена концепция искусственных нейронных сетей, даны основные понятия и описаны три фундаментальных класса архитектур сетей, а также две парадигмы их обучения. Более детально описано применение нейронных сетей для решения задач классификации и кластеризации – рассмотрены архитектуры основных типов сетей и наиболее популярные алгоритмы для их обучения. Раскрывается важность процесса редукции данных в задачах интеллектуального анализа данных, решаемых при помощи нейронных сетей. Приведены выводы по главе.

Вторая глава посвящена проектированию и разработке системы для классификации и кластеризации технических объектов, а также созданию методики предварительной инициализации матрицы весов слоя Кохонена для повышения качества результатов кластерного анализа при использовании данной сети, с последующим включением созданной методики в разрабатываемую систему. При проектировании была построена структурная схема системы, позволяющая наглядно отобразить общий принцип работы системы. Приведено описание разработанной методики использования генетического алгоритма для предварительной настройки матрицы весов слоя Кохонена в виде схемы, с проведением сравнительного анализа этапов работы алгоритма К-средних и генетического алгоритма. Использован математический аппарат сетей Петри для проектирования системы. Приведены выводы по главе.

В третьей главе дано описание структуры и возможностей реализованной автором системы. Апробация реализованной системы была проведена путем решения задач классификации и кластеризации технических объектов на примере решения задач анализа ДТП. В ходе апробации для объектов ДТП была предложена концептуальная модель СППР. Проведена апробация разработанной

методики повышения качества кластеризации при анализе объектов других предметных областей. Приведены выводы по главе.

Заключение содержит основные результаты работы. В приложениях приведены акты внедрения и свидетельство о регистрации программы для ЭВМ, а также часть программного кода одного из модулей системы.

Результаты диссертационного исследования опубликованы в 10 научных работах, из них 1 статья в журнале, индексируемом международной базой Scopus; 4 статьи в изданиях, входящих в Перечень ВАК: «Информационные технологии», «Научный вестник НГТУ», «Программная инженерия». Получено свидетельство о государственной регистрации программы для ЭВМ № 2017617294.

Глава 1. Теоретические основы исследований в области интеллектуального анализа данных и их применение для решения задач классификации и кластеризации технических объектов

1.1 Общая постановка задач классификации и кластеризации в анализе данных

Под классификацией объектов (наблюдений, событий), описываемых набором числовых признаков, понимают способы отнесения этих объектов к одному из заранее известных классов.

С математической точки зрения процесс классификации можно описать следующим образом:

Пусть X - множество описаний объектов, Y - конечное множество номеров (имён, меток) классов. Существует неизвестная целевая зависимость - отображение $y^*: X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = \{(x_1, y_1), \cdots, (x_m, y_m)\}$. Требуется построить алгоритм $\alpha: X \to Y$, способный классифицировать произвольный объект $x \in X$ [3].

Множество описаний объектов X нередко называют входными данными, которые могут быть представлены в виде: признакового описания, матрицы расстояний, временного ряда, изображения (видеоряда), а также в ряде других, гораздо менее распространенных видах.

Подавляющее большинство методов классификации можно отнести к одному из следующих подходов:

- классификация с помощью деревьев решений;
- байесовский классификатор;
- статистические методы;
- метод главных компонент;
- линейный разделитель;
- генетический алгоритм;

• искусственные нейронные сети (ИНС).

В дальнейшем будет детально описан подход к классификации, основанный на применении ИНС, поскольку в рамках диссертационного исследования используется в основном этот подход.

Задача кластеризации сходна с задачей классификации и является ее логическим продолжением. Сам термин кластерный анализ был впервые введен психологом Робертом Трионом в 1939 году [4].

Кластерный анализ (кластеризация) предназначен для разбиения совокупности объектов на однородные группы (кластеры). При решении задачи кластерного анализа различными методами, полученные кластеры могут быть пересекающимися (overlapping) и непересекающимися (non-overlapping) [5]. Примеры изображений с объектами, разделенными на пересекающиеся и непересекающиеся кластеры, приведены на рисунке 1.1.

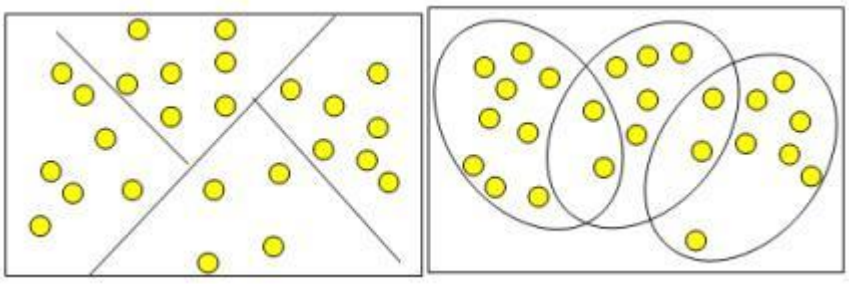

Рисунок 1.1 – Непересекающиеся и пересекающиеся кластеры

Общая постановка задачи кластеризации выглядит следующим образом.

Пусть X – множество описаний объектов. Задана функция расстояния между объектами $p(x, x')$. Имеется конечная обучающая выборка объектов $X^m = \{x_1, \cdots, x_m\} \in X$. Требуется разбить выборку на подмножества, называемые кластерами так, чтобы каждый кластер состоял из объектов, близких по метрике p, а объекты разных кластеров существенно (по выбранной метрике) отличались [5].

Существует множество метрик, но наиболее популярными являются следующие:

• Евклидово расстояние. Вычисляется по формуле $D(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$, где x и y – точки в -мерном пространстве.

• Квадрат евклидова расстояния: $D(x,y) = \sum_{i=1}^{n}(x_i - y_i)^2$.

• Манхэттенское расстояние: $D(x,y) = \sum_{i=1}^{n}|x_i - y_i|$.

• Расстояние Чебышёва: $D(x,y) = max(|x_i - y_i|)$.

Как и в случае с классификацией, множество описаний объектов X зачастую называют входными данными.

Но между процессами классификации и кластеризации есть одно, но очень существенное отличие – в задаче классификации используется Y (конечное множество номеров или имён классов), в задаче кластеризации аналогичного множества не существует.

В настоящее имеется более 100 различных алгоритмов и методов решения задачи кластеризации; каждый из этих методов можно отнести к одному из двух видов: иерархическому или итеративному [6].

Основной подход в иерархических методах кластеризации заключается в итерационном объединении небольших групп объектов (кластеров) в большие (агломеративные методы), либо наоборот - в разделении больших кластеров (дивизимные методы). Графическое представление результатов обычно осуществляется в виде дерева иерархической кластеризации (дендрограммы).

На практике иерархические методы кластерного анализа используются, в основном, при небольших объемах входных данных [7].

Для кластеризации больших объемов данных обычно используют итеративные (неиерархические) методы. Общий принцип работы итеративных методов заключается в разделении всего процесса кластеризации на отдельные итерации. Для выбранного алгоритма кластерного анализа устанавливается условие остановки алгоритма. На каждой итерации исходная выборка разделяется на кластеры и выполняется проверка этого условия.

К наиболее популярным итеративным алгоритмам можно отнести [8]:

- алгоритм K-means;
- алгоритм C-means;
- EM-алгоритм;
- алгоритм DBSCAN;
- нейронная сеть Кохонена (один из основных видов сетей, использующихся для проведения кластерного анализа).

Отдельного рассмотрения в рамках диссертационной работы заслуживают два алгоритма: алгоритм K-means и нейронная сеть Кохонена.

Алгоритм K-means (К-средних), также называемый быстрым кластерным анализом, был разработан Гуго Штейнгаузом (опубликован в 1956 г.) [9] и Стюартом Ллойдом (опубликован в 1957 г.) [10].

Общая идея алгоритма: заданное фиксированное число k кластеров наблюдения сопоставляются кластерам так, что средние в кластере максимально возможно отличаются друг от друга. Каждое наблюдение из выборки может находиться только в одном кластере (непересекающиеся кластеры).

Математически алгоритм можно описать следующим образом [8,11,12].

Имеется некоторая выборка $X = \{x_1, x_2, \cdots x_n\}$ состоящая из n числа наблюдений, задается число кластеров k: $k \in N, k \leq n$.

Требуется разделить выборку X на k кластеров $C_1, C_2, \cdots, C_k$, при этом должны выполняться следующие условия: $C_i \cap C_j = \emptyset, i \neq j$; $\bigcup_{i=1}^{k} C_i = X$.

Цель алгоритма К-средних: разделить выборку X на k кластеров $C_1, C_2, \cdots, C_k$, минимизируя сумму квадратов расстояний от каждого наблюдения до центра кластера, к которому оно отнесено:

$$arg \min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} p(x, u_i)^2$$

Алгоритм состоит из следующих шагов:

1. Инициализация кластеров.

На этом шаге производится определение начальных центров кластеров. Случайным образом выбирается k наблюдений $u_i, i = 1, \cdots, k$ из исходной выборки X, являющихся на этом этапе центрами кластеров $u_i^{(0)} = u_i$.

2. Разделение исходной выборки на кластеры.

Каждое наблюдение относят тому или иному кластеру согласно расстоянию от координат этого наблюдения до текущих координат центров кластеров.

$$\forall x_i \in X, i = 1, \cdots, n: x_i \in C_j \; arg\min_k p\left(x_i, u_k^{(t-1)}\right)^2$$

3. Пересчет координат центров кластеров.

Для каждого кластеры вычисляются новые координаты согласно формуле:

$$\forall i = 1, \cdots, k: u_i^{(t)} = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

4. Проверка условия останова алгоритма.

Алгоритм останавливается в том случае, когда каждое наблюдение принадлежит тому кластеру, которому принадлежало до текущей итерации:

$$\forall C_i, i = 1, \cdots, k: C_i^{(t-1)} = \mathrm{C}_i^t$$

Решив задачу классификации или кластеризации важно корректно произвести оценку качества данного решения. На основе проведенного анализа ряда авторитетных научных работ в следующий параграф были сведены основные подходы к оценке качества решения этих задач.

1.2 Оценка качества классификации и кластеризации

Существует ряд общеизвестных способов проведения оценки качества классификации, среди них [3]:

- общая доля правильно классифицированных объектов (наблюдений) (accuracy);
- полнота (recall)
- точность (precision);

- матрица ошибок (confusion matrix);
- кросс-проверка (cross-validation).

Простейший способ оценки качества решения задачи классификации – вычисление общей доли правильно классифицированных объектов (общая доля). Для ее расчета используется следующая формула:

$$Accuracy = \frac{P}{N}$$

где P – количество правильно классифицированных объектов, а N – это количество всех объектов в анализируемой выборке.

Полнота - это доля объектов истинно принадлежащих данному классу по отношению ко всем объектам истинно принадлежащих данному классу.

$$Recall = \frac{TP}{TP + FN}$$

Точность - это доля объектов истинно принадлежащих данному классу по отношению ко всем объектам, которые классификатор отнес к этому классу.

$$Precision = \frac{TP}{TP + FP}$$

Пусть Y - вектор истинных номеров классов, а Y' - вектор номеров классов, полученный в результате классификации. Тогда для i наблюдения существует три возможных случая: TP - истинно-положительное классифицирование наблюдения ($Y_i = Y_i'$); FP – ложно-положительное, классификатор отнес i наблюдение, относящееся к другому классу, в текущий ($Y_i \neq Y_i'$); FN – ложно-отрицательное, классификатор отнес i наблюдение, относящееся к этому классу, в другой класс $Y_i \neq Y_i'$.

Кросс-проверка - это способ оценки качества классификации на данных из тестового множества, которое также называют кросс-проверочным множеством. Точность классификации тестового множества сравнивается с точностью классификации обучающего множества.

Если классификация тестового множества дает приблизительно такие же результаты, как и классификация обучающего множества, считается, что данная классификация прошла кросс-проверку.

Матрица ошибок – это квадратная матрица, в которой число столбцов и строк равно числу классов C. Столбцы формируются из полученных в ходе классификации классов, а строки представляют собой истинные классы для объектов.

Матрица ошибок $M = \|m_{ij}\|(i = 1 \cdots C; j = 1 \cdots C)$ представлена в виде:

$$\begin{Vmatrix} m_{11} & m_{12} & & \\ m_{21} & m_{22} & & \\ & & \ddots & \\ & & & m_{CC} \end{Vmatrix}$$

Элементы по диагонали матрицы m_{ii} обозначают число корректно классифицированных объектов по каждому i классу. Элементы $m_{i1}, m_{i2}, \cdots, m_{iC}$ представляют количество некорректно классифицированных объектов из выборки, исключением является m_{ii}.

Задача оценки качества кластеризации является более сложной по сравнению с оценкой качества классификации. Такие оценки не должны зависеть от номеров кластеров, а только от самого разбиения выборки.

Наиболее популярными способами оценки качества являются: индекс Рэнда – Rand Index (RI) и отрегулированный индекс Рэнда - Adjusted Rand Index (ARI) [13]. В обоих случаях предполагается, что известны истинные номера кластеров объектов. ARI не зависит от самих номеров кластеров, а только от разбиения выборки на кластеры.

Пусть n - число объектов в выборке, A – вектор истинных номеров кластеров каждого объекта, B – вектор номеров кластеров, полученный алгоритмом кластеризации. Обозначим через a - число пар объектов, имеющих одинаковые номера кластеров в векторах A и B и находящихся в одном кластере, через b - число пар объектов, имеющих различные номера кластеров и находящихся в разных кластерах. Тогда Rand Index (RI):

$$RI = \frac{a+b}{\binom{n}{2}} = \frac{2(a+b)}{n(n-1)}$$

То есть это доля объектов, для которых эти разбиения (исходное и полученное в результате кластеризации) "согласованы" [13].

В отличие от RI, ARI не зависит от самих значений номеров кластеров и перестановок этих номеров, а только от разбиения выборки на кластеры.

Для вычисления ARI необходимо построить таблицу сопряженности (таблица 1.1).

Таблица 1.1 - Таблица сопряженности

Y / X	Y_1	Y_2	$\cdots$	Y_s	$Sums$
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
$Sums$	b_1	b_2	$\cdots$	b_s	

В таблице $X = \{X_1, X_2, \dots, X_r\}$ и $Y = \{Y_1, Y_2, \dots, Y_s\}$ - результаты кластерного анализа (множества X_i и Y_j– содержат некоторые наборы объектов, отнесенных к i-му и -му кластеру соответственно), либо результат анализа и целевое разбиение. r – число кластеров, полученное одним методом; s – другим методом (либо число s заранее определено целевым разбиением). Значения n_{ij} – число объектов, общих для множеств X_i и Y_j: $n_{ij} = |X_i \cap Y_j|$. Значение $a_R = \sum_{i=1}^{s} n_{Ri}$, значение $b_S = \sum_{j=1}^{r} n_{jS}$, где $R \in [1 \cdots r]$ и $S \in [1 \cdots s]$.

ARI вычисляется по формуле:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}$$

Данный индекс является мерой расстояния между различными разбиениями выборки. ARI принимает значения в диапазоне [−1,1]. Отрицательные значения соответствуют "независимым" разбиениям на кластеры, значения, близкие к нулю – случайным разбиениям, и положительные значения говорят о том, что два разбиения схожи (совпадают при ARI=1) [14].

Приведенный ниже пример наглядно демонстрирует вычисление ARI.

Пусть вектора X и Y имеют следующие значения:

X	1	2	3	3	2	1	1	3	3	1	2	2
Y	3	2	3	2	2	1	1	2	3	1	3	1

Все анализируемые объекты разделены на 3 кластера.

Строится таблица сопряженности (таблица 1.2).

Таблица 1.2 - Таблица сопряженности

X \ Y	Y_1	Y_2	Y_3	$Sums$
X_1	3	0	1	4
X_2	1	2	1	4
X_3	0	2	2	4
$Sums$	4	4	4	

Имея все значения n_{ij}, a_i, b_j, легко рассчитать значение отрегулированного индекса Рэнда.

В данном случае $ARI = \frac{6-[18\times18]/\binom{12}{2}}{\frac{1}{2}[18+18]-[18\times18]/\binom{12}{2}} = 0.08333$

Чем больше значения RI и ARI, тем больше соответствие между двумя разбиениями на кластеры.

1.3 Концепция искусственных нейронных сетей

Искусственные нейронные сети (далее нейронные сети) представляют собой технологию, основу которой заложили различные дисциплины: математика, статистика, физика, нейрофизиология, компьютерные науки.

Концепция нейронных сетей берет свое начало с работы Мак-Каллока и Питца изданной в 1943 г., в которой ученые впервые предложили идею использования нейронных сетей в качестве вычислительных машин [15].

В общем случае нейронная сеть представляет собой программную или аппаратную реализацию, в некотором смысле моделирующую способ обработки мозгом конкретной задачи.

В концепции нейронных сетей важнейшим понятием является понятие искусственного нейрона (далее нейрон). Нейрон – объект, служащий для обработки информации в нейронной сети. В состав нейрона включены следующие элементы: набор синапсов, сумматор, а также функция активации (рисунок 1.2).

На рисунке 1.2 можно отметить элемент b_k – это пороговый элемент, который в отечественной (и переведенной иностранной) научной литературе называют смещением b_k. Значение порогового элемента сигнализирует об уменьшении или увеличении значения входного сигнала, подаваемого в качестве аргумента в функцию активации.

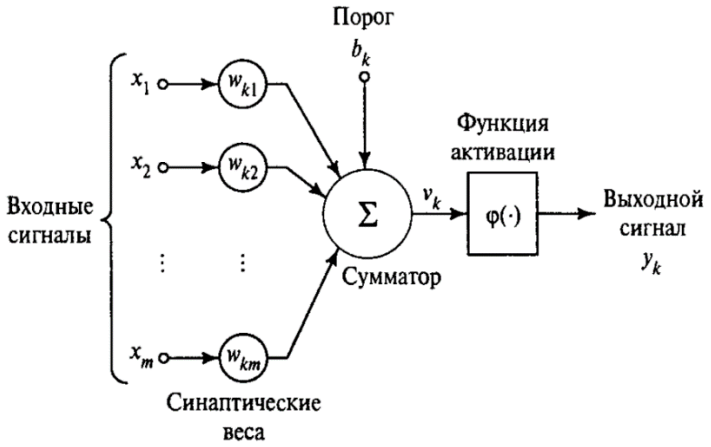

Рисунок 1.2 – Модель нейрона

Математически функционирование нейрона k можно описать парой уравнений:

$$v_k = \sum_{j=1}^{m} w_{kj}\, x_j$$

$$y_k = \varphi(v_k + b_k)$$

Функции активации определяют выходной сигнал нейрона. Традиционно исследователи выделяют три основных вида функций активации:

1) сигмоидальная функция, получившая наибольшее распространение;

2) линейная функция;

3) пороговая.

В 1957 году Розенблатт опубликовал свой научный труд, посвященный разработанной им нейронной сети (являющейся одной из первых) – персептрону Розенблатта [16].

Персептрон Розенблатта имеет простейшую архитектуру. Данная сеть предназначена для классификации линейно-разделимых сигналов: то есть все классифицируемые объекты из анализируемой выборки разделяются некоторой гиперплоскостью на 2 области. Гиперплоскость определяется уравнением:

$$\sum_{i=1}^{m} w_i x_i + b = 0$$

На рисунке 1.3 проиллюстрирован график разделяющей гиперплоскости при $m = 2$.

С тех пор мировое научное сообщество открыло множество архитектур нейронных сетей для решения различных задач.

Существует большое число видов нейронных сетей, но общепринято выделять три основополагающих вида сетей [17].

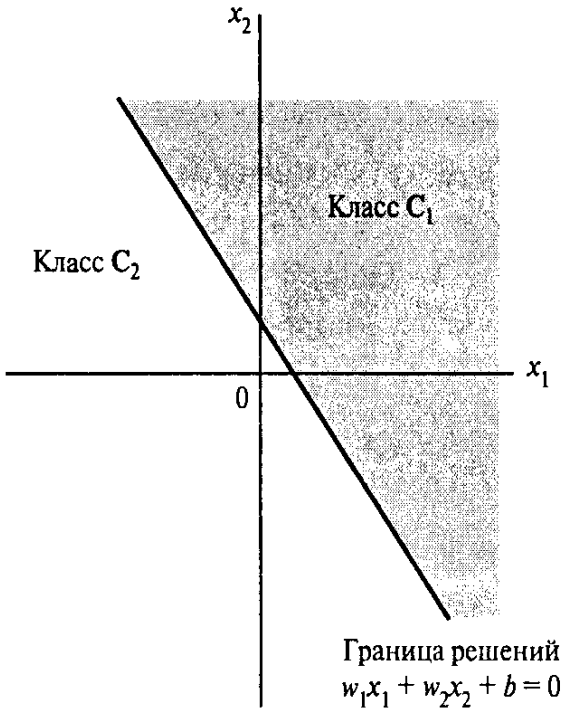

Рисунок 1.3 – Разделяющая гиперплоскость

•На рисунке 1.4 представлена архитектура однослойной сети прямого распространения. Сигнал идет строго в одном направлении - от нейронов на входном слое к нейронам на выходном.

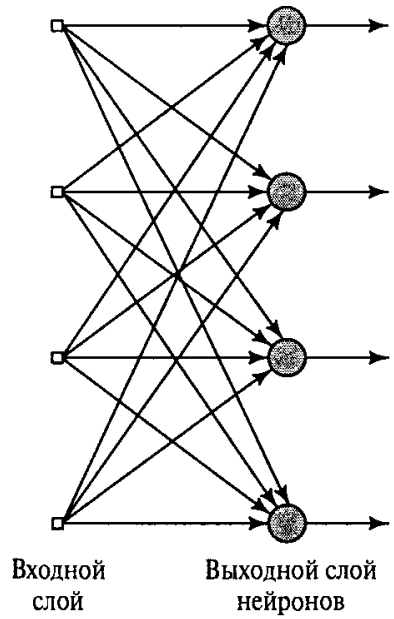

Рисунок 1.4 – Однослойная сеть прямого распространения

•Многослойные сети прямого распространения состоят из нескольких скрытых слоев (реже из одного). Скрытыми нейронами называются нейроны в таких слоях. В отличие от однослойной сети, использование многослойной сети позволяет находить и выделять статистические зависимости более высокого порядка, поскольку в ней присутствуют скрытые нейроны. Рисунок 1.5 содержит графическое представление многослойной нейронной сети прямого распространения.

•Рекуррентные сети обязательно содержат одну или более обратную связь от в отличие от сетей прямого распространения (рисунок 1.6).

Самым важным свойством нейронных сетей является их способность обучаться. В общем случае, процесс обучения нейронной сети представляет собой итеративный процесс, где на каждой итерации в сети происходит изменение значений синаптических весов и смещений.

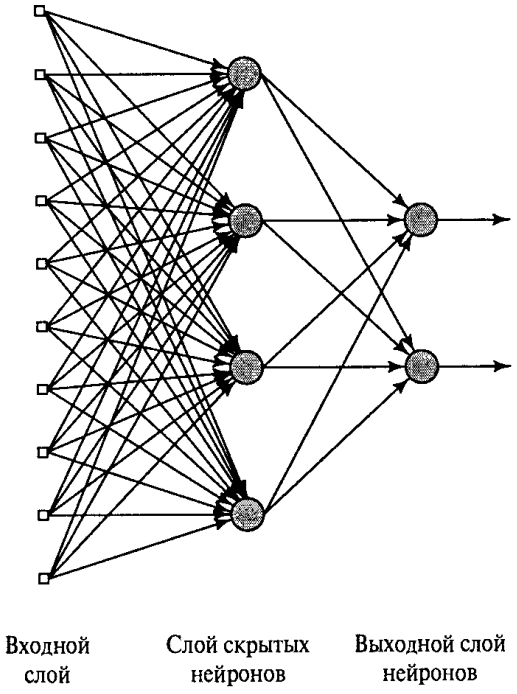

Рисунок 1.5 – Многослойная сеть прямого распространения

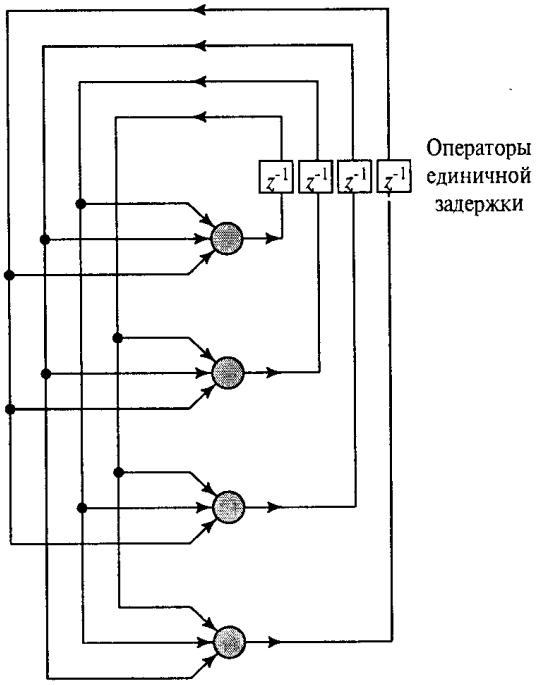

Рисунок 1.6 – Рекуррентная сеть

На данный момент существует множество алгоритмов обучения нейронных сетей. Однако вне зависимости от конкретного алгоритма обучения сети можно выделить общую идею процесса обучения, которая состоит в следующем:

1. В нейронную сеть поступают стимулы из внешней среды.

2. Элементы нейронной сети (синаптические веса нейронов, смещения) изменяют свои значения.

3. После произведенных изменений нейронная сеть отвечает на стимулы уже иным образом.

Практически каждый из ныне использующихся алгоритмов обучения можно отнести к одной из двух фундаментальных парадигм нейросетевого обучения [17].

Первая парадигма основана на обучении с учителем (supervised learning). В общем случае учитель формируется на основе полученных знаний об окружающей среде. Знания нередко представляются в виде выборки: набор признаков для описания объекта и желаемый результат при анализе. То есть, анализируемая выборка должна иметь целевой вектор. Учитель указывает желаемый отклик на выходном слое обучаемой нейронной сети для каждого

соответствующего входного вектора. Сигнал ошибки и обучающий вектор задействованы в процессе корректировки параметров сети. Таким образом, процесс обучения по парадигме обучения с учителем можно описать с помощью блочной диаграммы (рисунок 1.7).

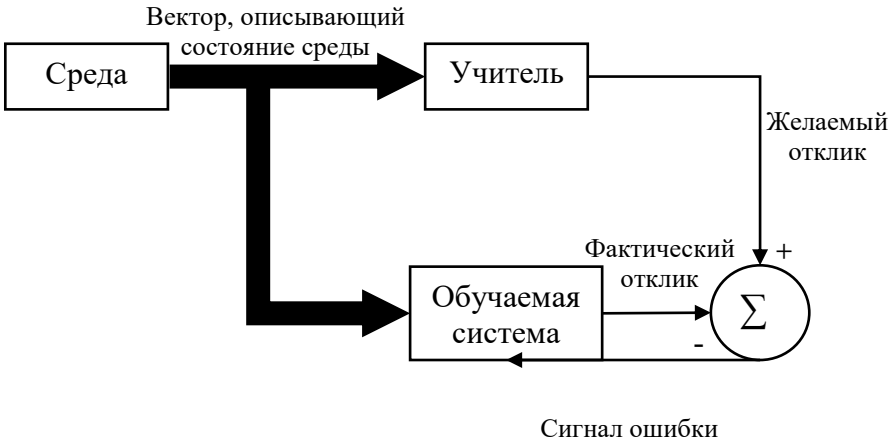

Рисунок 1.7 – Блочная диаграмма обучения с учителем

Другая парадигма основана на обучении без учителя (unsupervised learning). В некоторых случаях эту парадигму называют обучением на основе самоорганизации. Данная парадигма примечательна отсутствием учителя (целевого вектора), контролирующего процесс обучения сети. Блочная диаграмма обучения без учителя выглядит следующим образом (рисунок 1.8).

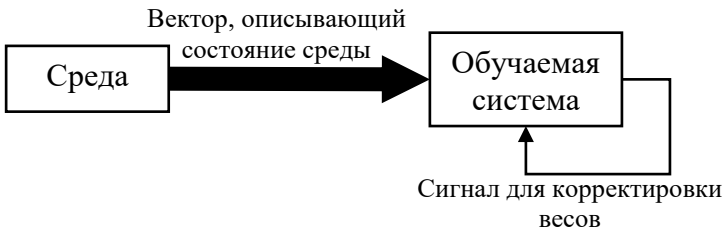

Рисунок 1.8 – Блочная диаграмма обучения без учителя

1.4 Использование нейронных сетей для решения задач классификации и кластеризации

Процесс классификации, основанный на нейросетевом подходе, реализуют при помощи следующих архитектур: однослойный персептрон, многослойный персептрон и сеть распознавания образов. Эти архитектуры объединяет одно – каждая из них является сетью прямого распространения.

При решении современных задачи классификации практически не применяют однослойные персептроны, поскольку он имеет тривиальную архитектуру и не может быть обучен решать такие задачи. Но многослойный персептрон до сих пор используется в современных научных исследованиях. В качестве примера многослойный персептрона с двумя скрытыми слоями приведен на рисунке 1.9.

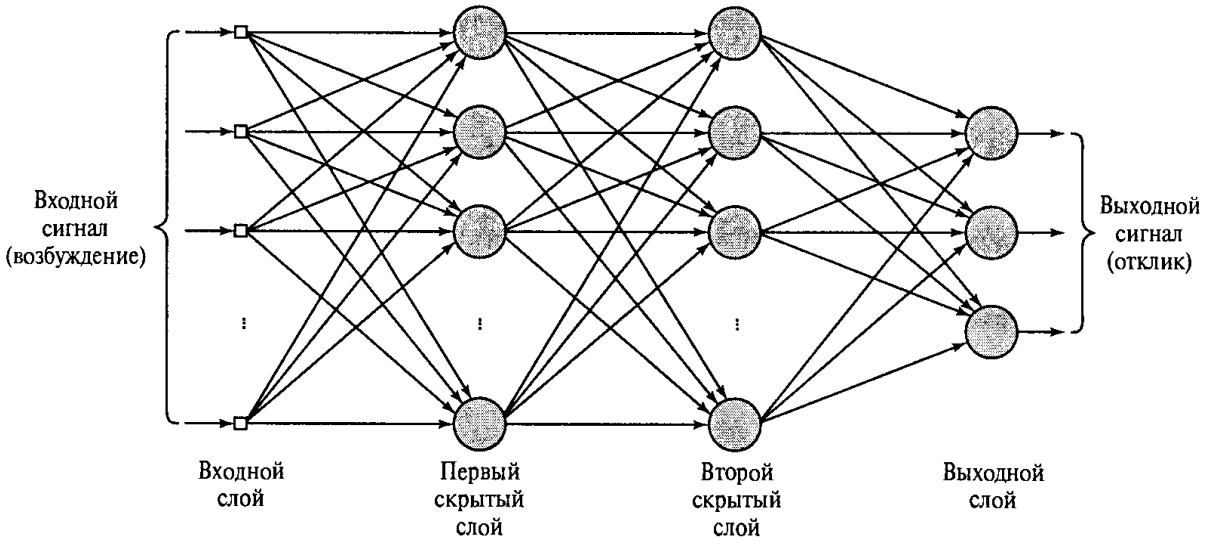

Рисунок 1.9 – Архитектура многослойного персептрона

Наиболее традиционный способ проведения обучения многослойного персептрона включает использование парадигмы обучения с учителем. Более конкретно в данном случае используется метод обратного распространения ошибок. Суть этого метода заключается в том, что он состоит из двух этапов (двух проходов по каждому слою сети): прямой и обратный проходы [17].

Прямой проход предполагает, что набор входных сигналов будет подан на вход сети, как результат - будет получен набор выходных сигналов. Абсолютно все синаптические веса сети во время прямого прохода статичны.

Корректировка всех синаптических весов производится при обратном проходе, при этом используется некоторое правило для коррекции ошибок.

Математически данный процесс описывается следующим образом.

•Прямой проход.

Пусть обучающий пример представлен парой $\big(x(n), d(n)\big)$, где $x(n)$ – входной вектор, поступающий на вход нейронной сети; $d(n)$ – желаемый отклик, представляемый выходному слою нейронов для формирования сигнала ошибки, где n – номер итерации.

Взвешенная сумма всех синаптических весов и смещения нейрона j слоя l вычисляется по формуле:

$$v_j^l(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n) y_i^{(l-1)}(n),$$

где $y_i^{(l-1)}(n)$ – выходной сигнал нейрона j, расположенного в предыдущем слое $l-1$, на итерации n; $w_{ji}^{(l)}(n)$ – синаптический вес связи нейрона j слоя l с нейроном i слоя $l-1$.

При $i = 0$: $y_i^{(l-1)}(n) = 1$, а $w_{ji}^{(l)}(n) = b_j^l(n)$ – смещение, применяемое к нейрону j слоя l.

Выходной сигнал нейрона j слоя l выражается следующим образом:

$$y_j^{(l)}(n) = \varphi_j\left(v_j(n)\right)$$

Сигнал ошибки вычисляется по формуле:

$$e_j(n) = d_j(n) - y_j(n)$$

•Обратный проход.

Локальные градиенты узлов сети вычисляются по следующей формуле:

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\varphi_j'\left(v_j^{(L)}(n)\right) \text{ для нейрона } j \text{ слоя } L \\ \varphi_j'\left(v_j^{(l)}(n)\right)\sum_k \delta_k^{(l+1)}(n)\delta_{kj}^{(l+1)}(n) \text{ для нейрона } j \text{ слоя } l' \end{cases}$$

где L – выходной слой, l – скрытый слой, а φ_j' обозначает дифференцирование по аргументу.

Изменение синаптических весов слоя l сети выполняется в соответствии с обобщенным дельта-правилом:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n)y_j^{(l-1)}(n),$$

где η – параметр скорости обучения; α – постоянная момента.

В основе большинства современных алгоритмов обучения многослойного персептрона лежит метод обратного распространения ошибки. В таблице 1.3 приведены наиболее распространенные алгоритмы.

Таблица 1.3 - Алгоритмы обучения многослойного персептрона

Семейство алгоритмов	Алгоритм обучения
Градиентные алгоритмы обучения	Градиентный спуск
	Градиентный спуск с возмущением
	Градиентный спуск с выбором параметра скорости настройки
	Алгоритм Rprop
Метод сопряженных градиентов	Алгоритм Флетчера – Ривса
	Алгоритм Полака – Рибейры
	Алгоритм Паула – Биеле
	Алгоритм Моллера
Квазиньютоновы алгоритмы	Алгоритм BFGS
	Алгоритм Левенберга-Марквардта

В большинстве случаев наиболее эффективным является алгоритм Левенберга-Марквардта [18-30]. В этом алгоритме среднеквадратичная ошибка модели на обучающей выборке является критерием оптимизации. Основная суть этого алгоритма состоит в последовательном приближении заданных начальных значений параметров к некоторому локальному оптимуму.

С математической точки зрения алгоритм Левенберга-Марквардта описывается следующим образом [31].

Задана обучающая выборка – множество пар $D = \{(x_n, y_n)\}_{n=1}^{N}$ независимой переменной $x \in R^M$ (входы сети) и зависимой переменной $y \in R$. Задана функциональная зависимость, представляющую собой регрессионную модель $y = f(w, x_n)$, где f – функция, непрерывно дифференцируемая в области $W \times X$. Требуется найти такое значение вектора параметров w, которое бы доставляло локальный минимум функции ошибки $E_D = \sum_{n=1}^{N}\big(y_n - f(w, x_n)\big)^2$.

Перед началом работы алгоритма задается начальный вектор параметров w. На каждом шаге итерации элементы этого вектора увеличиваются на величину Δw. Для оценки приращения Δw используется линейное приближение функции $f(w + \Delta w, x) \approx f(w, x) + J\Delta w$, где J – якобиан функции $f(w, x_n)$. Матрицу J размерностью $N \times R$ можно представить в виде:

$$J = \begin{bmatrix} \frac{\partial f(w,x_1)}{\partial w_1} & \cdots & \frac{\partial f(w,x_1)}{\partial w_R} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(w,x_N)}{\partial w_1} & \cdots & \frac{\partial f(w,x_N)}{\partial w_R} \end{bmatrix}.$$

Здесь вектор параметров $w = [w_1, \cdots, w_R]^T$.

Чтобы найти значение Δw, нужно решить систему линейных уравнений $\Delta w = (J^T J)^{-1} J^T\big(y - f(w)\big)$. Так как число обусловленности матрицы $J^T J$ есть квадрат числа обусловленности матрицы J, то матрица $J^T J$ может оказаться существенно вырожденной. Поэтому Марквардт предложил ввести параметр регуляризации $\lambda \geq 0$, следовательно, $\Delta w = (J^T J + \lambda I)^{-1} J^T\big(y - f(w)\big)$.

Однако решение задачи классификации возможно и с помощью другой полносвязной сети прямого распространения сигнала - сети распознавания образов.

Сеть распознавания образов также представляет собой многослойную сеть, в которой на выходном слое задействована функция активации мягкого максимума (softmax function) [32]. Ее архитектура имеет сходство с той, что отражена на рисунке 1.9, однако, в ее архитектуре предусматривается наличие лишь одного скрытого слоя.

Данную функцию часто используют на последнем слое глубоких нейронных сетей при решении задач классификации.

Функция мягкого максимума (софтмакс) имеет следующий вид [33]:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}},$$

где z_i - значение на выходе -го нейрона, K – общее число нейронов в слое. Функция преобразует вектор z размерности K в вектор σ той же размерности, где каждая координата σ_i полученного вектора представлена вещественным числом, находящимся в интервале $[0 \cdots 1]$ и сумма координат равна 1.

Основное достоинство функции мягкого максимума заключается в том, что она использует экспоненту.

Функция активации мягкого максимума позволяет трактовать выходные значения нейронов как некоторую вероятность, с которой вектор принадлежит классу, а за счет применения экспоненты только один нейрон имеет значение близкое к единице.

При обучении сети распознавания образов используют алгоритм сопряженных градиентов [34].

Слой софтмакс имеет архитектуру, состоящую из одного слоя, и имеющую функцию активации софтмакс на этом слое. Ее обучение происходит подобно описанному ранее способу обучения сети распознавания образов [35].

Задачу кластерного анализа на основе нейросетевого подхода, реализуют при помощи самоорганизующихся нейронных сетей. Этот вид сетей обучают с целью

выделения групп (кластеров) анализируемых объектов, которые имеют некоторые общими свойства. Самоорганизующиеся сети (сети Кохонена) подразделяются на 2 типа: слой Кохонена (нейроны в нем не упорядочены) и карта Кохонена (нейроны упорядочены) [31].

На рисунке 1.10 приведена архитектура слоя Кохонена [31].

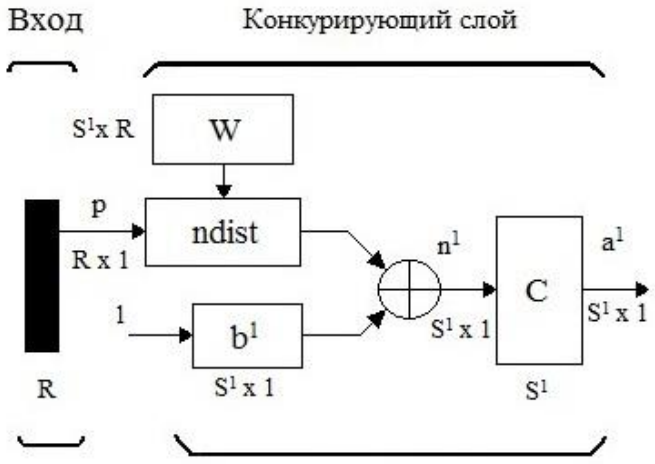

Рисунок 1.10 – Архитектура слоя Кохонена

Элемент ndist производит расчет евклидова расстояния между строками матрицы весов W и входным вектором p. Аргумент активационной функции n^1 представляет собой сумму вычисленного расстояния и вектора смещения b^1. В случае если все смещения равны нулю, то максимальное значение n^1 меньше, либо равно нулю. Вектор n^1 может иметь нулевые значения, в случае если вектор p является равным какому-либо вектору веса нейрона в сети. Значения большее 0 в элементах вектора n^1 возможны, в случае если смещения не равны нулю [36].

Как и любая другая функция активации, конкурирующая функция на основе значений элементов входного вектора формирует значения нейронов на выходном слое, однако при использовании данной функции значения всех нейронов (кроме нейрона-победителя) на выходном слое будут равны 0, значение нейрона-победителя будет иметь максимальное значение. Иными словами, на выходном слое a^1 все значения элементов равны 0, за исключением единственного элемента

(имеющего значение 1), который соответствует нейрону-победителю. Такая функция активации может быть описана следующим образом:

$$a_i^1 = \begin{cases} 1, \ i = i^*, i^* = arg(\max n_i^1) \\ 0, \ i \neq i^* \end{cases},$$

где i^* - номер активного нейрона, который определяет тот кластер, к которому наиболее близок входной вектор, а arg – аргумент конкурирующей функции активации.

Отличительной чертой процесса обучения данной сети является то, что необходимо настроить веса синапсов нейронов, а не минимизировать ошибку обучения. В случае когда при подаче входного вектора $p(q)$ на шаге обучения q нейрон i одерживает победу, в матрице весов IW в строке i происходит корректировка согласно правилу Кохонена:

$$W_i(q) = W_i(q-1) + \alpha\big(p(q) - W_i(q-1)\big) \qquad (1.1)$$

Это правило является рекуррентным соотношением, корректировка матрицы весов (а именно строки i) производится путем добавления взвешенной разности вектора входа и значения одноименного элемента строки на предыдущем шаге. По этому правилу вектор веса, имеющий наиболее близкие значения с входном вектором, корректируется так, чтобы дополнительно уменьшить расстояние между ними. Как следствие, обучение согласно приведенному ранее правилу приведет к тому, что нейрон-победитель с большей вероятностью выиграет конкуренцию в случае, когда на вход сети будет подан входной вектор схожий (близкий по расстоянию) с предыдущим, но вероятность его выигрыша будет снижена, когда существенно отличающийся от предыдущего входной вектор будет подан на вход сети. В процессе этого итерационного обучения на вход будут подаваться все большее векторов, а нейрон-победитель, будет вновь корректировать свой весовой вектор. Как результат, если в сети присутствует достаточное большое число нейронов, то каждый набор схожих векторов будет ассоциирован с одним из нейронов [31].

Архитектура карты Кохонена имеет схожую архитектуру, но при этом в ней нет элемента смещения [37].

В карте Кохонена используется аналогичный подход к определению нейрона-победителя, что и в слое. Однако в отличие от слоя, в карте Кохонена за одну итерацию происходит корректировка весовых коэффициентов нейрона-победителя, а также нейронов, попадающих в его окрестность. Происходит корректировка коэффициентов по формуле (1.1).

В окрестность нейрона-победителя входят все нейроны, которые расположены на расстоянии меньшем или равном радиусу d. В процессе обучения карты производится перераспределение нейронов, соседствующих с нейроном-победителем [31].

Сеть Кохонена, в отличие от большинства других сетей, предназначена именно для решения задачи кластерного анализа, однако можно выделить сходство этого вида сети с двунаправленной ассоциативной памятью, представляющей собой развитие нейронной сети Хопфилда.

Одной из основных задач двунаправленной ассоциативной памяти является распознавание образов, а также коррелирующая с ней задача восстановления «зашумленных» данных.

Упомянутое ранее сходство проявляется в соотнесении входного сигнала некоторому сохраненному образу: в сети Кохонена сохраненный образ будет представлять собой центр кластера, в двунаправленной ассоциативной памяти – некоторый нейрон с сохраненным объектом из обучающей выборки. В работе [38] авторы предложили свой подход к использованию двунаправленной ассоциативной памяти для решения задачи кластерного анализа, однако в их исследовании не были использованы крупные наборы данных (выборки, содержащие десятки тысяч объектов), а использованы небольшие наборы данных (максимальное число объектов – 252). В связи с этим двунаправленная ассоциативная память не содержит большого числа связанных нейронов, что означает сравнительно небольшой размер сети и, как следствие, вычислительные затраты на обучение такой сети не высоки. Однако в рамках диссертационного исследования был использован набор данных, имеющий большее число объектов, в результате именно сеть Кохонена является наиболее подходящей в этом случае.

В реальных задачах процесс обучения сети производится на самых различных экспериментальных данных, в том числе данных, имеющих большое число признаков. В этом случае наблюдается экспоненциальный рост количества необходимых экспериментальных данных при решении задач машинного обучения, зависящий от размерности пространства. В теории нейронных сетей данная проблема имеет название «проклятие размерности» [17].

Математически эту проблему можно описать следующим образом.

Пусть s – мера гладкости, определяемая как степень дифференцируемости функции (количество существующих производных). Тогда для обычной гладкой функции минимаксная скорость сходимости общего риска R имеет порядок $(1/N)^{2s/(2s+m_0)}$.

Термин «проклятие размерности» был введен Ричардом Беллманом в 1961 году в работе, посвященной процессам адаптивного управления [17].

«Проклятие размерности» вносит следующие проблемы в процесс обучения сетей на многомерных данных:

- Трудоемкость вычислений.
- Необходимость хранения огромного количества данных.
- Увеличение доли шумов.
- В линейных классификаторах увеличение числа признаков ведет к проблемам переобучения.
- В метрических классификаторах расстояния обычно вычисляются как средний модуль разностей по всем признакам. Согласно закону больших чисел, сумма n слагаемых стремится в некоторому фиксированному пределу при $n \to \infty$. Таким образом, расстояния во всех парах объектов стремятся к одному и тому же значению, а значит, становятся неинформативными [17].

Резюмируя описанные выше пункты, можно сказать, что из-за «проклятия размерности» процесс обучения может занять довольно продолжительное время в силу увеличения числа итераций обучения, добавления новых скрытых слоев в

разрабатываемую нейронную сеть, но иногда из-за высокой размерности входных данных обучение сети и вовсе не представляется возможным.

Основная ключ к решению проблемы «проклятия размерности» - это снижение размерности признакового пространства, а именно спроецировать данные на подпространство меньшей размерности без существенных потерь информации о предметной области, для этого используются методы редукции данных.

1.5 Методы редукции данных

На сегодняшний день существует достаточно большое число методов редукции данных, которые относятся к одной из следующих групп [39]:

- автокодировщики;
- факторный анализ;

Простейшая архитектура автокодировщика - сеть прямого распространения без обратных связей, наиболее схожая с персептроном и содержащая входной слой, промежуточный скрытый слой и выходной слой [40].

Выходной слой автокодировщика должен содержать столько же нейронов, сколько и входной слой, а скрытый слой должен содержать число нейронов, меньшее числа нейронов в входном (выходном) слое.

В общем случае автокодировщик состоит из двух частей: кодера и декодера.

Автокодировщик способен обучиться воспроизведению входного сигнала на выходном слое сети. Эту сеть обучают, используя парадигму обучения без учителя, однако процесс обучения основан на оптимизации функции стоимости (cost function) [41]. Эта функция определяет ошибку L между входным вектором x и воспроизведенным выходным вектором $\hat{x}$:

$$L = |x - \hat{x}| \tag{1.2}$$

Подробнее принцип обучения автокодировщика можно описать следующим образом:

1. На вход сети поступает вектор $x = (x_1, x_2, \dots, x_n)$, происходит кодирование входного вектора в вектор $z = (z_1, z_2, \dots, z_m)$ меньшей размерности $(m < n)$, согласно следующему уравнению:

$$z = h(Wx + b_1),$$

где h - функция активации кодера, W – матрица весов, b_1 – смещение в кодере.

2. Декодер принимает на вход вектор z и восстанавливает исходный вектор x:

$$\hat{x} = h(Wz + b_2),$$

где h - функция активации декодера, b_2 – смещение в декодере.

3. По формуле (1.2) вычисляется ошибка.

4. С помощью метода обратного распространения ошибки происходит обновление весов сети.

Пункты 1-4 повторяются заданное число итераций, либо пока градиент функции не достигнет заданного минимума [42].

Графически принцип обучения автокодировщика приведен на рисунке 1.11.

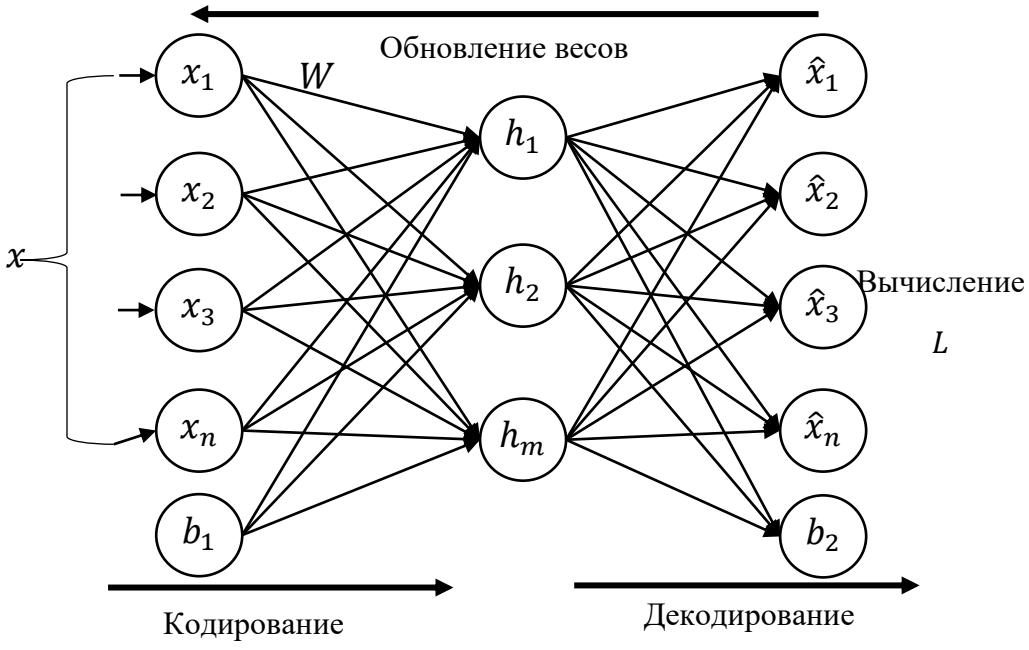

Рисунок 1.11 – Схема обучения автокодировщика

Факторный анализ является частью многомерного статистического анализа, объединяющей методы сокращения размерности множества наблюдаемых переменных посредством исследования структуры ковариационных или корреляционных матриц [43].

При анализе структуры ковариационных и корреляционных матриц чаще всего используются два подхода: факторный анализ и метод главных компонент (компонентный анализ). Метод главных компонент является одним из методов факторного анализа.

В факторном анализе предполагается, что наблюдаемые переменные (признаки) являются линейной комбинацией некоторых латентных (скрытых) факторов. Некоторые из этих факторов зависят от для двух и более переменных, а другие – характерными для каждой переменной в отдельности.

В литературе, а именно в книге [44], даются следующие определения терминов факторного анализа.

Факторы – непосредственно неизмеряемые, скрытые переменные, в терминах которых описываются измеряемые переменные.

Общий фактор – неизмеряемая скрытая величина, которая учитывает корреляцию по крайней мере между двумя наблюдаемыми переменными.

Характерный фактор – фактор, влияющий только на данную переменную.

Значение фактора – оценка скрытого фактора в терминах наблюдаемых переменных.

Факторная нагрузка – общий термин, означающий коэффициенты матрицы факторного отображения или структуры.

Общность – доля дисперсии наблюдаемых переменных, обусловленная общими факторами.

Характерность – доля дисперсии наблюдаемой переменной, не связанная с общими факторами и свойственная именно данной переменной; разделяется на специфичность и дисперсию ошибки.

Специфичность – доля дисперсии наблюдаемой переменной, соответствующая специфичному фактору.

Дисперсия ошибки – часть дисперсии наблюдаемой переменной.

Основная модель факторного анализа имеет следующий вид [45]:

$$z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jm}F_m + d_jU_j \ (j = 1,2,\cdots,n)$$

Переменная z_j линейно зависит от m общих факторов и характерного фактора. Общие факторы учитывают корреляции между переменными, характерный фактор учитывает оставшуюся дисперсию. Коэффициенты при факторах называют нагрузками. Считают, что характерные факторы независимы как между собой, так и по отношению к общим факторам.

Задача факторного анализа состоит в том, чтобы выразить переменную z_j в терминах скрытых факторов.

Модель компонентного анализа [45]:

$$z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jn}F_n \ (j = 1,2,\cdots,n),$$

где каждая из наблюдаемых переменных линейно зависит от n некоррелированных между собой новых компонентов $F_1, F_2, \cdots, F_n$.

Важное свойство метода состоит в том, что каждая очередная компонента дает максимально возможный вклад в суммарную дисперсию переменных.

Для получения оценок факторных нагрузок и остаточных дисперсий могут быть применены следующие методы факторного анализа:

- метод главных факторов;
- метод максимального правдоподобия;
- групповой метод;
- центроидный метод;
- метод минимальных остатков;
- альфа-факторный анализ.

На сегодняшний день при проведении факторного анализа используют метод максимального правдоподобия [45].

Метод максимального правдоподобия сводится к решению характеристического уравнения, которое может быть представлено в виде:

$$\Delta(R_2 - \lambda I) = 0$$

R_2 определяется соотношением вида:

$$R_2 = U^{-1}(R - U^2)U^{-1} = U^{-1}R_1U^{-1},$$

где U^2 – оценка дисперсии характерных переменных.

В вычисляемую на каждом шаге оценку общностей с большим весом входят корреляции с переменными, имеющими меньшую характерность.

Важной составляющей факторного анализа является процедура вращения факторов. Данная процедура производится после того, как найдены факторные нагрузки одним из приведенных ранее методов.

Всего существует два типа вращения, графическое представление которых приведено на рисунках 1.12-1.13 [46]:

1. Ортогональное (квартимакс, варимакс, эквимакс методы).

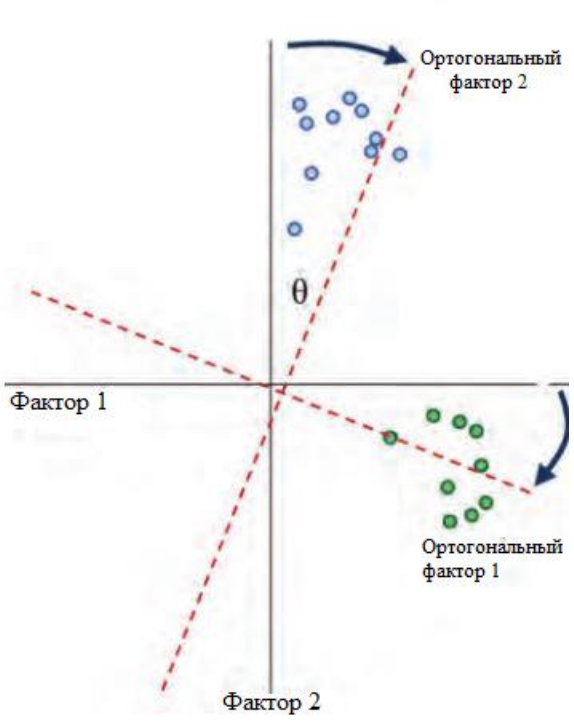

Рисунок 1.12 – Ортогональное вращение для двухфакторной модели факторного анализа

2. Косоугольное (облимакс, облимин, квартимин методы).

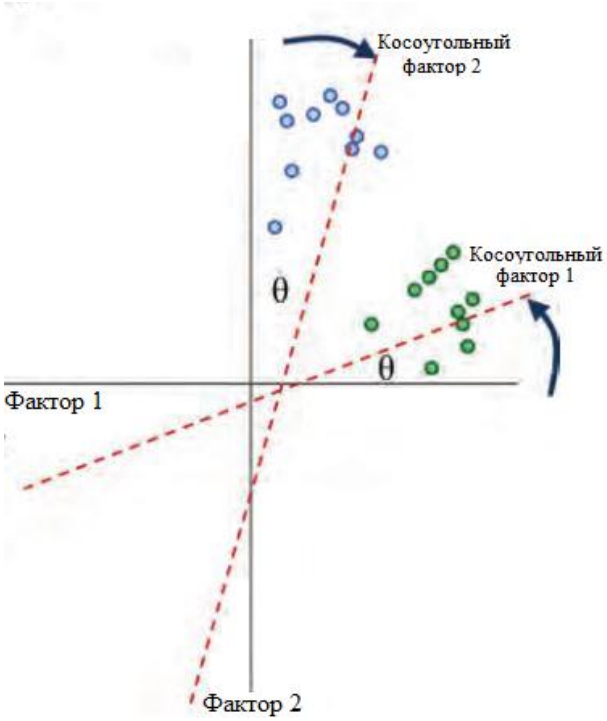

Рисунок 1.13 – Косоугольное вращение для двухфакторной модели факторного анализа

Наиболее часто используемым вычислительным методом вращения является варимакс (varimax), предложенный Кайзером [47], который максимизирует разброс квадратов нагрузок для каждого фактора, что приводит к увеличению больших и уменьшению малых значений факторных нагрузок.

Согласно Кайзеру простота фактора p определяется дисперсией квадратов его нагрузок, т.е.:

$$s_p^2 = \frac{1}{n}\sum_{j=1}^{n}\left(b_{jp}^2\right)^2 - \frac{1}{n^2}\left(\sum_{j=1}^{n} b_{jp}^2\right)^2 \quad (p = 1,2,\cdots,m)$$

Если эта дисперсия максимальна, то фактор наилучшим образом интерпретируем, поскольку при этом его нагрузки близки в основном к единице или к нулю [47].

Критерий максимизации полной факторной матрицы сводится, следовательно, к максимизации суммы величин по всем факторам:

$$s^2 = \sum_{p=1}^{m} s_p^2$$

Этот критерий Кайзер назвал «нормальным». На сегодняшний день применяется улучшенный критерий варимакс, в котором требуется, чтобы финальные нагрузки отвечали максимуму функции [39]:

$$V = n\sum_{p=1}^{m}\sum_{j=1}^{n}\left(\frac{b_{jp}}{n_j}\right)^4 - \sum_{p=1}^{m}\left(\sum_{j=1}^{n}\frac{b_{jp}^2}{n_j^2}\right)^2$$

Основной сложностью факторного анализа является определение оптимального числа факторов. Существует несколько часто употребляемых критериев определения числа факторов. Некоторые из них являются альтернативными по отношению к другим, а часть этих критериев можно использовать совместно с тем, чтобы один дополнял другой. Наиболее широко применяемые критерии:

1. Критерий Кайзера или критерий собственных чисел. Этот критерий предложен Кайзером, и является, вероятно, наиболее широко используемым. Отбираются только факторы с собственными значениями, равными или большими 1. Это означает, что если фактор не выделяет дисперсию, эквивалентную, по крайней мере, дисперсии одной переменной, то он опускается [44]. Собственные значения факторов вычисляются на основе значений корреляционной матрицы.

2. Критерий доли воспроизводимой дисперсии. Факторы ранжируются по доле воспроизводимой дисперсии, когда процент дисперсии оказывается несущественным, выделение следует остановить [44].

Исследование по применимости изложенных в параграфах 1.1-1.5 подходов к решению задач классификации и кластеризации, а также подходов к последующей оценке качества полученных решений целесообразно проводить в экономически важных отраслях экономики и промышленности.

Выводы по главе 1

1. Показана актуальность решения задач классификации и кластеризации на основе применения нейронных сетей различных видов. На основе анализа научных публикаций на тему оценки эффективности различных алгоритмов выделен один из наиболее эффективных алгоритмов – алгоритм Левенберга-Марквардта.

2. Приведены способы оценки качества решения задач классификации и кластеризации, в том числе проанализированы международные научные работы по успешному применению малоизвестных в отечественных работах индексов Рэнда.

3. Раскрывается необходимость применения редукции данных в задачах интеллектуального анализа данных, решаемых при помощи нейронных сетей.

Все рассмотренные в параграфах 1.1-1.5 методы интеллектуального анализа актуальны для выполнения цели диссертационного исследования.

Однако совместное использование большого числа приведенных методов является нетривиальной задачей, что явилось обоснованием для разработки СППР, позволяющей исследователям проводить обработку и анализ данных с различными методами.

Глава 2. Проектирование и разработка системы для классификации и кластеризации технических объектов

2.1 Проектирование СППР. Определение структурной схемы системы. Анализ потоков данных в системе

На данный момент практически во всех сферах человеческой деятельности людям ежедневно требуется принимать решения различной сложности. Во избежание принятия неверных решений применяют СППР (системы поддержки принятия решений).

В общем случае СППР предназначена для поддержки многокритериальных решений в сложной информационной среде. Под многокритериальностью подразумевается, что решения принимаются не по одному, а по множеству критериев.

Согласно [48], СППР – это интерактивные автоматизированные системы, помогающие лицу, принимающему решения, использовать данные и модели для решения слабоструктурированных проблем.

СППР решают две основные задачи:

- выбор наилучшего решения из множества возможных;
- упорядочивание (ранжирование) возможных решений.

В ряде случаев в СППР активно применяются методы интеллектуального анализа данных, благодаря которым такие системы успешно решают вышеперечисленные задачи. Такие системы называются ИСППР (Интеллектуальные системы поддержки принятия решений).

Поскольку СППР являются достаточно сложными системами, то для их проектирования используют системный анализ [49-51].

Методика системного анализа разрабатывается и применяется в тех случаях, когда у лиц, принимающих решения, на начальном этапе нет достаточных сведений о проблемной ситуации, позволяющих выбрать метод ее формализованного представления, сформировать математическую модель или

применить один из новых подходов к моделированию, сочетающих качественные и количественные приемы [52].

Основой системного анализа считают общую теорию систем и системный подход. На их основе строятся исходные представления и предпосылки. При проведении системного анализа значительную роль играет структура системы, то, есть ее декомпозиция на составные части.

В рамках диссертационной работы на этапе проектирования СППР была определена структура системы, состоящей из нескольких частей, отображенная в виде диаграммы активностей на рисунке 2.1. Диаграмма была построена в бесплатном онлайн-сервисе draw.io от компании JGraph Ltd (https://www.draw.io/). В ней выделены 4 составные части: препроцессинга; редукции данных; обучения и тестирования нейронных сетей; экспорта результатов анализа. Каждая составная часть является подсистемой, поскольку все они обладают важным свойством системы - имеют подцель (подцели), на которое ориентирована подсистема.

Согласно теории систем, каждая подсистема имеет свою цель.

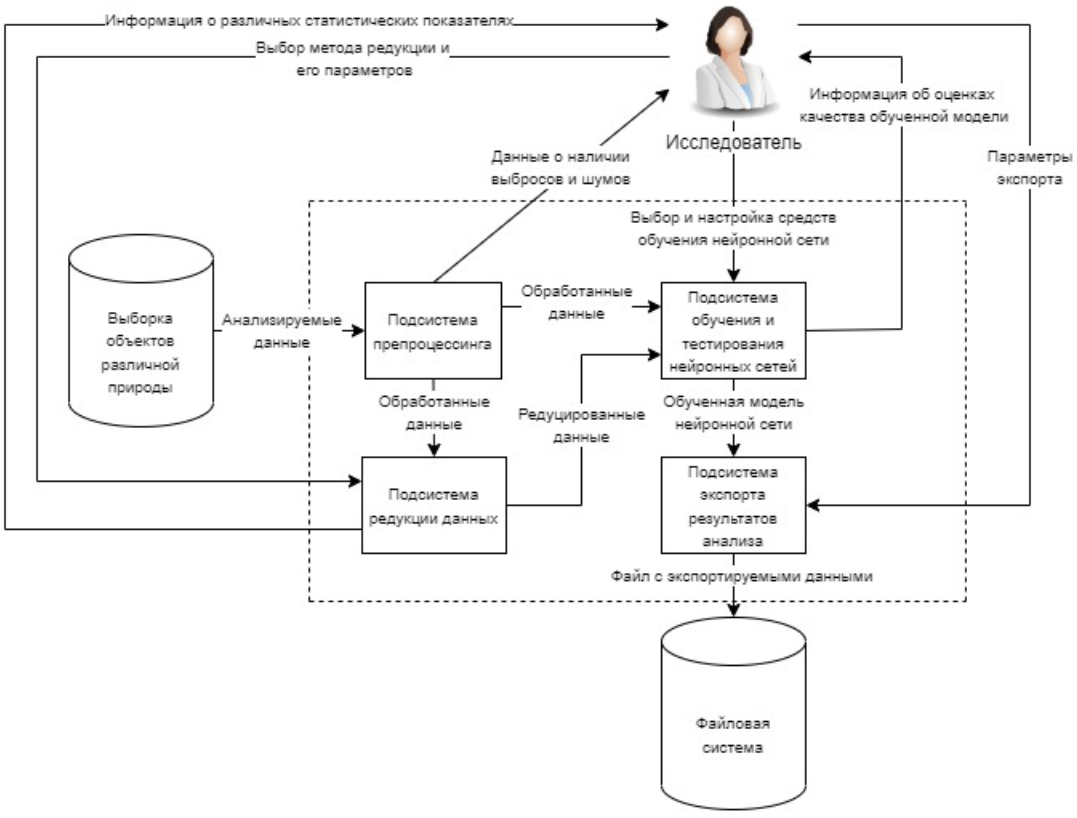

Рисунок 2.1 – Структурная схема СППР

Подсистема препроцессинга имеет следующие цели: проведение проверки на валидность формата анализируемых данных, обеспечение возможности селекции необходимых для анализа признаков.

Цель подсистемы редукции данных заключается в предоставлении исследователю информации о различных статистических показателях, касающихся анализируемых данных с целью оказания поддержки в выборе метода редукции и определении его параметров.

Подсистема обучения и тестирования нейронных сетей имеет следующие цели: предоставить исследователю доступ к созданию нейронных сетей для решения задач классификации и кластеризации, а также обеспечить возможность обучения этих сетей с предоставлением различных оценок качества решения этих задач.

Целью последней подсистемы является обеспечение экспорта результатов анализа в общепринятые форматы обмена информацией с целью обеспечения взаимодействия разрабатываемой системы с другими.

В свою очередь целесообразна дальнейшая декомпозиция разрабатываемой подсистемы обучения и тестирования нейронных сетей на элементы и определение связей между ними.

1. Обучающий набор данных может использоваться для решения задач классификации и кластеризации.
2. В случае решения задачи кластеризации применяется разработанная методика, описанная в параграфе 2.2.
3. В случае решения задачи классификации выбор архитектуры сети из представленных и ее конфигурация (изменение числа слоев, нейронов, активационных функций).
4. Обучение и тестирование сети.
5. Отправка информации исследователю о качестве решения задачи.

После определения структурной схемы системы были более детально описаны процессы, происходящие в ней, в особенности в подсистеме обучения и тестирования нейронных сетей, поскольку именно в ней реализована новая

методика инициализации весов слоя Кохонена. Детальное описание процессов позволило лучше изучить поведение разрабатываемой системы. Для этого часто используют различные методологии:

- IDEF0 – методология структурного анализа и проектирования;
- IDEF3 – методология сбора данных, необходимых для проведения структурного анализа системы;
- IDEF5 – методология онтологического описания данных;
- DFD (Data Flow Diagram) – методология структурного анализа потоков данных.

Так как проектируется СППР, активно использующая множество данных, то наиболее релевантным является использование методологии DFD.

Диаграммы DFD позволяют описать процесс обмена информацией между элементами изучаемой подсистемы, чего нельзя выполнить с помощью методологий IDEF [53].

На рисунке 2.2 изображена диаграмма DFD, построенная для соответствующей СППР, изображенной на рисунке 2.1.

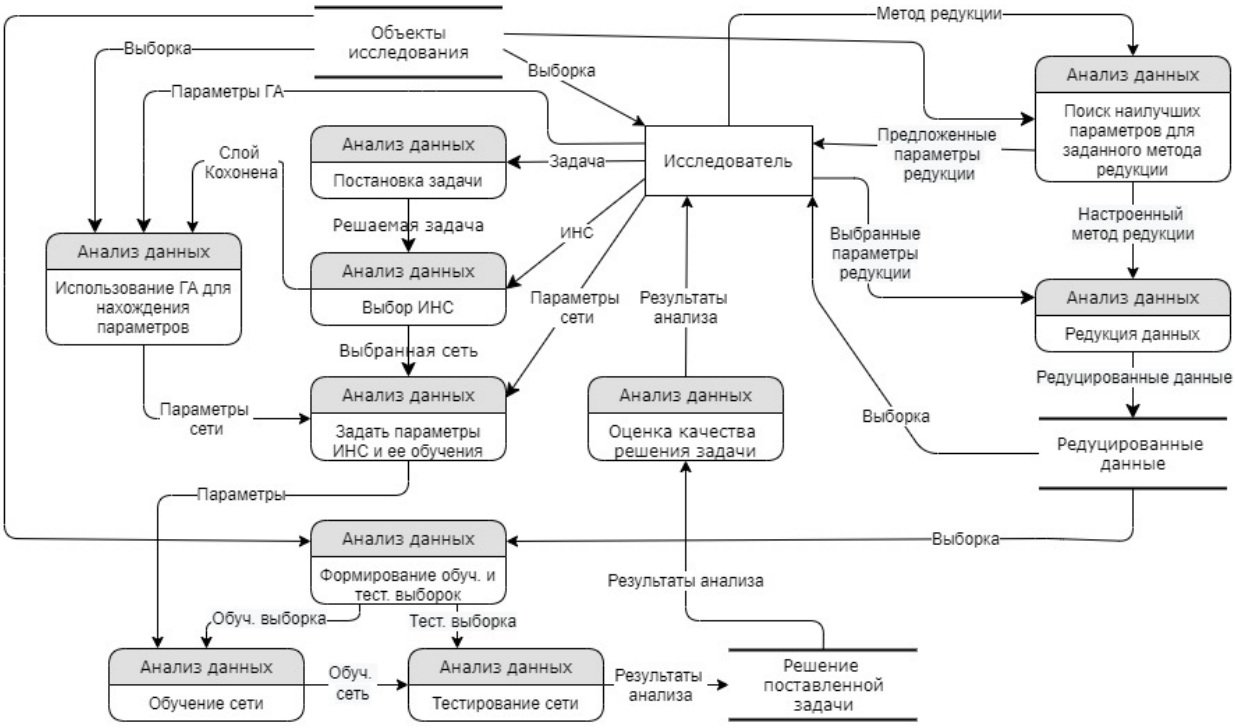

Рисунок 2.2 – Диаграмма DFD для СППР

DFD отображает источники и адресаты данных, идентифицирует процессы (изображены в виде прямоугольников с закругленными углами), связывающие в потоки одну функцию с другой, а также, что важно, определяет хранилища данных (прямоугольники без боковых границ), которые используются в исследуемом процессе. Внешние сущности обеспечивают необходимые входы для системы и/или являются приемниками для ее выходов (прямоугольники). Стрелки описывают поток объектов от одной части системы к другой.

Центральным элементом в ней является внешняя сущность – исследователь, которая взаимодействует с этой подсистемой. Также используются 3 хранилища:

- объекты исследования (выборка технических объектов);
- редуцированные данные (выборка меньшей размерности);
- результат решения поставленной задачи (числовой вектор номеров классов для каждого объекта).

Структурный анализ потоков данных (DFD) заложил фундамент для проектирования системы СППР, в виде программного обеспечения, аппаратом сетей Петри.

2.2 Разработка методики применения генетического алгоритма совместно с нейронной сетью Кохонена

Эвристический алгоритм – это алгоритм решения задачи, точность которого не доказана для всех возможных случаев, имеющих место в процессе решения задачи, но, тем не менее, известно, что данный алгоритм дает решение, достаточно близкое к точному в большинстве случаев [54]. Алгоритм можно применять, если при этом он дает неверный результат только в отдельных, достаточно редких и хорошо выделяемых случаях, или же он дает, хотя и не абсолютно точный, но все же приемлемый результат [54-56]. Иными словами, эвристический алгоритм – это не полностью математически обоснованный, но

при этом практически значимый алгоритм, в подавляющем большинстве случаев дающий решение, достаточно близкое к точному [55].

Эвристический алгоритм, в отличие от точного алгоритма, обладает тремя характерными особенностями:

1) вероятность получения абсолютно точного результата при использовании данного алгоритма не равна единице;

2) в некоторых частных случаях алгоритм может привести к неверному результату;

3) в исключительных случаях возможен «пропуск цели», то есть решение не будет найдено, даже если известно, что оно заведомо существует [54].

На текущий момент эвристические алгоритмы представляют собой широкое множество различных алгоритмов, среди которых важное место для анализа данных и систем искусственного интеллекта занимают генетические алгоритмы.

Основоположник генетических алгоритмов Джон Холланд и его коллеги, основываясь на принципах естественного отбора, сформулированных Чарльзом Дарвином, реализовали первый генетический алгоритм, который позже Холланд описал в своей книге [57].

Как известно, принцип естественного отбора заключается в том, что в конкурентной борьбе выживает наиболее приспособленный. Для этого используется функция приспособленности (fitness function). Эта функция позволяет оценить степень приспособленности конкретных особей в популяции.

Генетические алгоритмы отличаются от других оптимизационных и поисковых процедур следующим [58]:

- Используют целевую функцию (функцию приспособленности), а не ее различные приращения для оценки качества принятия решений;
- Применяют не детерминированные, а вероятностные правила анализа оптимизационных задач.

В генетических алгоритмах используется ряд терминов, заимствованных из генетики, прежде всего: ген, хромосома, особь, популяция.

Хромосома – упорядоченная последовательность генов.

Ген – атомарный элемент хромосомы.

Популяция – конечное множество особей.

Особь – конечное множество хромосом.

Одно пробное решение называется особью, а набор всех пробных решений – популяцией.

Основные принципы работы генетического алгоритма (ГА) заключены в следующей схеме (рисунок 2.3) [59].

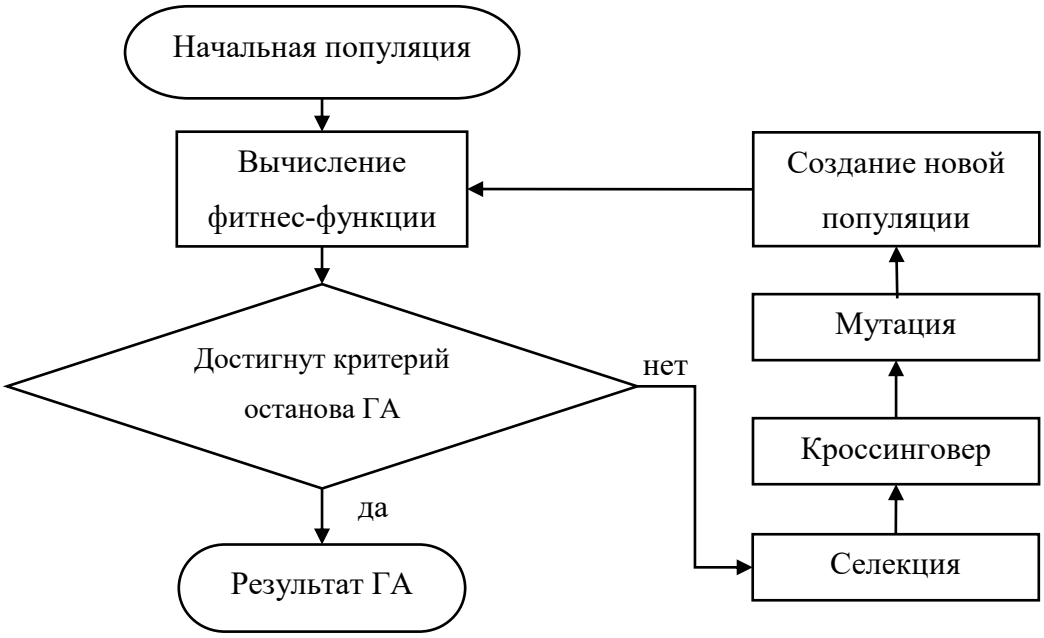

Рисунок 2.3 – Блок-схема генетического алгоритма

Из приведенной блок-схемы видно, что вычисление функции приспособленности производится для каждой генерируемой популяции, начиная с начальной. От полученных значений функции приспособленности зависит дальнейшая работа генетического алгоритма: если критерий останова удовлетворен, то алгоритм прекращает работу; если критерий не был достигнут, то на основе значений функции приспособленности создается новая популяция и цикл повторяется.

В генетическом алгоритме создание новой популяции реализуются путем применения селекции и генетических операторов: оператора скрещивания и оператора мутации.

В книге [59] дается следующее определение термина генетический оператор.

Генетический оператор – это конструкция, представляющая один шаг из последовательности действий генетического алгоритма.

Существует несколько видов методов селекции [59]:

• Метод селекции основанный на принципе колеса рулетки считается для генетических алгоритмов основным методом отбора особей для родительской популяции. Несмотря на случайный характер процедуры селекции, родительские особи выбираются пропорционально значениям их функции приспособленности, т.е. согласно вероятности селекции

$$p_s(ch_i) = \frac{F(ch_i)}{\sum_{i=1}^{N} F(ch_i)},$$

где N – численность популяции, $F(ch_i)$ – значение функции приспособленности хромосомы ch_i, а $p_s(ch_i)$ – вероятность селекции хромосомы ch_i.

• Турнирный метод. При турнирной селекции все особи популяции разбиваются на подгруппы с последующим выбором в каждой из них особи с наилучшей приспособленностью.

• Ранговая селекция. Особи популяции ранжируются по значениям их функции приспособленности. При такой селекции популяция представляет собой отсортированный список особей, упорядоченных по направлению от наиболее приспособленных к менее, в котором каждой особи приписывается число (ранг), определяющее ее место в списке.

Процесс размножения (кроссиговера) состоит в селекции родительских пар для скрещивания таким образом, чтобы решения (особи) в новой популяции были ближе к искомому глобальному минимуму функции приспособленности.

Следующим шагом в работе генетического алгоритма являются мутации, то есть случайные изменения полученных в результате скрещивания хромосом. Мутации способны улучшить или ухудшить приспособленность особи-потомка.

Оператор мутации обычно состоит из двух этапов [59]:

1. В хромосоме $A = (a_1, a_2, , \cdots, a_{L-1}, a_L)$ определяются случайным образом две позиции.

2. Гены, соответствующие выбранным позициям переставляются, и формируется новая хромосома $A' = (a_1, a_{L-1}, \cdots, a_2, a_L)$.

Совместное использование генетических алгоритмов и нейронных сетей известно в литературе под аббревиатурой COGANN (Combinations of Genetic Algorithms and Neural Networks).

На текущий момент существует три подхода к COGANN [60]:

• Независимый - генетические алгоритмы и нейронные сети по отдельности решают ту или иную задачу, при этом они никак не взаимодействуют друг с другом.

• Вспомогательный генетические алгоритмы и нейронные сети применяются последовательно, решая ту или иную проблему, которую не представлялось бы возможным разрешить, применяя каждый из этих способов по отдельности. Например, генетический алгоритм может провести анализ обученной сети, либо выполнить поиск оптимального набора параметров для обучения сети или нейронная сеть может сформировать исходную популяцию для генетического алгоритма.

• Равноправный – данный подход к совместному использованию предполагает, что реализуемые нейронная сеть и генетический алгоритм прочно связаны друг с другом, вплоть до того, что заменяются классические алгоритмы обучения сети на эволюционное обучение сети (эволюцию весов связей). Возможен и обратный случай, в котором нейронная сеть будет заменять некоторые классические операторы генетического алгоритма (селекцию, скрещивание, мутацию) [61].

В рамках диссертационной работы проводились исследования, в которых для нейронной сети Кохонена (слоя Кохонена) проводилась предварительная настройка параметров сети (матрицы весов) посредством применения генетического алгоритма, реализуя тем самым вспомогательный подход.

Предварительная настройка матрицы весов полезна в том случае, когда на результат кластерного анализа оказывает сильное влияние начальное определение матрицы весов сети (в случае слоя Кохонена – координат центров кластеров). В особенности это касается случаев, когда анализируется выборка с высокой размерностью [62-66].

Широко используются следующие стандартные общепринятые способы определения начальных координат центров кластеров слоя Кохонена:

- случайный выбор k наблюдений, где k – число нейронов скрытого слоя;
- каждая координата равна среднему арифметическому значений соответствующего признака по всей обучающей выборке.

Основной недостаток при использовании приведенных стандартных способов: качество решения задачи кластерного анализа может не соответствовать требуемому уровню.

Для решения этой проблемы автором была разработана методика определения начальных координат центров кластеров слоя Кохонена с применяем генетического алгоритма, основанная на принципах работы алгоритма К-средних. Выбор данного алгоритма обоснован тем, что принцип работы слоя Кохонена во многом схож с принципами работы алгоритма К-средних. Дополнительно, как показало проводимое в рамках диссертационной работы исследование, описанное в параграфе 3.3.1, полученные результаты решения задачи кластерного анализа с помощью приведенных алгоритмов схожи. Таблица 2.1 содержит информацию о сравнении работы алгоритмов К-средних и слоя Кохонена при решении задачи кластерного анализа.

Таблица 2.1 – Сравнение алгоритмов работы слоя Кохонена и К-средних

Критерий сходства	Слой Кохонена	Алгоритм К-средних
Предварительное определение числа кластеров	Обязательно предварительное определение числа кластеров (число нейронов в выходном слое)	Обязательно предварительное определение числа кластеров
Определение центра кластеров	Координаты случайного объекта из выборки	Координаты случайного объекта из выборки
Отнесение объекта к кластеру	Активация нейрона-победителя в выходном слое на основе расстояний от анализируемого объекта	Основано на выборе минимального расстояния до всех центров кластеров от анализируемого объекта

В случае К-средних распространен критерий – минимизация суммы квадратов расстояний от точек до центров кластеров, к которым они относятся [8]. Иными словами, задачу кластерного анализа методом К-средних можно свести к задаче минимизации функции

$$F = \sum_{i=1}^{k} \sum_{x_j \in S_i} \left(x_j - c_i\right)^2,$$

где k - число кластеров, S_i - полученные кластеры (подмножества множества всех анализируемых наблюдений), x_j - наблюдения, c_i - центры кластеров [8]. Именно эта функция в рамках реализованного подхода является целевой для генетического алгоритма.

На основании приведенного критерия минимизации была сформирована функция приспособленности. В данном случае целью генетического алгоритма является минимизация приведенной выше функции F и нахождение ее глобального экстремума.

Графически реализованный подход удобно отобразить в виде диаграммы активности (рисунок 2.4).

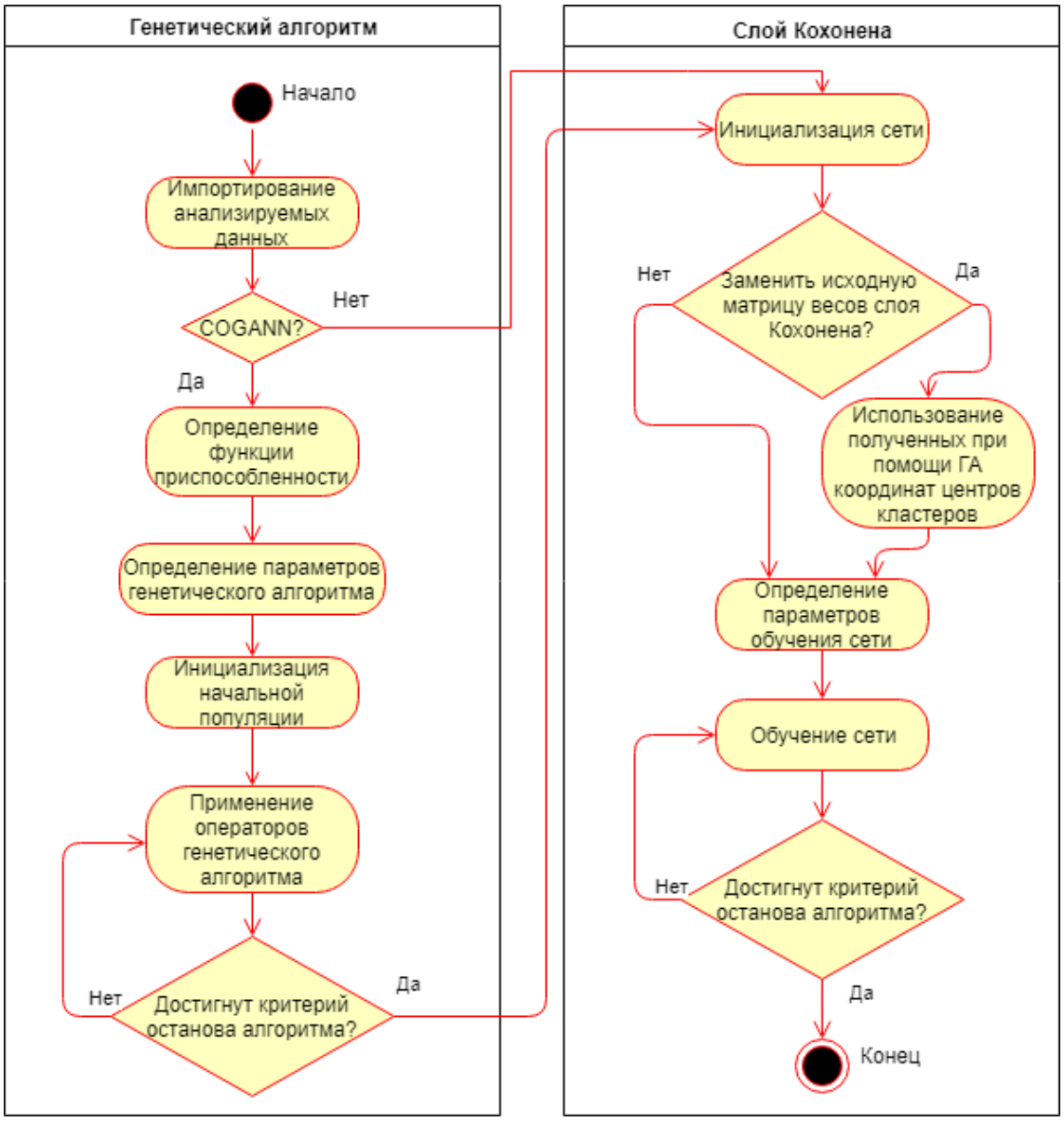

Рисунок 2.4 – Реализованный подход COGANN для слоя Кохонена

Однако сравнение принципов работы алгоритма K-средних с принципами работы генетического алгоритма в качестве алгоритма обучения слоя Кохонена выявляет ряд особенностей (таблица 2.2).

Таким образом, было выявлено, что оба алгоритма имеют 3 общих этапа работы. На каждом этапе генетический алгоритм, в отличие от алгоритма K-средних, оперирует популяциями.

Таблица 2.2 – Сравнение этапов работы алгоритмов

Этап	Алгоритм K- средних	Генетический алгоритм
Начало работы алгоритма	Определение координат центров кластеров	Формирование начальной популяции (вектор, содержащий координаты центров кластеров)
Проверка останова алгоритма	Координаты центров кластеров на текущей итерации не отличаются от соответствующих координат на предыдущей итерации алгоритма	Формирование новой популяции не приводит к значимому улучшению значения фитнес-функции
Изменение координат центров кластеров	Перерасчет координат происходит по четко определенной формуле	Выполняется ряд операций – селекция, скрещивание, мутация и формирование новой популяции – нового вектора, содержащего координаты центров кластеров

Как уже говорилось ранее, ключевыми терминами в генетическом алгоритме являются: популяция, особь, хромосома, ген. Таким образом, в данном случае:

- популяция представляет собой некоторое конечное множество объектов в n-мерном пространстве, являющееся потенциальными центрами кластеров;
- особь – некоторый потенциальный центр кластера, описываемый набором координат в n-мерном пространстве;
- хромосома – одна из координат потенциального центра кластера в десятичном формате;
- ген – атомарная часть в двоичном представлении значения хромосомы (координаты потенциального центра кластера).

2.3 Использование аппарата сетей Петри для проектирования и анализа разрабатываемой системы

Проектирование систем связано с синтезом оптимального состава модулей программного обеспечения на этапе технического проектирования программной системы. В большинстве случаев такие системы содержат в себе достаточно большое число модулей, некоторым образом взаимодействующих друг с другом. В результате возникает необходимость в разработке модели системы, четко описывающей взаимодействие модулей, входящих в эту систему.

Одним из способов моделирования разрабатываемых систем является формализация их функционирования в виде сетей Петри. Можно построить такую модель, описывающую функционирование любой системы, алгоритма или процесса.

В самом общем случае сетью Петри называется совокупность множеств

$$C = \{P, T, I, O\}$$

P – конечное множество, элементы которого называются позициями;

T – конечное множество, элементы которого называются переходами;

I – множество входных функций, $I : T \to P$;

O – множество выходных функций, $O : T \to P$.

Для моделирования сети необходимо иметь маркировку сети μ. Маркировка μ определяется как вектор $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ размерности n, где $n = |P|$ и каждое $\mu_i \in N, i = 1, \ldots, n$. Вектор μ определяет для каждой позиции p_i сети Петри количество меток в этой позиции [67].

Маркированная сеть Петри может быть записана в виде: $M = (C, \mu)$ или $M = (P, T, I, O, \mu)$.

Сеть Петри моделируется посредством запуска переходов. Переход запускается удалением меток из его входных позиций и образованием новых меток, помещаемых в его выходные позиции. Следовательно, при моделировании сети маркировка сети будет изменяться.

Переход может запускаться только в том случае, если он разрешен. Переход $t_j \in T$ в маркированной сети Петри $C = \{P, T, I, O\}$ с маркировкой μ разрешен, если для всех $p_i \in P$

$$\mu(p_i) \geq \#(p_i, I(t_j))$$

Переход запускается удалением всех разрешающих меток из его входных позиций и последующим помещением по одной метке в каждую из его выходных позиций [68].

Простое представление системы сетью Петри основано на двух основополагающих понятиях: событиях и условиях. События – это действия, имеющие место в системе. Условие есть предикат, либо логическое описание состояния системы. В сети Петри условия моделируются позициями, события – переходами. Возникновение события равносильно запуску соответствующего перехода [68].

Для моделирования системы аппаратом сетей Петри необходимо определить ее вершины: позиции и переходы.

На основе принципов построения СППР, определенных в параграфе 2.2, проектируемая система должна соответствовать следующим требованиям:

- поддержка процесса редукции данных, с применением факторного анализа и автокодировщика;
- построение нейронной сети прямого распространения и ее конфигурирование для решения задачи классификации;
- возможность использования разработанной методики, описанной в параграфе 2.3;
- отображение результатов анализа в виде таблиц и графиков;
- формирование отчетных документов, содержащих полученные результаты.

Исходя из вышеперечисленных условий, были определены вершины сети Петри и связи между ними.

Графически позицию принято обозначать овалом, переход – прямоугольником. На рисунках 2.5а-2.5б представлена сеть Петри для

проектируемой системы, которая содержит 52 вершины. В начальном состоянии сети лишь одна позиция сети «Выбор в главном меню программы» содержит метку.

В большинстве случаев описания проектируемой системы аппаратом сетей Петри недостаточно – необходимо выполнить анализ построенной сети, который дает представление о ее поведении [69].

Анализ сети производился в программной среде CPN Tools версии 4.0.1 [70]. Данная среда обладает автоматизированными средствами анализа и генерирует отчеты о тупиковых состояниях системы, мертвых переходах. Анализ, проводимый в этой среде, по своей сути решает одну их основных задач теории сетей Петри – задачу достижимости.

Формально задача достижимости состоит в следующем: для сети Петри C с начальной маркировкой μ_0 и заданной маркировкой μ установить справедливость включения $\mu \in R(C, \mu_0)$ [68].

Иными словами, требуется выяснить, существует ли допустимая последовательность срабатываний переходов, переводящая сеть Петри из начальной маркировки в заданную [71,72].

В отличие от многих других программных продуктов, реализующих логику математического аппарата сетей Петри и обеспечивающих пользовательский интерфейс для работы с ними, в приведенной версии среды CPN Tools поддерживается автоматизированное построение деревьев достижимости [73].

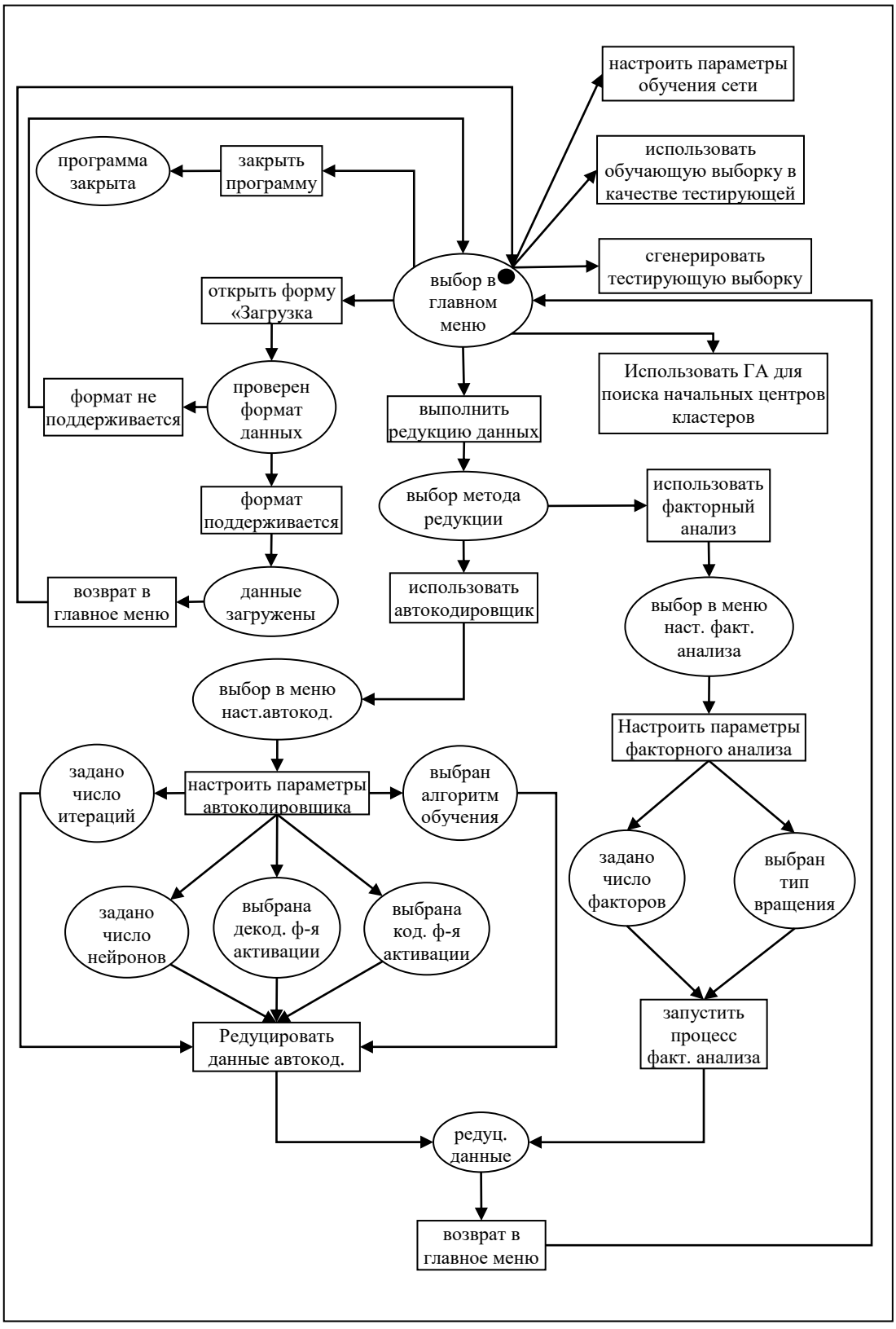

Рисунок 2.5а – Сеть Петри, описывающая проектируемую систему

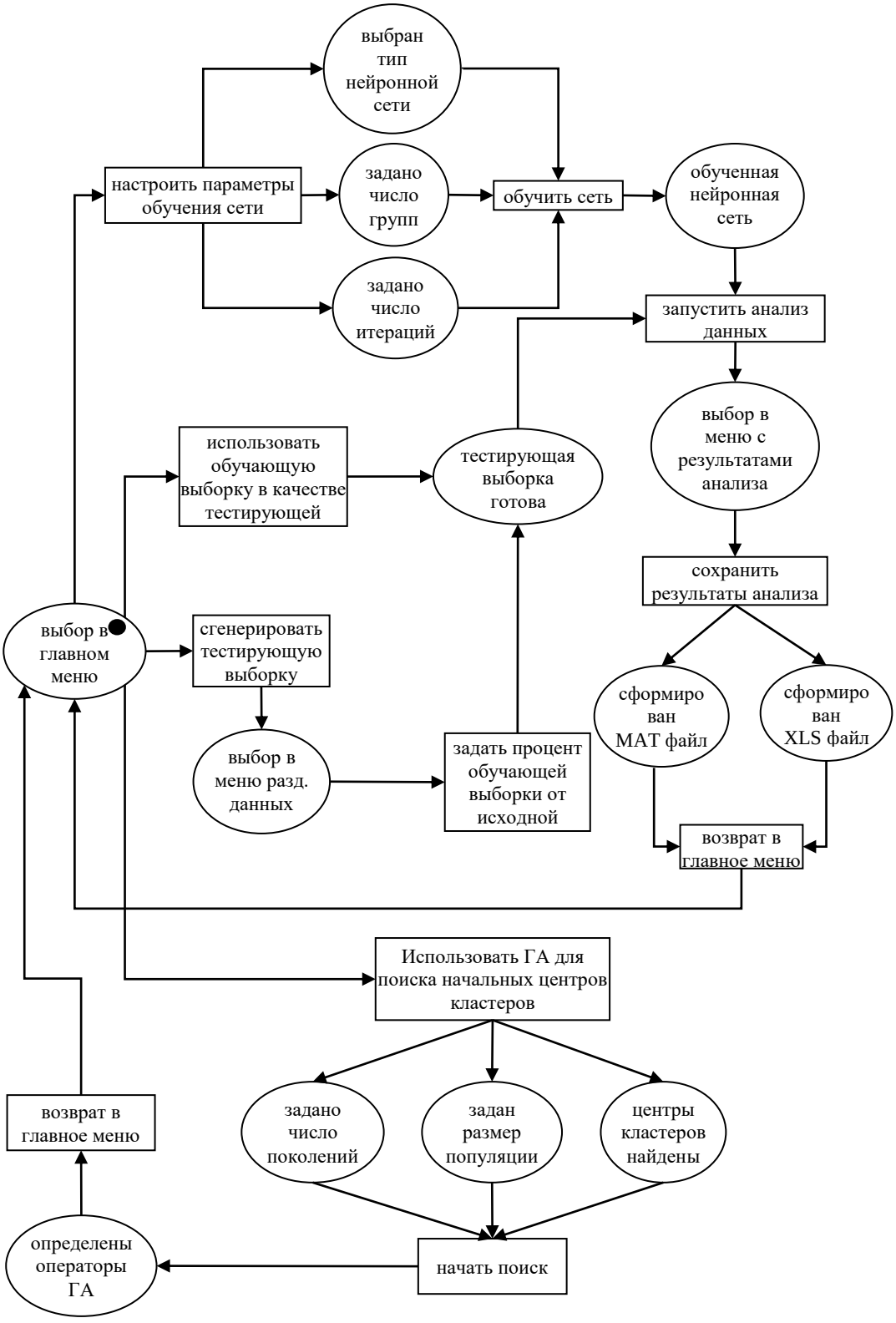

Рисунок 2.5б – Сеть Петри, описывающая проектируемую систему (продолжение)

В среде CPN Tools была построена сеть, идентичная приведенной на рисунках 2.5а и 2.5б, и проведен ее анализ.

Узлы дерева в среде CPN Tools отображаются в виде (рисунок 2.6):

Рисунок 2.6 – Отображение узла дерева в среде CPN Tools

где *N* – числовой номер узла дерева, *IN* – число входных дуг (ветвей), *OUT* – число выходных дуг.

Рисунок 2.7 отображает полученное в результате анализа дерево достижимости [74].

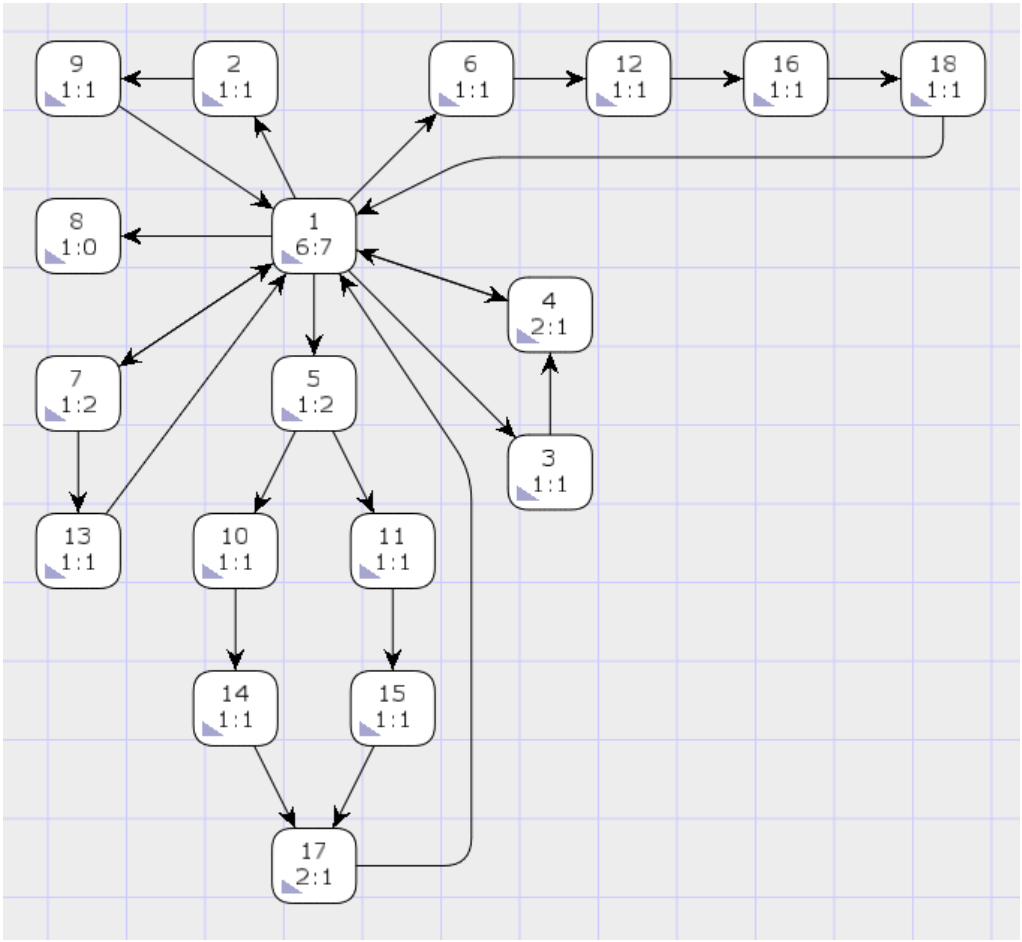

Рисунок 2.7 – Дерево достижимости для построенной сети Петри

Построение дерева достижимости начинается с первого (корневого) узла дерева (на рисунке 2.7 узел дерева имеет числовой номер «1»). Можно заметить, что число узлов дерева заметно меньше числа вершин графа, приведенного на рисунках 2.5а-2.5б. Причина этого – активация таких переходов, как: «настроить параметры автокодировщика» «настроить параметры обучения сети» «настроить параметры факторного анализа». В результате активации приведенных переходов происходил перенос меток из входной позиции в несколько выходных позиций. Однако на дереве достижимости подобное увеличение числа меток в позициях отображается одним узлом.

Как показал анализ, в построенной сети Петри отсутствуют мертвые переходы, что сигнализирует о корректно спроектированной архитектуре системы [69,73].

Выводы по главе 2

1. Обосновано применение системного анализа для проектирования и разработки СППР. Предложена структурная схема СППР, состоящая из 4 подсистем: препроцессинга; редукции данных; обучения и тестирования нейронных сетей; экспорта результатов анализа. Построена диаграмма DFD, позволяющая наглядно отобразить принцип работы разрабатываемой системы.

2. Автором предложена методика использования генетического алгоритма для предварительной настройки матрицы весов слоя Кохонена.

3. Применен аппарат сетей Петри для проектирования и анализа работы системы. Получено решение задачи достижимости в виде дерева, не содержащего мертвые переходы, что доказывает корректность спроектированной архитектуры системы.

Глава 3. Реализация СППР и ее апробация

3.1 Описание программной реализации СППР

Исходя из набора требований к разрабатываемой системе, было принято решение использовать MATLAB в качестве языка программирования и среды разработки.

Данное решение обосновано следующими возможностями MATLAB [75,76]:

- высокоуровневый язык программирования MATLAB, по сравнению с традиционными языками программирования (C/C++, Java, C#), позволяет на порядок сократить время решения типовых задач и значительно упрощает разработку новых алгоритмов;
- обеспечивается возможность максимально просто и эффективно работать с матрицами реальных, комплексных и аналитических типов данных и со структурами данных;
- исследователю предоставляется базовый набор встроенных функций построения двумерных и трехмерных графиков, а также функции объёмной визуализации.

Однако главным достоинством MATLAB является наличие большого числа библиотек, поставляемых официальными разработчиками среды [77-79].

В ходе реализации были использованы следующие библиотеки (Toolbox):

- Statistics and Machine Learning Toolbox использовалась, в основном, для реализации модуля редукции данных, а также в модуле предварительной обработки данных;
- Deep Learning Toolbox применялась для реализации следующих нейронных сетей: автокодировщик, сеть Кохонена, многослойный персептрон, слой софтмакс, сеть распознавания образов;
- Global Optimization Toolbox использовалась в ходе создания модуля, реализующего генетический алгоритм для нахождения начальных параметров слоя Кохонена [80].

На рисунке 3.1 приведена иерархия модулей разработанной СППР.

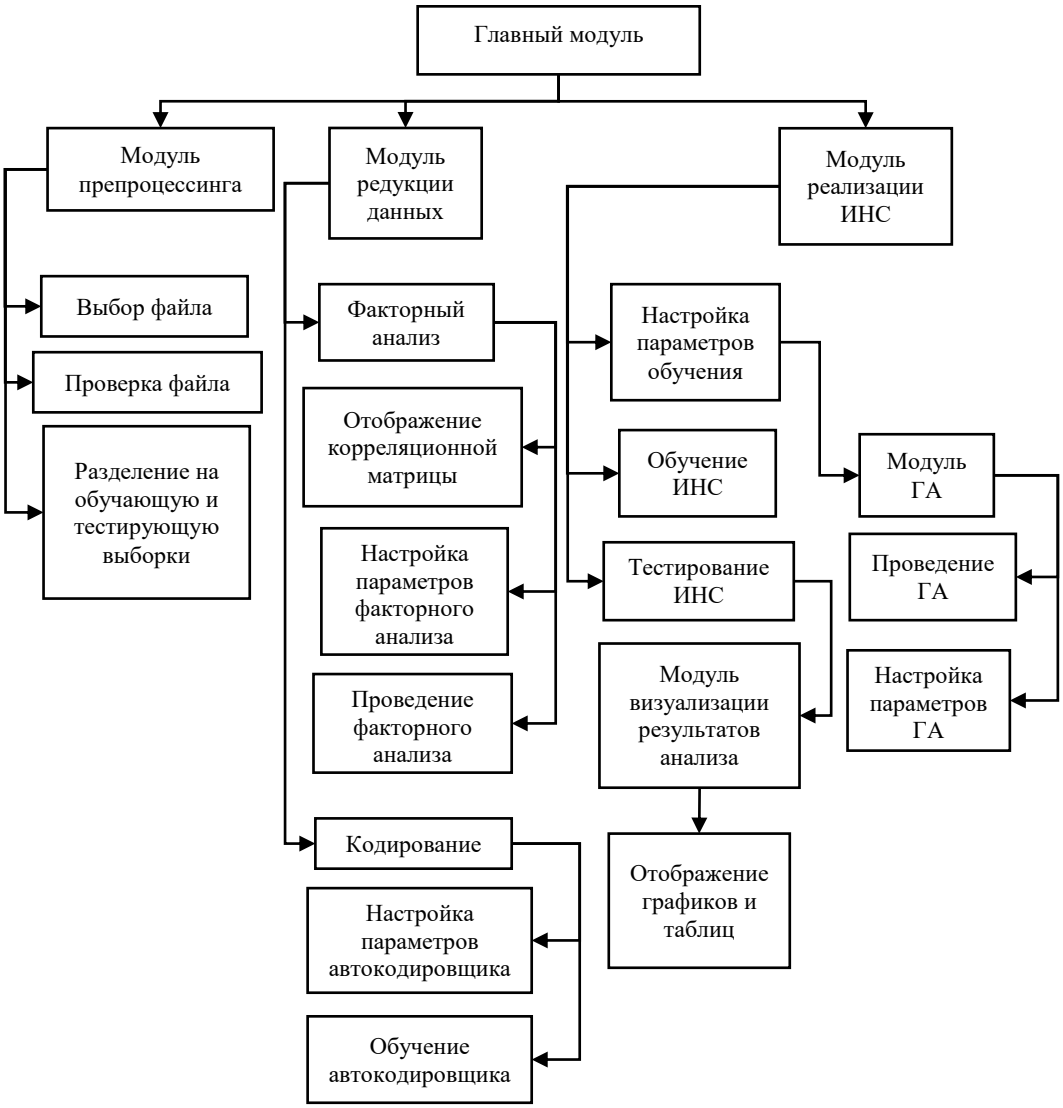

Рисунок 3.1 – Иерархия модулей СППР

Разработанная система представляет собой приложение с графическим интерфейсом пользователя [81].

Модуль предварительной обработки данных (препроцессинга) позволяет исследователю выбрать импортируемый файл, который проверяется на наличие в нем выборки, представляющей матрицу чисел.

Исследователь может разделить выборку на две: обучающую и тестирующую. Процентное соотношение обучающей и тестирующей выборки исследователь определяет сам.

В модуле редукции данных исследователь имеет возможность использовать два способа снижения размерности пространства признаков выборки: факторный анализ и автокодировщик.

При использовании факторного анализа (рисунок 3.2) исследователь имеет следующие возможности:

- Просмотр корреляционной матрицы.

В системе значения корреляционной матрицы имеют различные цвета при отображении.

Согласно шкале Чеддока существует пять степеней связи: слабая – от 0.1 до 0.3 (зеленый); умеренная – от 0.3 до 0.5 (фиолетовый); заметная – от 0.5 до 0.7 (синий); высокая – от 0.7 до 0.9 (оранжевый); очень высокая – от 0.9 до 1.0 (красный).

- Выбор числа факторов.

Определяется максимальное число факторов для анализируемых данных согласно неравенству:

$$(d + m) \leq (d - m)^2,$$

где d – число признаков, m – число факторов.

- Выбор типа вращения.
- Получение факторных значений и нагрузок.

Значения факторных нагрузок также имеют различные цвета, благодаря чему исследователь может легко определить число факторов. В то же время система автоматически определяет оптимальное число факторов, основываясь на критериях Кайзера и доли воспроизводимой дисперсии, и предлагает использовать эти значения.

- Экспорт полученных значений.

Система позволяет экспортировать редуцированные данные для последующего анализа.

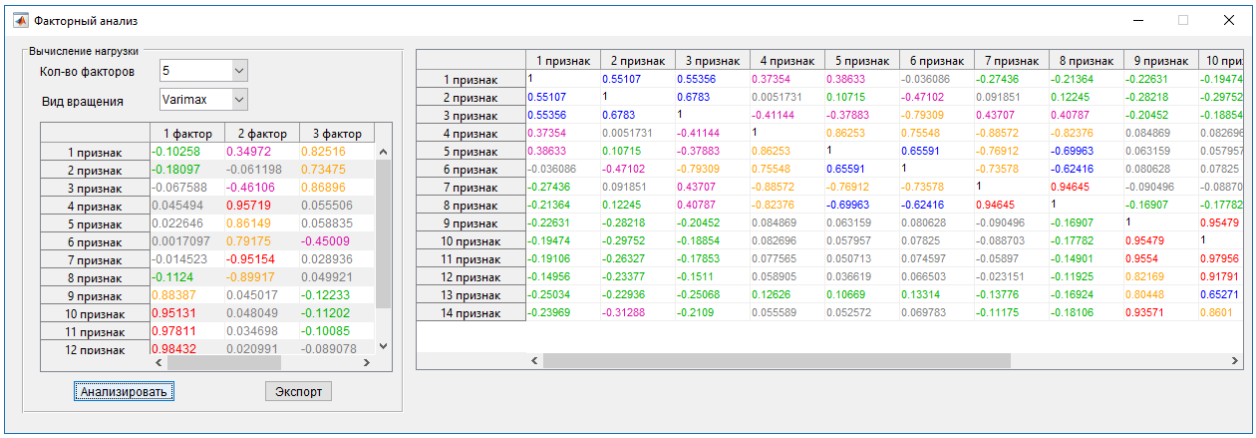

Рисунок 3.2 – Интерфейс модуля факторного анализа

В процессе использования автокодировщика (рисунок 3.3) исследователю предоставляются следующие возможности:

- Задание числа нейронов в скрытом слое.
- Задание максимального числа итераций обучения автокодировщика.
- Выбор функций активации кодера и декодера.

По умолчанию используются сигмоидальные функции активации, которые можно сменить на линейные функции активации с насыщением.

- Выбор алгоритма обучения сети.

Приоритетным алгоритмом обучения является алгоритм Моллера, однако можно использовать другой алгоритм, основанный на методе сопряженных градиентов.

- Проведение обучения и тестирования сети.

Обучение и тестирование автокодировщика (рисунок 3.4) производится на одних и тех же данных. После обучения на вход сети подается вся матрица: кодер снижает размерность матрицы и на выходе этого слоя система получает кодированные значения.

- Экспорт матрицы кодированных значений.

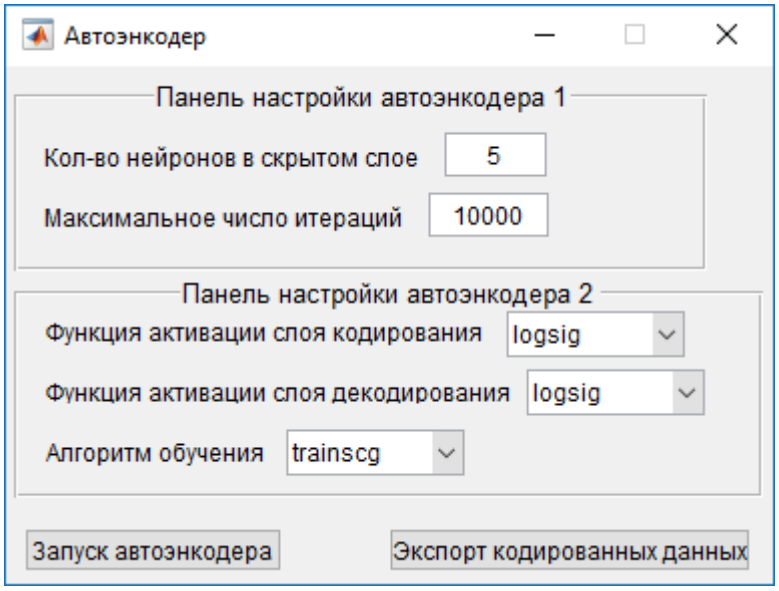

Рисунок 3.3 – Интерфейс модуля кодирования

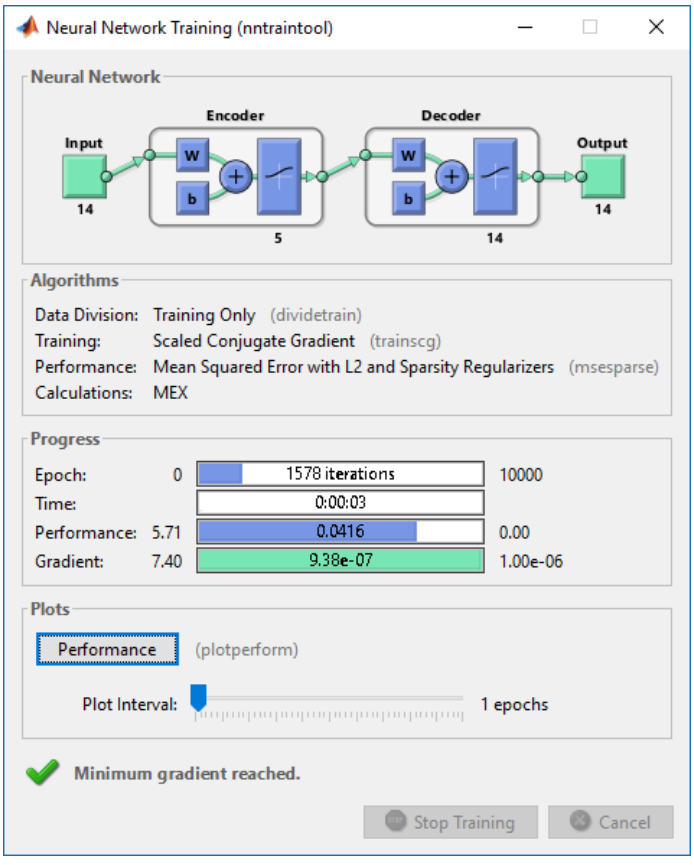

Рисунок 3.4 – Интерфейс окна обучения автокодировщика

Модуль реализации ИНС обеспечивает исследователю следующие возможности:

• Выбор кластеризации или классификации в качестве решаемой задачи анализа данных, а также типа сети для ее решения.

• Задание числа классов (при решении задачи классификации) или кластеров (при решении задачи кластеризации).

• Задание максимального числа итераций обучения сети.

• Определение дополнительных параметров обучения для используемой сети.

В задаче кластерного анализа, решаемой путем использования слоя Кохонена, исследователь может использовать генетический алгоритм для нахождения начальной матрицы весов сети (рисунок 3.5).

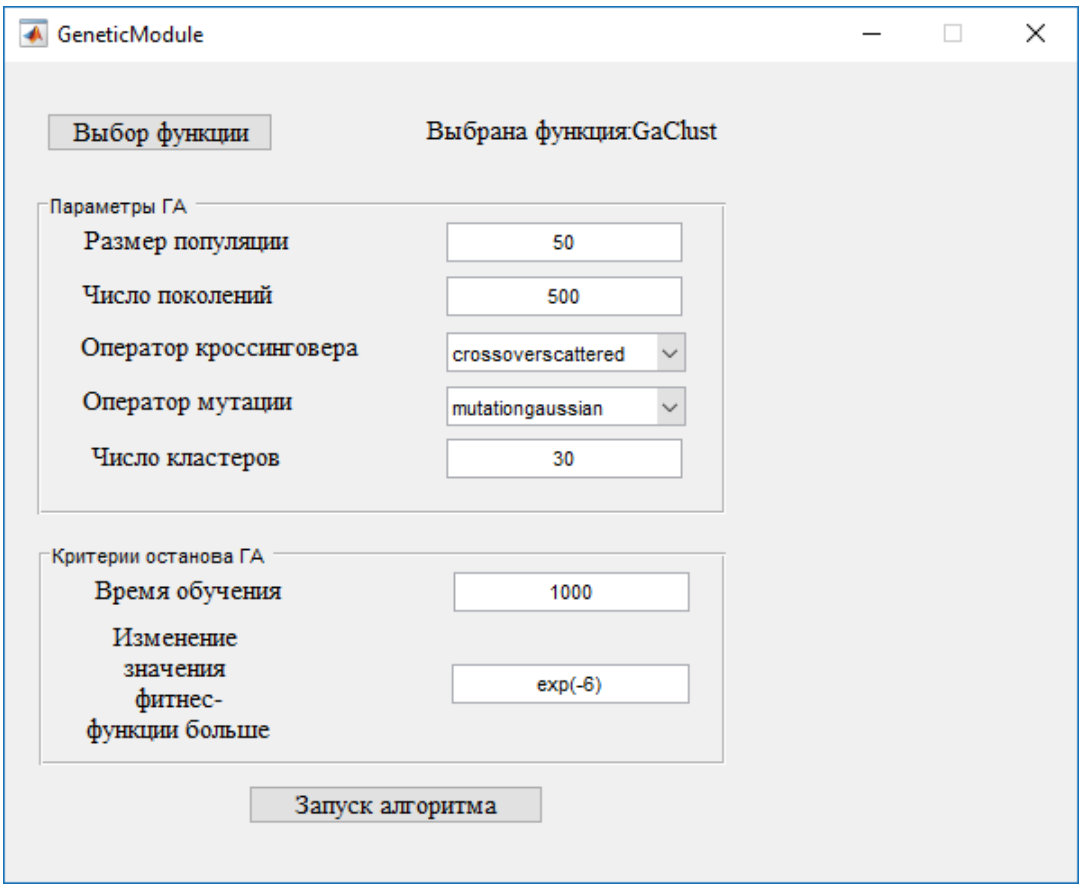

Рисунок 3.5 – Интерфейс модуля генетического алгоритма

В общем случае, система позволяет настроить для каждой сети такие часто используемые параметры как: алгоритм обучения, скорость обучения (learning

rate), минимальное значение градиента и ряд других (зависит от выбора нейронной сети).

• Выбор обучающей и тестирующей выборок.

Исследователь может использовать для обучения всю выборку или только обучающую. Аналогично, для тестирования можно использовать всю выборку или только тестирующую.

• Обучение и тестирование сети.

На рисунке 3.6 в качестве примера представлен процесс обучения многослойного персептрона, решающего задачу классификации.

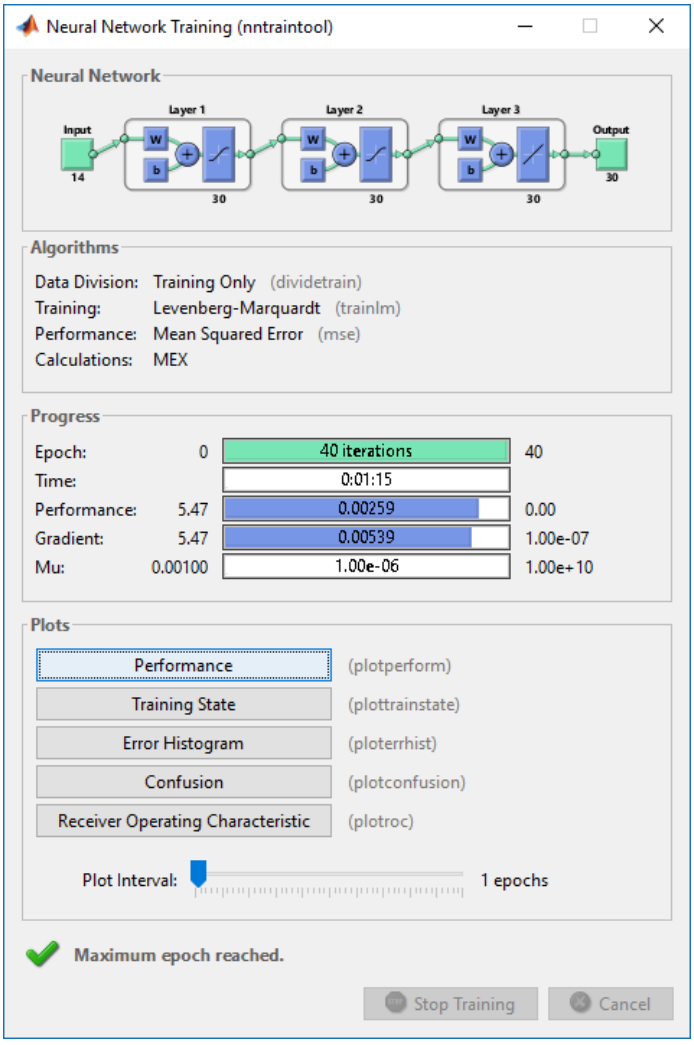

Рисунок 3.6 – Интерфейс окна обучения сети

Из рисунка видно, что сеть состоит из трех скрытых слоев, имеющих следующие названия: Layer 1, Layer 2, Layer 3. Входной слой имеет 14 нейронов, выходной – 30. В процессе обучения сети использует широко известный алгоритм Левенберга-Марквардта, работа которого останавливается по достижению заданного исследователем числа итераций обучения. Фиксируется время, затраченное на обучение сети.

После этого производится тестирование обученной сети - на ее вход подается тестирующая выборка. Вычисляются:

- общий процент ошибочно классифицированных наблюдений по всей тестирующей выборке;
- процент ошибочно классифицированных наблюдений по каждому классу.

Ранее представленный на рисунке 3.5 модуль генетического алгоритма отвечает за нахождение начальных центров кластеров или же начальной матрицы весов сети.

В данном модуле предусмотрено:

- выбор функции.
- Задание числа популяции и размера поколений.
- Выбор операторов ГА.

В системе реализован выбор основных операторов ГА:

- Оператор кроссинговера (скрещивания) может быть реализован следующими видами: одноточечное, двухточечное, эвристическое, арифметическое и рассеянное скрещивания.
- Оператор мутации реализуется путем выбора одного из двух видов мутации: гауссовская мутация и равномерная мутация.
- Задание числа кластеров.
- Определение критериев останова генетического алгоритма.

Используется два критерия останова:

- формирование новой популяции не приводит к значимому улучшению значения функции приспособленности;

• достигнут лимит на время выполнения ГА, либо лимит на число поколений.

На рисунке 3.7 в качестве примера представлен результат работы генетического алгоритма. Из рисунка видно, что алгоритм остановился по достижении максимального числа поколений. Отображены значения функции приспособленности для каждого поколения.

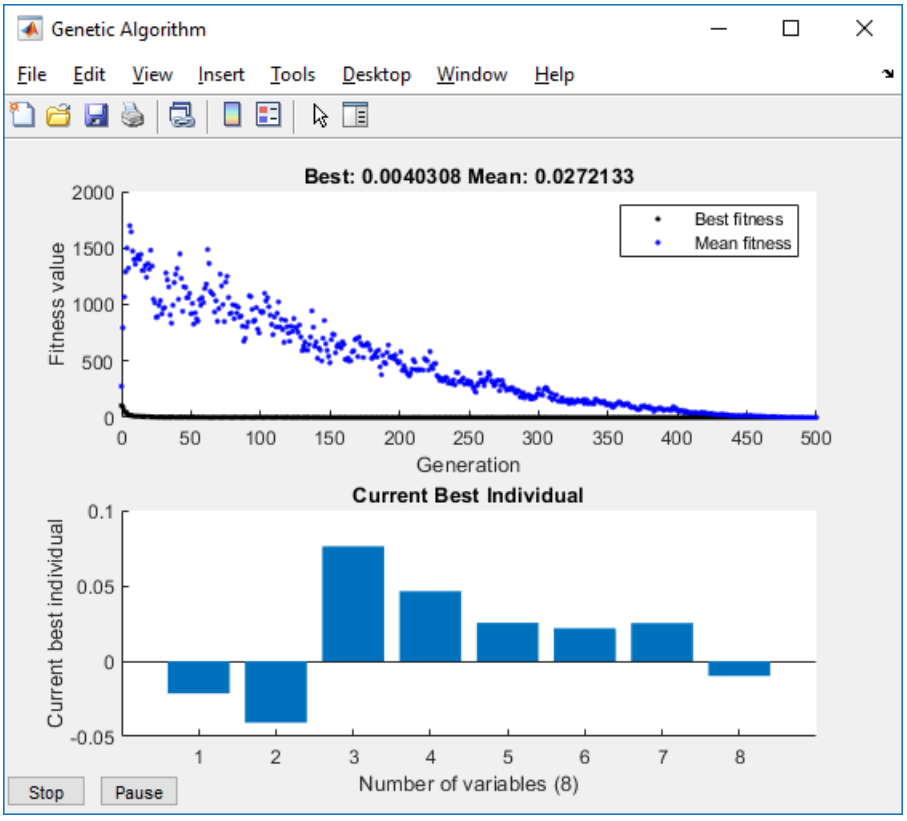

Рисунок 3.7 – Результат работы генетического алгоритма

Представленный на рисунке 3.8 модуль визуализации обладает простым интерфейсом, благодаря которому исследователю будет представлена подробная информация о результатах проведенного анализа. Данную информацию исследователь может сохранить в файлах с расширениями .mat и .xls.

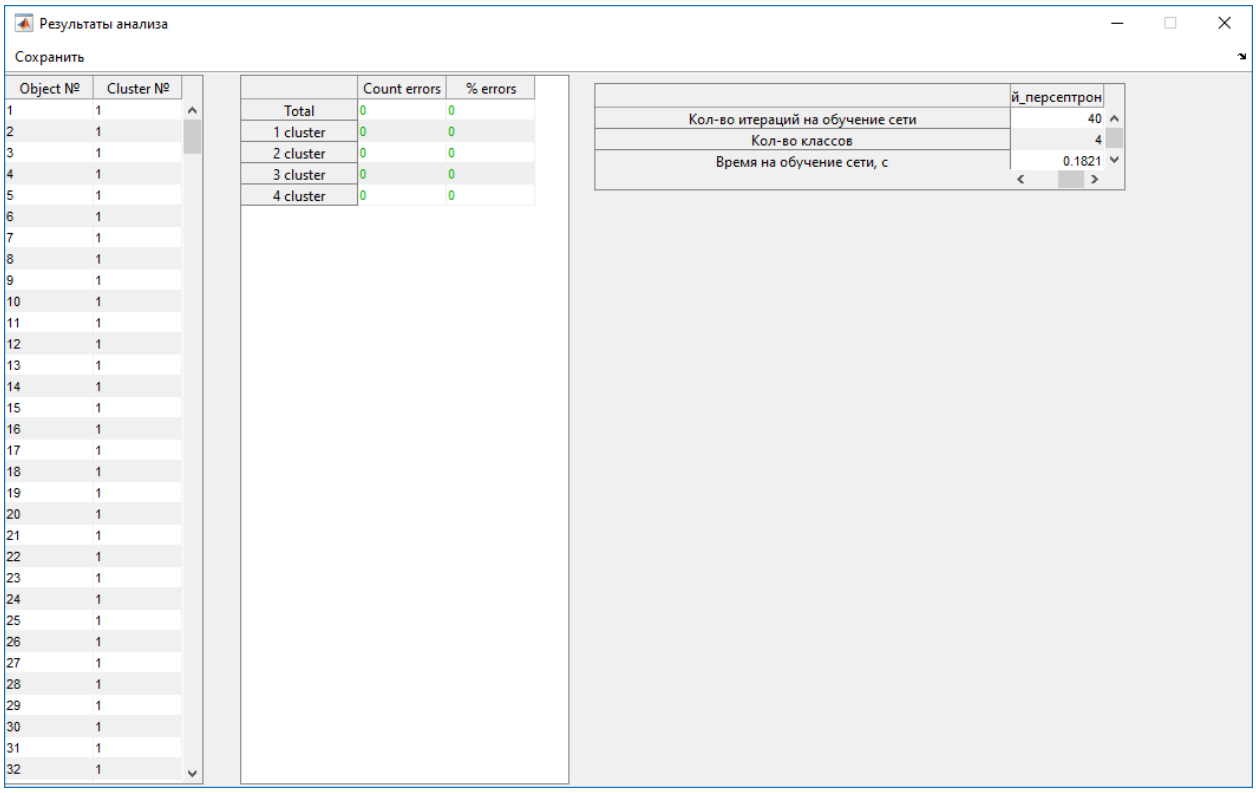

Рисунок 3.8 – Интерфейс модуля визуализации данных

Слева отображается таблица, содержащая информацию по каждому наблюдению: красным цветом отмечаются неверно классифицированные наблюдения.

Центральная таблица служит для отображения процентов неверно классифицированных наблюдений в каждом классе и общего процента ошибок при анализе данных.

Последняя таблица содержит следующую информацию:

- число итераций обучения;
- число классов/кластеров;
- время, затраченное на обучение сети.

Для более комплексного отображения результатов анализа используются графики, которые система может построить, если выполнено одно из условий:

1) анализировалась выборка, содержащая не более 3 признаков;

2) исследователь производил редукцию данных (снизил размерность исходной выборки до двухмерной, либо трехмерной размерностей).

В качестве примера на рисунке 3.9 визуализированы результаты классификации случайных объектов, имеющих 3 признака и разделенных на 4 класса, которые ранее были приведены на рисунке 3.8.

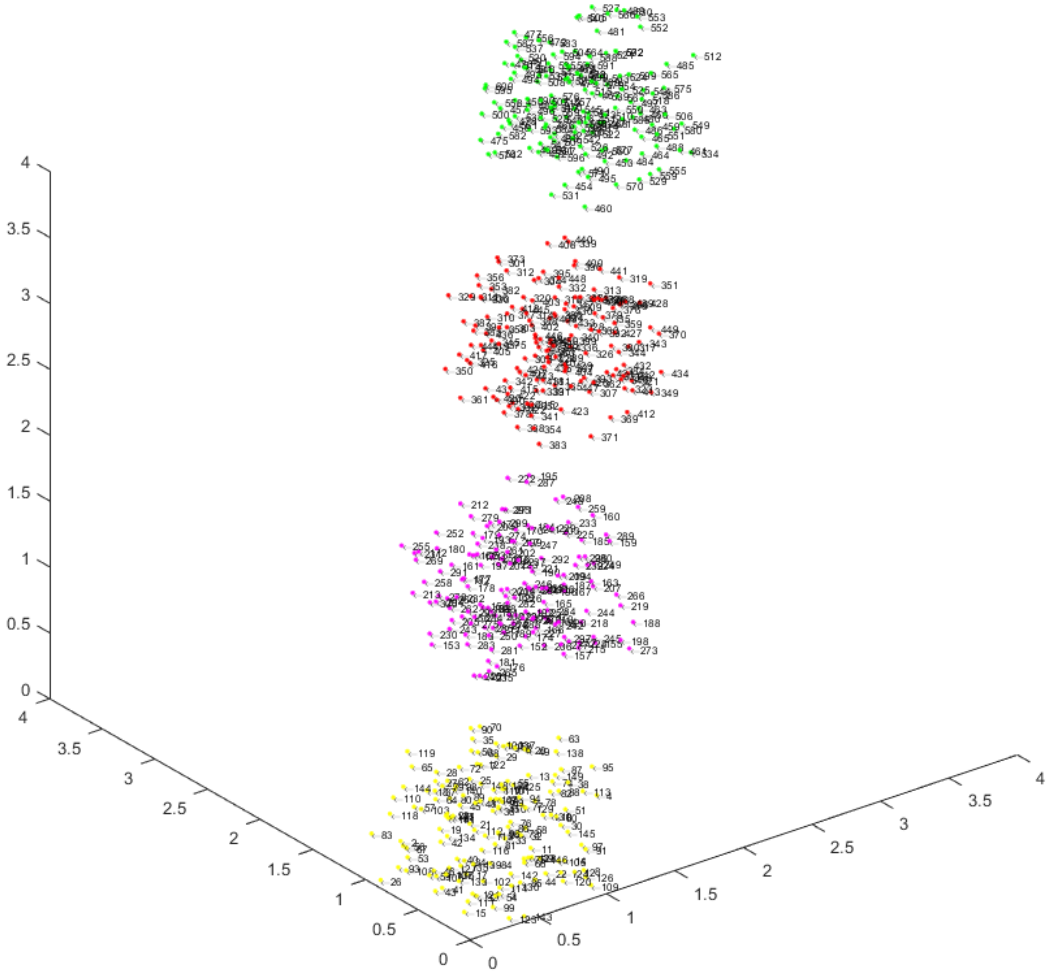

Рисунок 3.9 – Отображение результатов классификации в виде 3D графика

3.2 Апробация реализованной системы на примере решения задач анализа дорожно-транспортных происшествий

На текущий момент в мире существует широкое множество картографических сервисов, производящих оценку дорожной обстановки, в

частности их влияния на пропускную способность. Для сбора информации во многих развитых странах сервисы используют стационарные датчики интенсивности дорожного движения. В большинстве случаев эти датчики позволяют получать информацию о [82]:

- интенсивности;
- занятости полос;
- средней скорости по полосам;
- временном интервале движения;
- дистанции;
- времени детектирования объекта;
- типе транспортного средства;
- скорости автомобиля;
- плотности потока.

Использование стационарных датчиков интенсивности дорожного движения практически полностью гарантирует получение «чистых» данных о пропускной способности дорог, но покрытие ими дорожной системы даже в границах одного крупного города потребует значительных финансовых вложений.

Во многом по этой причине, один из популярнейших картографических сервисов Яндекс.Карты использует данные, полученные от пользователей приложений: Яндекс.Карты и Яндекс.Навигатор.

Мобильные устройства пользователей передают сервису Яндекс.Пробки (именно эта система отвечает за оценку и прогнозирование загруженности дорог) данные о движении автомобилей. Каждые несколько секунд устройство водителя передаёт свои географические координаты, направление и скорость движения в компьютерную систему Яндекс.Пробки. В целях конфиденциальности все данные обезличены, то есть не содержат никакой информации о пользователе или его автомобиле [83].

Кроме этого автомобилисты могут сообщать сервису дополнительную информацию об авариях, ремонтных работах или других дорожных

неприятностях. Например, какой-нибудь сознательный водитель, увидев ДТП, предупредил о нём других автолюбителей, поставив соответствующую точку в мобильных Яндекс.Картах.

При наличии ДТП и по мере приближения автомобилистов к месту ДТП, их скорость будет уменьшаться, и мобильные устройства начнут отправлять сервису запросы, содержащую информацию о вероятном заторе на дороге. Таким образом, Яндекс производит актуальную оценку дорожной обстановки практически на всей территории РФ.

Прогнозирование дорожной обстановки является значительно более сложной задачей и обычно ее решение требует огромных вычислительных ресурсов. В компании Яндекс одним из основных подходов к ее решению является построение графа, который представляет собой карту дорог. Каждый перекрёсток – вершина этого графа, а участки дороги между двумя перекрёстками – рёбра. У последних есть атрибуты «длина» и «скорость». Длина известна заранее, а скорость вычисляется в реальном времени. Решение задачи прогнозирования заключается в вычислении наиболее вероятных значений второго атрибута всех ребер графа.

Как уже описывалось ранее, Яндекс собирает данные о местоположении пользователей, по этим данным затем строится единый маршрут движения с информацией о скорости его прохождения – GPS-трек. Каждый такой трек – это цепочка сигналов о местоположении (широте и долготе) автомобиля в конкретное время. GPS-треки поступают не только от частных водителей, но и от машин компаний-партнеров (организации с большим парком автомобилей, курсирующих по городу). GPS-треки имеют привязку к дорожному графу, то есть информация о том, по каким улицам проходил каждый маршрут. Затем усредняется скорость всех GPS-треков, проходящих по одним и тем же рёбрам [84].

В Яндексе создана некоторая модель (информации об этой модели нет в свободном доступе), которая обучается раз в сутки. При обучении модели используется огромный объём данных. Граф дорог занимает около 100 гигабайт: это информация о топологии – какое ребро с каким связано, а также о геометрии –

GPS-координаты начала, конца и точек изгиба каждого сегмента дороги. История треков, которую хранит Яндекс, занимает десятки терабайт.

При обучении модель, на основе развития событий за какой-то временной промежуток в прошлом, строит прогноз. Этот прогноз сравнивается с последующим развитием событий. В модели подбираются коэффициенты, минимизирующие расхождение между прогнозом и случившейся дорожной обстановкой. Для каждого времени суток система рассчитывает свои коэффициенты, поскольку характер движения в течение дня меняется.

В итоге каждые несколько минут по готовой модели и свежим трекам рассчитывается прогноз на ближайший час.

Данный подход не является принципиально новым. Он базируется на ряде исследований, проводимых в различных университетах и исследовательских центрах мира. Среди этих работ стоит отметить следующие:

- работу исследовательского коллектива Кембриджского университета, содержащую экспериментальное сравнение линейно-регрессионного метода и метода k-ближайших соседей [85];
- работу исследовательского отдела IBM, которая описывает успешный опыт применения авторегрессии в прогнозировании дорожного трафика [86];
- работу сингапурских и китайских ученых по успешному применению нейронных сетей для прогнозирования дорожного трафика [87].

Последняя работа описывает построение и использование сетей прямого распространения сигнала, а именно сетей радиальных базисных функций, основная задача которых – прогнозирование временных рядов. Однако данные сети также способны решать задачу классификации.

Поскольку ДТП является одной из основных причин непрогнозируемого изменения пропускной способности и, как следствие, изменения дорожной обстановки, то точность сделанного ранее прогноза снижается в связи с отсутствием оперативного определения класса ДТП. Получение данной оценки, в

частности, можно трактовать как решение задачи классификации и кластеризации с использованием подходов описанных в параграфах 1.1–1.5.

Описанное выше позволяет предложить с следующую концептуальную модель СППР (рисунок 3.10).

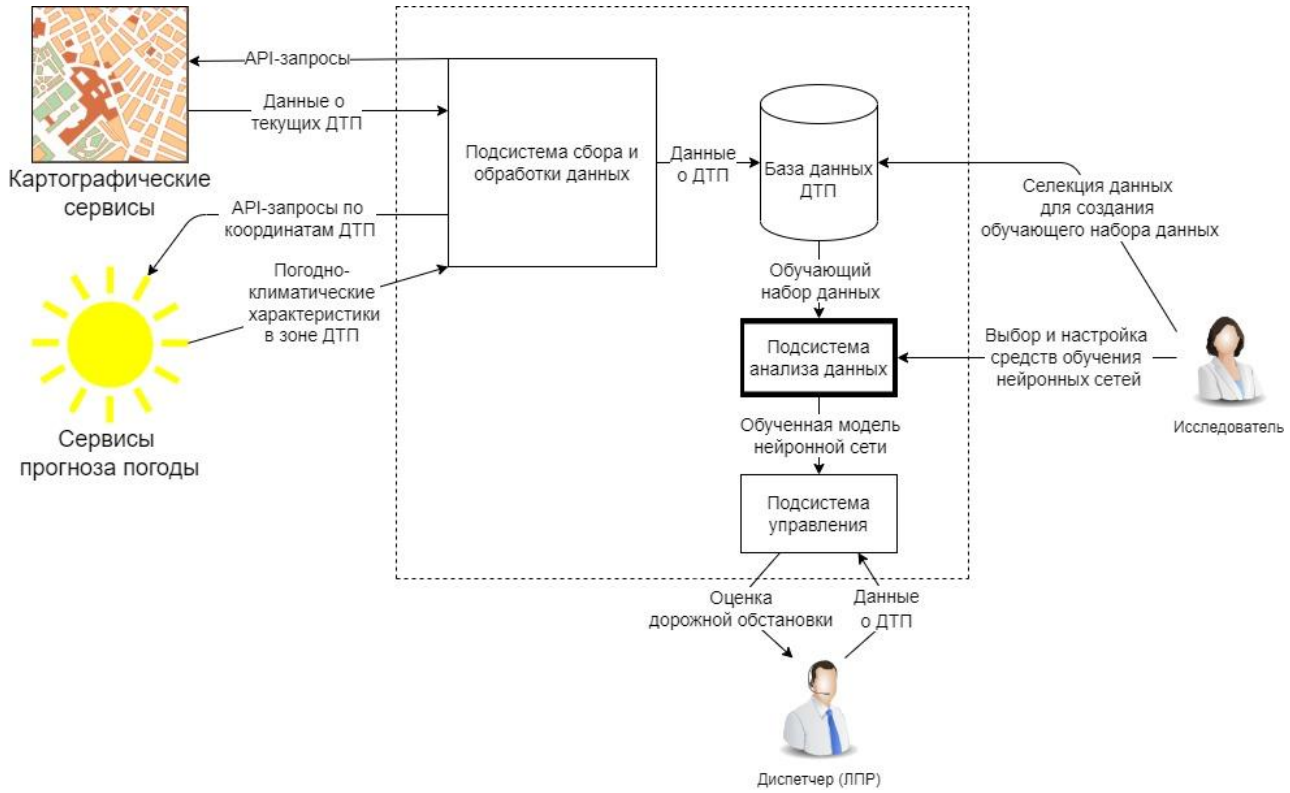

Рисунок 3.10 – Концептуальная модель СППР для классификации ДТП по их влиянию на пропускную способность дорожного участка

Концептуально целью всей системы является обеспечение поддержки принятия диспетчером решений при оценке серьезности ДТП на основе наиболее вероятного влияния на пропускную способность дорожного участка.

Подсистема «Сбор и обработка данных» имеет следующие подцели: обеспечить своевременный периодический сбор данных, их препроцессинг и запись. Целью подсистемы «Анализа данных» является обеспечение исследователя широким спектром алгоритмов и методов анализа данных. Цель последней подсистемы заключается в предоставлении диспетчеру удобного интерфейса для взаимодействия с созданным классификатором.

3.2.1 Решение задачи классификации ДТП, описываемых географическими, временными и погодными признаками

На сбор, обработку и хранение данных, с целью создания необходимого для анализа набора данных, нередко требуется много времени, а также финансовых вложений. Особенно если это касается создания классификатора на основе нейросетевого подхода.

По этой причине на популярнейшей в мире публичной веб-платформе Kaggle был найден единственный подходящий набор данных [88]. Данные в нем были собраны сторонним исследователем с использованием API-методов различных онлайн сервисов, поэтому их можно считать достоверными. Объекты в этом наборе данных представляют собой информацию о ДТП. Подходящий набор данных, содержащий информацию о ДТП, произошедших на территории России, не был обнаружен в открытых источниках.

Основная часть исследований по апробации реализованной системы и разработанной методики производятся с использованием этого набора данных.

Сбор данных производился на территории США на протяжении 5 лет: с 2016 по 2020 гг. Таким образом, была собрана информация о свыше 3.5 миллионов ДТП. Происшествия описываются 48 признаками, которые подразделяются на 4 группы (территориальные, погодно-климатические, временные и группа средств регулирования дорожного движения).

Картографический веб-сервис MapQuest определяет уровень серьезности каждого зафиксированного ДТП. Этот уровень зависит от того, насколько сильно данное происшествие повлияло на пропускную способность дорожного участка. Чем сильнее это влияние, тем выше уровень серьезности ДТП. В сервисе MapQuest существует 4 уровня, где 1 уровень присваивается тому ДТП, которое практически или совсем не оказало негативное влияние на пропускную способность дорожного участка.

Полученный набор данных хранится в csv формате и имеет размер свыше 1 гигабайта. Однако не обо всех объектах собрана информация: могут отсутствовать значения признаков в наборе данных. Поэтому необходимо выполнить препроцессинг (предварительную обработку) данных.

Основные задачи, решаемые в ходе препроцессинга: очистка и оптимизация данных.

Очистка данных служит для заполнения отсутствующих значений, обнаружения и удаления шумов и выбросов в данных. Кроме этого, в процессе очистки преобразуются некорректные форматы.

Оптимизация данных снижает размерность пространства признаков путем выявления и исключения признаков, оказывающих незначительное влияние на определение класса объекта. Позволяет адаптировать данные к конкретной задаче, что повысит эффективность работы с ними.

При работе с набором данных такого объема на уровне среды разработки (IDE), заметно используется оперативная память и другие ресурсы вычислительной системы. По этой причине целесообразно использовать базу данных для хранения этого набора. Это также позволит использовать встроенные алгоритмы оптимизатора запросов для проведения очистки данных.

Для этих целей была автором создана база данных на сервере Microsoft SQL Server 2019 Express. Изначально эта база включала одну таблицу, которая хранит все данные из csv файла. Язык T-SQL позволяет эффективно работать с данными на сервере баз данных, расходуя значительно меньше вычислительных ресурсов системы.

Графически этап препроцессинга в этой работе можно описать в виде диаграммы поток данных (рисунок 3.11).

В первую очередь автором решалась задача оптимизации данных. Выбор несущественных признаков (оказывающих незначительное влияние на определение класса ДТП) был сделан согласно документу [89], в котором пропускная способность определяется как максимальное число автомобилей, которое может пропустить участок дороги в единицу времени в одном или двух

направлениях в рассматриваемых дорожных и погодно-климатических условиях. Поскольку набор данных не содержит признаков, описывающих состояния дорожного покрытия, обочины, а также видов разметки, были выбраны признаки описывающие погодно-климатические условия, при которых было зафиксировано ДТП. В дополнение к ним были добавлены признаки, описывающие: географическое положение ДТП; период дня, в котором оно произошло.

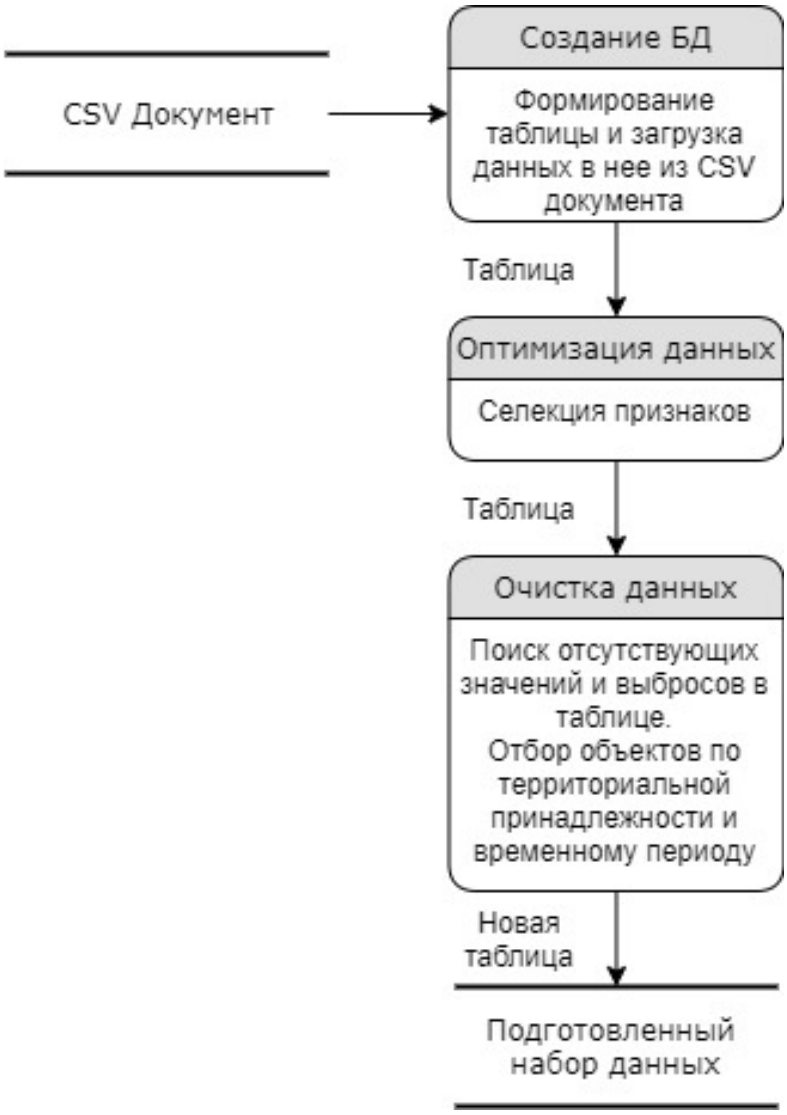

Рисунок 3.11 – Диаграмма потоков данных, описывающая этап препроцессинга

Это позволило сократить число признаков с 48 до 6:

- широта (градусы);
- долгота (градусы);

- температура (Цельсий);
- дальность видимости (километры);
- период дня (светлое /темное время суток);
- гражданские сумерки (да/нет).

Различают гражданские, навигационные и астрономические сумерки. Утренние гражданские сумерки начинаются перед восходом Солнца, когда высота Солнца равна 6 градусам, и заканчиваются в момент восхода Солнца. Вечерние гражданские сумерки начинаются в момент захода Солнца и продолжаются до тех пор, пока высота Солнца не станет равной 6 градусам. Высота Солнца над горизонтом – это угол между Солнцем и горизонтом. Данная величина изменяется от 0 до 90 градусов. Ноль градусов соответствует восходу или закату, т.е. положению Солнца над горизонтом, а 90° – Солнцу в зените [90].

Для проведения очистки данных был написан ряд SQL-запросов, который позволил:

- найти субъект (штат) с достаточным количеством объектов каждого класса (не менее 15% от общего числа ДТП в этом штате);
- найти и исключить строки с отсутствующими значениями, либо со значениями, выходящими за рамки допустимых (выбросы);

ДТП с первым и четвертым уровнем серьезности были исключены из набора данных по причине того, что такие ДТП происходят крайне редко.

В результате был получен список из более чем 20 штатов со всеми зафиксированными в них ДТП. Но погодно-климатические условия на территории одного штата могут сильно отличатся от условий в другом, поэтому был выбран один штат – Миннесота. Целесообразность выбора данного штата основывается на том, что погодно-климатические условия в нем схожи с условиями в большинстве регионов России.

В дополнение ко всем перечисленным этапам целесообразно учитывать ДТП, произошедших в более актуальный временной период: за 2019 г.

Результатом препроцессинга стал подготовленный для анализа набор данных, хранящийся в новой таблице, которая содержит 12094 строки (объекта), каждый из которых описывается 6 ранее приведенными признаками.

Значения признаков были нормализованы с помощью предложенного автором способа использования автокодировщика как альтернативного метода нормализации.

Для оценки снижения пропускной способности дорожного участка под влиянием ДТП были реализованы классификаторы на основе сетей прямого распространения: слоя софтмакс, сети распознавания образов, многослойного персептрона и каскадной сети прямого распространения сигнала. Значения признаков были нормализованы с целью приведения значений к единому диапазону.

В модуле реализации ИНС производилась настройка основных параметров обучаемых сетей:

- задание числа итераций обучения для каждой используемой сети;
- выбор алгоритма обучения сети;

Сети обучались по наиболее рекомендуемым в литературе алгоритмам (слой софтмакс и сеть распознавания образов по алгоритму Моллера, многослойный персептрон и каскадный многослойный персептрон по алгоритму Левенберга-Марквардта). В обучающую выборку вошли 80% случайно выбранных объектов, в валидационную и тестирующую – по 10% от общего числа.

Первые созданные в рамках данного исследования классификаторы представляли собой сеть с одним скрытым слоем. Краткая информация о результатах классификации, которые были получены при помощи выбранных сетей приведены в таблице 3.1. В последующих таблицах полужирным шрифтом выделены наилучшие результаты проведенных экспериментов.

Простейший способ оценки качества решения задачи классификации – вычисление общей доли правильно классифицированных объектов (общая доля).

Она вычисляется как отношение числа правильно классифицированных объектов к числу всех объектов в выборке.

Точность – это доля объектов (наблюдений) истинно принадлежащих данному классу по отношению ко всем объектам, которые классификатор отнес к этому классу.

Полнота – это доля объектов истинно принадлежащих данному классу по отношению ко всем объектам истинно принадлежащих данному классу.

Таблица 3.1 – Результаты классификации

Сеть	Число нейронов в скрытых слоях	Общая доля, %	Точность, %	Полнота, %
Слой софтмакс	-	59.99	60.88, 14.29	97.33, 0.71
Сеть распознавания образов	1000	60.29	63.92, 47.63	81.01, 27.42
	3000	60.41	62.83, 46.79	87.12, 17.99
	5000	47.63	59.62, 37.24	45.21, 51.42
Многослойный персептрон	40	67.53	71.11, 59.77	79.29 48.87
	160	65.44	66.92, 59.66	86.32, 32.13
Каскадный многослойный персептрон	**40**	**69.85**	**74.19, 61.98**	**78.01, 56.91**
	160	67.72	70.72, 60.58	80.88, 46.76

Информация, приведенная в таблице 3.1, сигнализирует о том, что сети с таким числом скрытых слоев не могут обеспечить достаточное качество классификации, в особенности это заметно по результатам полученным слоем софтмакс. Увеличение числа нейронов в скрытом слое также не повышает значительно качество классификации.

Следующий этап реализации классификатора состоял в обучении многослойных нейронных сетей, содержащих в себе два и более скрытых слоя. Краткая информация об их архитектуре, параметрах обучения и результатах

классификации, которые были получены при помощи них, приведены в таблице 3.2.

Таблица 3.2 – Результаты классификации (многослойные сети)

Сеть	Число нейронов в скрытых слоях	Общая доля, %	Точность, %	Полнота, %
Сеть распознавания образов	1000, 1000	60.78	63.31, 48.38	86.17, 20.56
	500, 1000, 2000	60.81	65.24, 48.99	77.72, 34.12
Многослойный персептрон	40, 20	68.77	72.31, 61.37	79.56, 51.51
	40, 20, 10	72.46	76.49, 65.36	79.6, 61.08
	40, 20, 10, 5	72.33	76.57, 64.81	78.92, 61.77
Каскадный многослойный персептрон	40, 20	69.81	73.26, 62.82	80, 53.64
	40, 20, 10	72.63	77.27, 64.96	78.48, 63.35
	40, 20, 10, 5	**72.95**	**77.21, 65.69**	**79.63, 62.84**

Соответствующие значения из таблиц 3.1 и 3.2 имеют отличия в качестве классификации. Таким образом, была найдена модель нейронной сети, позволяющая получать достаточно хорошее, для решения этой задачи, качество классификации ДТП, производить качественную оценку их влияния на пропускную способность дорожных участков по их географическим и погодно-климатическим признакам.

3.2.2 Решение задачи классификации ДТП при использовании в качестве признаков средства регулирования, географические и погодно-климатические характеристики

В большинстве случаев ДТП происходят в населенных пунктах, либо в прилегающих к ним территориях, а также на междугородних трассах. Как правило, дорожные участки в этих областях оборудованы различными средствами

регулирования дорожного движения: светофорами, дорожными знаками и разметкой. Большая часть этих средств регулирования направлена на предотвращение ДТП путем контроля движения транспортных потоков, а также для информирования водителей транспортных средств.

На территории РФ все дорожные знаки относятся к одной из восьми групп:

- запрещающие;
- предупреждающие;
- знаки приоритета;
- предписывающие;
- знаки особого предписания;
- информационные;
- сервисные;
- знаки с дополнительной информацией.

На наиболее опасных дорожных участках устанавливаются знаки, относящиеся к первым четырех группам. Большая часть этих знаков являются международными и применяются в большинстве стран мира.

В исследовании, описанном в параграфе 3.2.1, использовались GPS-координаты каждого ДТП из набора данных, и именно благодаря им автором набора данных была собрана информация о наличии средств регулирования дорожного движения вблизи мест ДТП. Всего набор данных содержит информацию о наличии следующих средств регулирования дорожного движения:

- дорожный знак «Неровная дорога»;
- дорожный знак «Пешеходный переход»;
- дорожный знак «Уступи дорогу»;
- дорожный знак «Перекресток» (наиболее эквивалентные знаки в РФ – «Участок перекрестка», «Пересечение равнозначных дорог», «Пересечение со второстепенной дорогой» и знаки с различным примыканием второстепенной дороги);
- дорожный знак «Въезд запрещен»;

- дорожный знак «Пересечение железнодорожных путей» (наиболее эквивалентные знаки в РФ – «Однопутная железная дорога», «Многопутная железная дорога»);
- дорожный знак «Круговое движение»;
- дорожный знак «Автозаправочная станция»;
- дорожный знак «Движение без остановки запрещено»;
- Светофорное регулирование;
- Дорожный знак «Разворот запрещен».

Однако наличие или отсутствие какого-либо из приведенных средств регулирования не всегда однозначно определяет серьезность произошедшей аварии.

В своей работе исследователь Artur Filipowicz [91] провел статистический анализ того же набора данных и определил, что наибольшее влияние на серьезность ДТП оказывает наличие знаков: «Перекресток», «Уступи дорогу», «Въезд запрещен». В большинстве случаев эти знаки применяются совместно, и устанавливаются на перекрестках.

На опасных перекрестках (с ограниченной видимостью, плотным движением и другими факторами, усложняющими его пересечение) устанавливается знак «Уступи дорогу». В населенных пунктах и их пригородах в таких местах часто устанавливается светофорное регулирование. Также очень опасным участком является пересечение железнодорожных путей, поэтому данный фактор стоит учитывать.

Таким образом, производится изменение набора признаков для описания ДТП: повторная оптимизация исходного набора данных.

В результате к используемым ранее 6 признакам были добавлены еще 5, имеющие наибольшее влияние на изменение пропускной способности дорожного участка из представленных в исходной выборке:

1. «Уступи дорогу»;
2. «Перекресток»;

3. «Пересечение железнодорожных путей»;

4. «Движение без остановки запрещено»;

5. «Светофорное регулирование».

Каждый из этих признаков является бинарным (есть / нет).

Для классификации производилось обучение сетей, показавших наилучшие результаты в прошлом исследовании: многослойного персептрона и каскадной сети прямого распространения сигнала. Алгоритмы обучения, нормализации и разделения выборки остались без изменений. Краткая информация об их архитектуре, параметрах обучения и результатах классификации, которые были получены при помощи них, приведены в таблице 3.3.

Таблица 3.3 – Результаты классификации сетями с одним скрытым слоем

Тип сети	Число нейронов в скрытых слоях	Общая доля правильно классифицированных объектов, %	Точность (% по 1 классу, % по 2 классу)	Полнота (% по 1 классу, % по 2 классу)
Многослойный персептрон	40	68.44	72, 60.97	79.46, 50.94
	160	66.91	72.57, 57.43	74.06, 55.56
Каскадная сеть прямого распространения сигнала	**40**	**69.18**	**74.08, 60.7**	**76.56, 57.47**
	160	67.17	71.58, 58.59	77.12, 51.39

Исходя из результатов классификации, приведенных в таблице 3.3, следует, что качественное изменение набора признаков не позволило повысить качество классификации, реализуемое сетями с одним скрытым слоем. Однако, как показали предыдущие эксперименты (таблицы 3.1, 3.2) каскадная сеть прямого распространения сигнала обеспечивает наилучшее качество классификации из

всех представленных в этой работе типов сетей. Основываясь на этом, целесообразно использовать для классификации данный тип сети. В таблице 3.4 приведены результаты классификации, полученные при использовании каскадных сетей с несколькими скрытыми слоями.

Таблица 3.4 – Результаты классификации многослойными сетями

Тип сети	Число нейронов в скрытых слоях	Общая доля правильно классифицированных объектов, %	Точность (% по 1 классу, % по 2 классу)	Полнота (% по 1 классу, % по 2 классу)
Каскадная сеть прямого распространения сигнала	40, 20	70.86	75.7, 62.74	77.33, 60.59
	40, 20, 10	**74.98**	**80.21, 67.09**	**78.61, 69.21**
	40, 20, 10, 5	71.77	77.19, 63.33	76.64, 64.04

На основании созданного классификатора был реализован графический интерфейс системы управления, с которой взаимодействует лицо, принимающее решение (рисунок 3.12).

Визуально местоположение ДТП можно отобразить на карте сервиса MapQuest (рисунок 3.13). На этом рисунке маркером черного цвета обозначено местоположение ДТП. Приведенные ниже маркера числа представляют географические координаты места, в котором было зафиксировано ДТП: широту и долготу. На рисунке 3.13 видно, что ДТП было зафиксировано в центре крупного населенного пункта – Миннеаполиса, а информация, приведенная на рисунке 3.12 указывает на то, что это ДТП в значительной степени снизило пропускную способность дорожного участка.

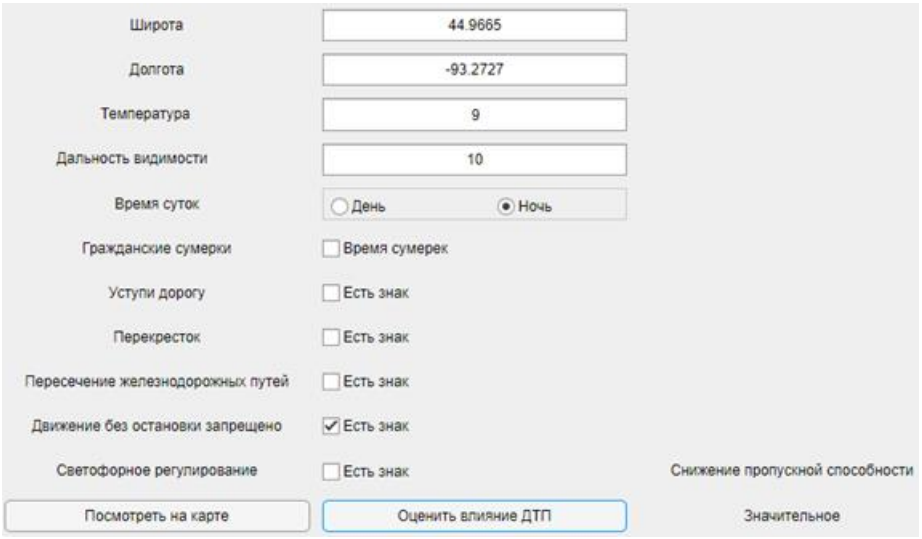

Рисунок 3.12 – Графический интерфейс системы

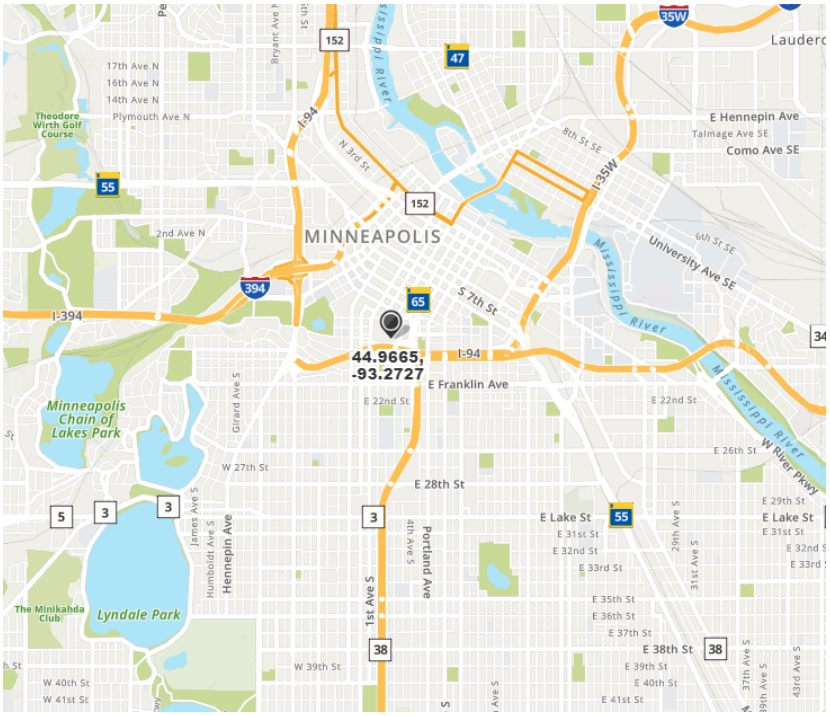

Рисунок 3.13 – Визуализация географического местоположения ДТП

Данное исследование показало, что при добавлении новой группы признаков качество результатов классификации повысилось.

3.2.3 Решение задачи кластеризации ДТП на основе разработанных методик

В данном исследовании была апробирована методика, описанная в параграфе 2.2, повышающая качество кластерного анализа при использовании слоя Кохонена. Дополнительно был проведен процесс редукции данных, что позволяет ускорить процесс работы генетического алгоритма и сократить время обучения нейронной сети. Был использован набор данных из параграфа 3.2.1.

Таким образом, оценка влияния ДТП на пропускную способность решалась путем проведения кластерного анализа следующими способами:

- слой Кохонена;
- слой Кохонена, который использует данные после проведения редукции;
- предварительно настроенный слой Кохонена, который использует данные после проведения редукции.

Экспериментально было определено, что оптимальное число итераций обучения слоя Кохонена приблизительно равно 10. Дальнейшее увеличение числа итераций не приводит к улучшению качества кластерного анализа.

Для реализации генетического алгоритма были заданы следующие параметры:

- число популяции равно 500;
- число особей в одном поколении равно 20;
- оператор кроссинговера реализован согласно алгоритму scatter скрещивания при $P_{sc} = 0.8$, где P_{sc} – вероятность скрещивания;
- процесс мутации был организован по принципу Гауссовской мутации, согласно которому случайно выбранное с помощью распределения Гаусса число добавляется к каждому элементу родительского вектора [80];
- используется два критерия останова:
  - формирование новой популяции не приводит к значимому улучшению значения функции приспособленности (значение изменилось на величину меньшую e^{-6});

- достигнут лимит на время выполнения ГА, либо лимит на число поколений (максимальное время выполнения – 2000 секунд).

Поскольку генетический алгоритм является эвристическим алгоритмом, будет более корректно резюмировать результаты кластерного анализа по ряду проведенных экспериментов. В диссертационной работе было проведено 25 экспериментов, в каждом из которых кластерный анализ проводился несколькими методами. Качество проведенной кластеризации оценивалось индексами Рэнда: RI (доля объектов, для которых исходное и полученное разбиения согласованы; ARI (мера расстояния между различными разбиениями выборки).

На первом этапе исследования кластерный анализ производился с помощью: слоя Кохонена, как без предварительной нормализации, так и с нормализацией, и после проведения редукции с помощью автокодировщика. В таблице 3.5 приведены результаты проведения первого этапа анализа.

Таблица 3.5 – Сравнение результатов первого этапа анализа

Метод кластерного анализа	Среднее значение RI по экспериментам	Среднее значение ARI по экспериментам	Среднее время обучения, с
Слой Кохонена (без нормализации и редукции)	0.5006	0.0459	26.985
Слой Кохонена (нормализация)	0.5067	-0.0048	27.2386
Слой Кохонена (автокодировщик)	**0.5023**	**0.0029**	**24.7662**

Использование редуцированного набора данных позволило немного сократить издержки времени на обучение слоя Кохонена, не теряя при этом качество полученных результатов.

На втором этапе исследования был применен генетический алгоритм для нахождения начальной матрицы весов слоя Кохонена по разработанной методике.

Результаты экспериментов приведены в таблице 3.6. Для сравнения в 1 строке этой таблицы присутствует наилучшее решение задачи кластеризации из первого этапа.

Таблица 3.6 – Сравнение результатов второго этапа анализа

Метод кластерного анализа	Среднее значение RI по экспериментам	Среднее значение ARI по экспериментам
Слой Кохонена (автокодировщик)	0.5023	0.0029
Слой Кохонена (автокодировщик и генетический алгоритм)	**0.5148**	**0.0086**

Получено увеличение средних арифметических значений индексов Рэнда при предварительной настройке матрицы весов Кохонена, что сигнализирует об улучшении качества кластерного анализа. На рисунке 3.14 визуализирован результат одного из экспериментов.

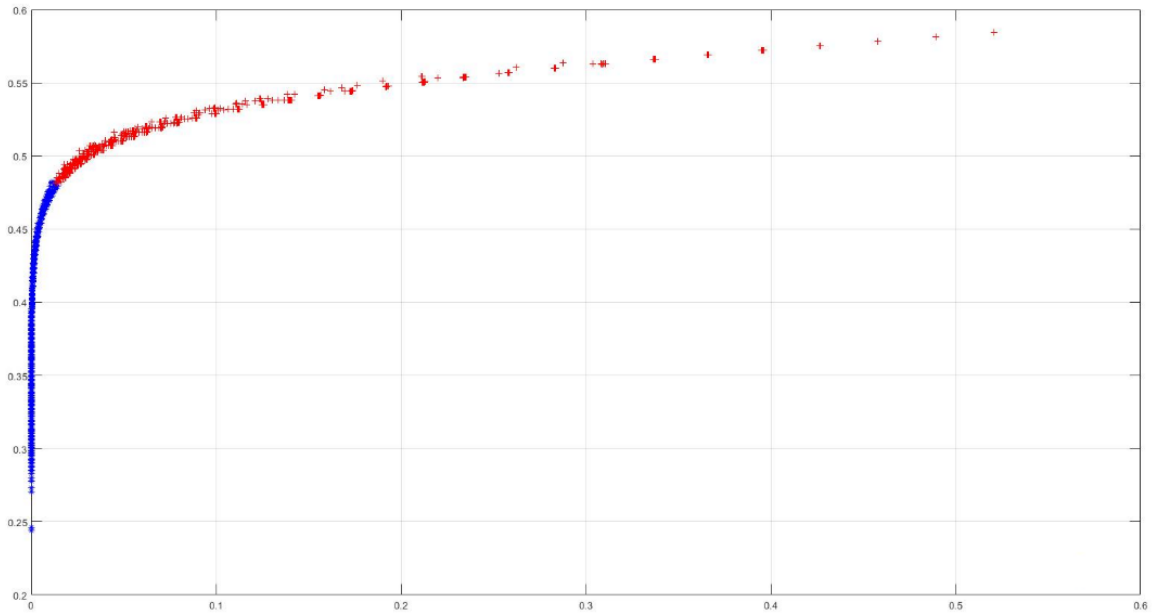

Рисунок 3.14 – Разделение объектов на кластеры

Объекты одной группы обозначены синими «*», другой – красными «+».

В результате классификатор, реализованный слоем Кохонена с применением данной методики будет более точно определять серьезность ДТП по сравнению с классическим использованием слоя Кохонена, и как следствие, более корректно оценивать влияние аварии на загруженность участка дороги, на котором она произошла.

В дополнение к этому разработанная методика по повышению качества кластерного анализа прошла апробацию на различных предметных областях, таких как: машиностроение, биология, банковская сфера, сельское хозяйство. Результаты изложены в работах [92-96]. Подробнее это описывается в параграфе 3.3.

Эти исследования указывают на возможность использования методики в других предметных областях.

3.3 Апробация разработанной методики повышения качества кластеризации при анализе объектов других предметных областей

3.3.1 Исследование способов кластеризации деталей машиностроения на основе нейронных сетей

Совершенствование технологической подготовки машиностроительного производства за счет его автоматизации является в настоящее время одним из важнейших направлений в управлении предприятием этой отрасли.

Для сокращения объемов технологической подготовки важную роль играет групповой метод, а именно метод унификации. При его применении для групп однородной по тем или иным конструктивно-технологическим признакам продукции устанавливаются однотипные высокопроизводительные методы обработки с использованием однородных и быстро переналаживаемых станков. Соответственно, чем выше уровень унификации технологии на базе группового метода, тем проще и рациональнее организационные формы производства [97].

Зачастую основу унификации технологии составляет классификация продукции. В ходе разделения изготавливаемой продукции на однородные группы может решаться как задача классификации, так и задача кластерного анализа.

В качестве данных для проведения кластерного анализа использовалось признаковое описание 34 деталей машиностроительного производства – втулок, полученное из [97]. Набор признаков каждой детали включает: максимальный диаметр; высоту; количество внешних диаметров; количество внутренних диаметров; количество конических поверхностей; количество резьб.

Была проведена редукция данных с помощью факторного анализа. Для данной выборки допустимое число факторов лежит в диапазоне от 1 до 3. Для точного определения числа факторов были проанализированы факторные нагрузки (табл. 3.7-3.8).

Таблица 3.7 – Факторные нагрузки для 2 факторов

Факторы / Признаки	Фактор 1	Фактор 2
Максимальный диаметр	0.241	0.968
Высота	0.019	0.32
Количествово внешних диаметров	0.018	0.601
Количество внутренних диаметров	0.229	-0.047
Количество конических поверхностей	0.968	0.242
Количество резьб	0.801	0.26

Таблица 3.8 – Факторные нагрузки для 3 факторов

Факторы / Признаки	Фактор 1	Фактор 2	Фактор 3
Максимальный диаметр	0.457	0.584	0.182
Высота	0.211	0.073	0.972
Количество внешних диаметров	0.081	0.984	-0.143
Количество внутренних диаметров	0.167	0.06	-0.344
Количество конических поверхностей	0.988	0.063	-0.122
Количество резьб	0.832	0.272	0.013

Приведенные факторные нагрузки в таблице 3.7 указывают на то, что два фактора не могут быть использованы, так как признак «количество внутренних диаметров» имеет слабую степень корреляции с обоими факторами [92]. Данный подход был использован в работах автора диссертационного исследования для определения числа нейронов в автокодировщике при проведении процесса редукции данных, являющийся частью разработанной автором методики.

Согласно статье [98] число кластеров для рассматриваемой выборки равно 6, однако принадлежность каждой втулки определенному кластеру достоверно неизвестна, по этой причине решение задачи классификации не представлялось возможным, и в рамках диссертационной работы решалась задача кластеризации при помощи слоя Кохонена и карты Кохонена.

Таблица 3.9 содержит полученные результаты кластерного анализа, а также результаты из статьи [99], в которой были использованы следующие алгоритмы: нечеткая кластеризация (FCM); FOREL; K-средних (K-means); а также иерархический метод кластеризации.

Таблица 3.9 – Результаты кластерного анализа

№ кластера	№ деталей, FCM	№ деталей, FOREL	№ деталей, K-средних	№ деталей, Иерархический метод	№ деталей, Слой Кохонена	№ деталей, Карта Кохонена
1	2 3 4 5 26 33	2 3 4 5 26 33	2 3 4 5 26 33	1 2 3 4 5 8 9 10 11 26 30 32 33 34	2 3 4 5 26 33	2 3 4 5 26 33
2	1 8 10 11 30 32 34	1 8 9 10 11 30 32 34	1 10 32	13	1 10 11 32 34	1 11 32
3	13 31	13 31	12 13 31	12 29	12 16 18 24 29	12 29
4	19 20 21 23	19 20 21 22 23 25 27	19 20 21 22 23 25 27	19 20 21 22 23 25 27 6 7 14 15 16 17 18 28	19 20 21 22 23 25 27	19 20 21 22 23 25 27
5	12 22 25 27	12 24 29	8 9 11 30 34	31	8 9 13 30 31	8 9 10 13 30 31 34
6	6 7 14 15 16 17 18 24 29	6 7 14 15 16 17 18 28	6 7 14 15 16 17 18 24 28 29	24	6 7 14 15 17 28	6 7 14 15 16 17 18 24 28

Как видно из таблицы 3.9, можно обнаружить, что разные алгоритмы обеспечивают схожие разбиения на кластеры. Особо стоит обратить внимание на результаты кластерного анализа при использовании сетей Кохонена - они имеют наибольшее сходство с результатом алгоритма К-средних [93].

С помощью индексов Рэнда дана количественная оценка сходству между результатами, полученными с помощью алгоритма К-средних и слоя Кохонена. Значение индекса Рэнда – 0,9323, а отрегулированного индекса Рэнда – 0,7466. Оба значения индексов указывают на сильное сходство между полученными результатами.

Этот результат указал на возможность определения начальной матрицы весов слоя Кохонена путем использования концепции генетического алгоритма, которая будет задействовать основанные этапы алгоритма К-средних. Что послужило основой для разработанной автором методики по повышению качества кластерного анализа.

Стоит отметить, что для проведения кластерного анализа выборка обладает следующими неблагоприятными свойствами:

- малое число объектов при сравнительно большом, по отношению к числу этих объектов, числе кластеров: 34 объекта и 6 кластеров;
- малое число объектов при сравнительно большом числе признаков, служащих описанием для объектов данной выборки: 34 объекта и 6 признаков.

Небольшое число объектов является самым главным недостатком данной выборки, по причине которого разработанная методика не оказывает должного воздействия на повышение качества кластерного анализа в данном исследовании. Однако приведенные в параграфах 3.2.3, 3.3.2 и 3.3.3 исследования явно указывают на повышение качества проводимого кластерного анализа при наличии достаточного числа объектов в анализируемой выборке.

3.3.2 Анализ денежных купюр Европейского центрального банка

Фальшивомонетничество берет свое начало с появлением первых металлических монет. Но со временем популярность монет снизилась и на сегодняшний день широко используются бумажные купюры. По причине этого подделка бумажных купюр фальшивомонетчиками является проблемой глобального уровня. Во всем мире подделка билетов банка является тяжким преступлением. Улучшение процесса выявление поддельных банкнот является приоритетной задачей во всем мире.

Цель данного исследования – апробация предлагаемой автором методики повышения качества кластерного анализа на реальных данных, имеющих практическую ценность.

Кластерный анализ проводился на наборе данных из репозитория для различных задач машинного обучения [100]. Этот набор описывает банкноты Европейского центрального банка. Исходная выборка содержит 1372 наблюдений, разделенных на 2 группы – подлинные и фальшивые купюры.

Авторы этого набора данных, содержащего информацию об исследуемых банкнотах, получили его путем применения различных методов вейвлет-преобразований на множестве исходных изображений купюр [101]. В результате каждый объект описывается 4 признаками: дисперсия изображения, коэффициент асимметрии изображения, коэффициент эксцесса изображения, энтропия изображения.

К достоинствам данной выборки можно отнести следующее:

1. Относительно большой объем анализируемых данных при малом количестве кластеров, что делает процесс работы алгоритмов не столь затратным по объему производимых вычислений, сохраняя при этом достаточное число объектов для обучения сети.

2. Наличие целевого вектора (вектора, содержащего истинные номера кластеров объектов).

3. Поскольку создатели выборки исследовали реальные физически существовавшие купюры (подлинные и фальшивые), то результаты работы алгоритмов, реализующих кластерный анализ, будет представлять научный, а также практический интересы.

Для проведения кластерного анализа применялись 3 метода: метод k-means, слой Кохонена и предварительно настроенный слой Кохонена, матрица весов которого прошла предварительную настройку согласно предлагаемой автором диссертационной работы методике.

Экспериментально было определено, что оптимальное число итераций обучения слоя Кохонена приблизительно равно 100. Дальнейшее увеличение числа итераций не приводит к улучшению качества кластерного анализа. Определение качества проведенного анализа осуществляется с помощью индексов Рэнда.

Также экспериментально было выявлено, что для генетического алгоритма возможно использование следующих параметров:

- число популяций равно 500;
- число особей в одном поколении равно 50;
- оператор кроссинговера реализован согласно алгоритму scatter скрещивания при $P_{sc} = 0.8$, где P_{sc} – вероятность скрещивания;
- процесс мутации был организован по принципу Гауссовской мутации, согласно которому случайно выбранное с помощью распределения Гаусса число добавляется к каждому элементу родительского вектора [60];
- используется два критерия останова:
  - формирование новой популяции не приводит к значимому улучшению значения функции приспособленности (значение изменилось на величину меньшую e^{-6});
  - достигнут лимит на время выполнения ГА, либо лимит на число поколений (максимальное время выполнения – 2000 секунд).

Поскольку генетический алгоритм относится к эвристическим, то будет корректнее резюмировать результаты кластерного анализа с его использованием по ряду проведенных экспериментов. В диссертационной работе было проведено 25 экспериментов, в каждом из которых кластерный анализ проводился несколькими методами. В процессе исследования было выялено, что меньшее число проводимых экспериментов не отражает возможных изменений индексов Рэнда.

На рисунке 3.15 отображены значения индексов Рэнда, вычисленные по результатам экспериментов [95]. Аналогично, на рисунке 3.16 отображены значения отрегулированных индексов Рэнда [95].

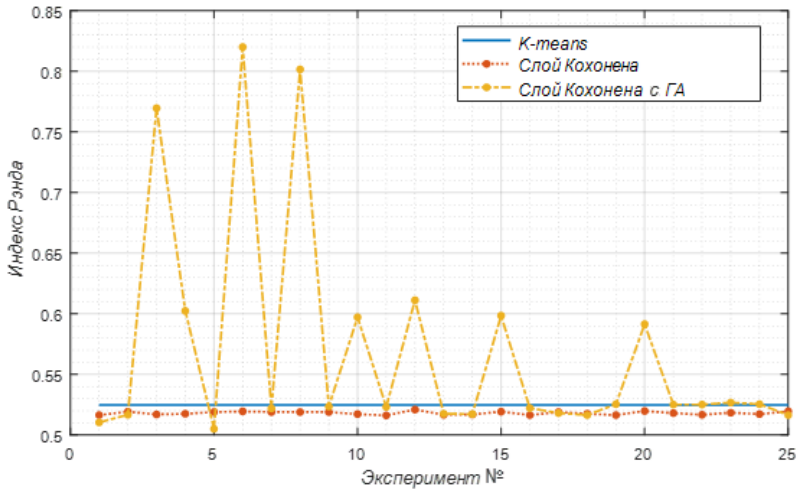

Рисунок 3.15 – Индексы Рэнда

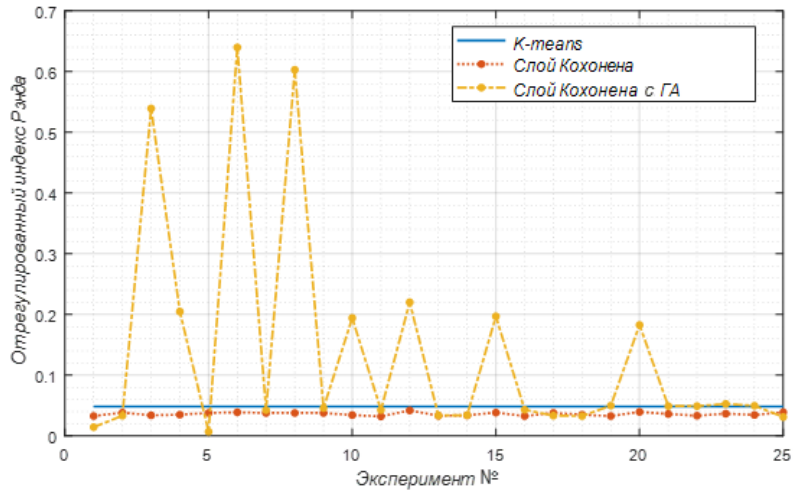

Рисунок 3.16 – Отрегулированные индексы Рэнда

Чем больше значения индексов, тем лучше качество кластерного анализа. Слой Кохонена, обученный с предварительной настройкой матрицы весов сети, обладает широким диапазоном значений индексов. На приведенных рисунках 3.15-3.16 в 8 из 25 экспериментов значения обоих индексов были выше (в 3 экспериментах индексы имеют значения, значительно превосходящие средние значения индексов, полученных другими методами), чем в остальных рассматриваемых методах. В 4 экспериментах индексы был ниже, чем индексы во всех других методах.

Таблица 3.10 содержит средние значения индексов Рэнда по результатам проведенных экспериментов, которые были отображены графически на рисунках 3.15-3.16.

Таблица 3.10 – Значения индексов Рэнда

Метод кластерного анализа	Среднее значение RI по экспериментам	Среднее значение ARI по экспериментам
K-means	0.5249	0.0485
Слой Кохонена (без предварительной настройки)	0.5182	0.0362
Слой Кохонена (с предварительной настройкой)	**0.5692**	**0.1372**

Среднее значение индексов Рэнда, вычисленное по результатам, полученным от слоя Кохонена обученного с предварительной настройкой матрицы весов сети за 25 экспериментов, выше соответствующих значений: K-means (RI на 0.0444, ARI на 0.0886), слоя Кохонена обученного без предварительной настройки (RI на 0.0511, ARI на 0.1009). Из этого следует, что качество кластерного анализа, полученное слоем Кохонена с использованием генетического алгоритма для предварительной настройки матрицы весов, выше по сравнению с качеством, полученным в результате работы других рассматриваемых в этом исследовании методов.

3.3.3 Анализ урожайности картофеля в различных районах Иркутской области

На сегодняшний день различные решения задач интеллектуального анализа данных осуществляются практически во всех отраслях промышленности, сельскохозяйственная промышленность не является исключением. Основными продуктами, производимыми этой отраслью в России являются: зерно, мясо, молоко, яйца и картофель.

Картофель является важным элементом сбалансированного, здорового питания человека. Являясь незаменимым продуктом питания, картофель представляет собой важный источник витамина С (аскорбиновая кислота). Кроме того, в картофеле содержатся витамины В1 (тиамин), В2 (рибофлавин), РР (никотиновая кислота), В6 (пиридоксин), а также каротин, который в организме человека и животных превращается в витамин А.

Картофелеводство в нашей стране до 90-х гг. осуществлялось на основе интенсивно-экстенсивных методов ведения хозяйства – повышение уровня механизации технологических процессов, применение средств химизации, увеличение площади мелиоративных земель.

По данным Федеральной службы государственной статистики в 2018 году уровень самообеспечения (в процентах) приведенными выше продуктами следующий: зерно (147,2), мясо (95,7), молоко (83,9), яйца (98,8), картофель (95,3) [102]. Исходя из приведенной статистики заметно, что продукты производство которых связано с растениеводством имеют более высокие показатели, однако уровень самообеспечения картофелем на 51,9% ниже уровня самообеспечения зерном.

Тема производства и экономической роли картофеля в мире всегда была актуальна, и на сегодняшний день не перестает быть одной из приоритетных в сфере сельского хозяйства.

Традиционно личные подсобные хозяйства населения характеризуются низкой эффективностью производства: высокой долей ручного труда, нарушением научно-обоснованных норм и иных регламентов внесения удобрения, частым использованием некондиционных семян, нарушением севооборота и т. д.

Рост затрат на удобрения, сельскохозяйственную технику, горюче-смазочные материалы и др. на фоне низкой урожайности привел к резкому росту себестоимости картофеля.

Все это подчеркивает высокую значимость изучения проблем и разработки стратегических направлений развития рынка картофеля в России в целом и его регионах в отдельности, в частности, в Иркутской области.

В данном исследовании был проведен анализ подавляющей части районов Иркутской области с целью выявления неких групп (кластеров), результаты этого анализа могли бы послужить началом разработки новой стратегии развития районов с целью повышения урожайности картофеля.

Для проведения любого анализа необходимы данные об изучаемом объекте, явлении и т.д. Сбором статистических данных занимается Федеральная служба государственной статистики по Иркутской области. Официальный сайт этой организации (irkutskstat.gks.ru) содержит ссылку на другой сайт (gks.ru/dbscripts/munst/munst25/DBInet.cgi), предоставляющий пользователям свободный на чтение доступ к базе данных [103].

Эта база хранит огромное множество данных из самых разных сфер деятельности: строительство, спорт, образование, здравоохранение и, разумеется, данные о сельском хозяйстве в регионе.

На сайте, через который пользователь взаимодействует с базой данных, реализована функция экспорта статистических данных в различные форматы, в частности, в формат XLS. Таким образом были получены данные по 15 муниципальным районам (Мамско-Чуйскому, Нижнеилимскому, Братскому, Усть-Илимскому, Усть-Кутскому, Катангскому, Киренскому, Бодайбинский,

Зиминскому, Заларинскому, Черемховскому, Слюдянскому, Усольскому, Шелеховскому и Иркутскому) в период с 2007 по 2018 гг.

Однако сформированные отчеты содержат данные в непригодном для анализа нейронной сетью формате (рисунок 3.17).

	A	B	C	D
1	Иркутский муниципальный район, Муниципальный район, Иркутский муниципальный район,			
2		2007	2008	2009
3	Вся посевная площадь			
4	Посевные площади сельскохозяйственных культур, гектар	45574,51	45266,86	43295,4
5	Зерновые и зернобобовые культуры - всего			
6	Посевные площади сельскохозяйственных культур, гектар	18986,1	16796,58	18090,2
7	Валовые сборы сельскохозяйственных культур, центнер	336122	333476	304678
79	Картофель			
80	Посевные площади сельскохозяйственных культур, гектар	5189,98	5410,84	5353,5
81	Валовые сборы сельскохозяйственных культур, центнер	916986	843454	747434
82	Урожайность сельскохозяйственных культур (в расчете на убранную площадь), центнер с гектара убранной площади	176,91	157,65	141,31
83	Овощи (всего)			
84	Валовые сборы сельскохозяйственных культур, центнер	297469	299470	276775

Рисунок 3.17 – Пример сформированного XLS документа

При анализе урожайности картофеля в различных муниципальных районах области целесообразно использовать следующие 4 признака, характеризующих исследуемый объект: всю посевную площадь в районе (гектар), посевную площадь картофеля (гектар), валовые сборы продукта (центнер), а также его урожайность (центнеров с одного гектара убранной площади).

Рисунок 3.17 демонстрирует непригодный формат данных, однако из него можно получить пригодную для анализа выборку, используя модуль препроцессинга.

Алгоритм препроцессинга этих данных имеет смысл описать в виде диаграммы DFD (рисунок 3.18).

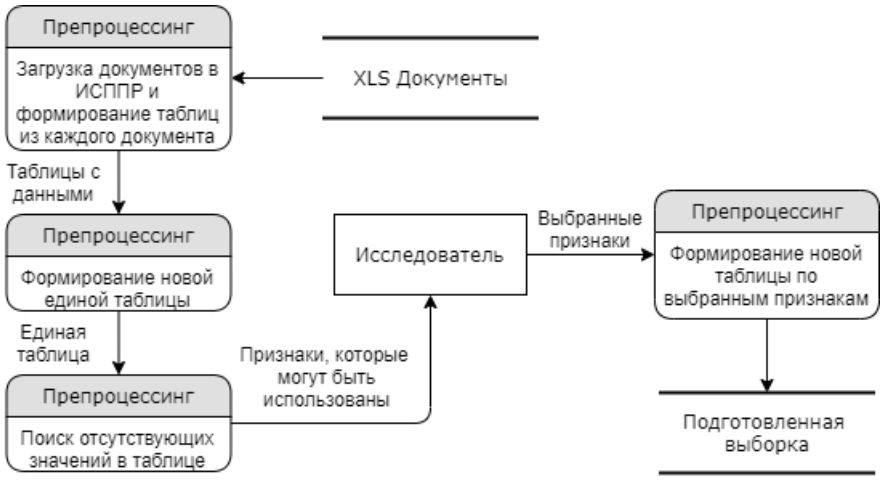

Рисунок 3.18 – Диаграмма DFD, описывающая препроцессинг данных

В результате была получена выборка, содержащая 180 объектов, каждое из которых описывается 4 ранее названными признаками.

Поскольку Иркутская область занимает обширные территории, то такие показатели как: средняя температура, количество осадков, а также количество солнечных дней может в значительной степени варьироваться в зависимости от географического положения муниципального района. Исходя из этого, был сформирован целевой вектор, разделяющий все муниципальные районы на южные и северные.

Для проведения анализа был использован слой Кохонена, иными словами, была произведена кластеризация.

Признаки в сформированной выборке имеют очень широкий диапазон значений от 22 до 916986, поэтому их необходимо нормализовать к перед анализом. Предлагаемый автором иной способ «нормализовать» данные (получить выборку, значения в которой лежали бы в некотором определенном диапазоне) состоит в использовании автокодировщика, который также произведет при этом редукцию данных, что позволит сократить время обучения сети.

В таблице 3.11 приведены результаты первого этапа анализа. Этот этап состоял из 25 экспериментов, в таблице приведены средние показатели по всем экспериментам.

Таблица 3.11 – Сравнение результатов первого этапа анализа

Метод кластерного анализа	Среднее значение RI по экспериментам	Среднее значение ARI по экспериментам	Среднее время обучения, с
Слой Кохонена (без нормализации и редукции)	0.5531	0.1067	8.639
Слой Кохонена (нормализация)	0.6465	0.293	8.6547
Слой Кохонена (автокодировщик)	**0.6654**	**0.3308**	**8.6359**

При использовании автокодировщика получено наилучшее решение задачи кластеризации, а время на получение этого решения чуть меньше, чем в других случаях.

На рисунках 3.19-3.20 представлены исследуемые объекты, описанные двумя кодированными значениями.

Все районы, которые были отнесены к северным на данном рисунке отмечены синими «*», а южные, соответственно, красными «+». Отображаемое разделение было построено согласно сформированному целевому вектору.

На рисунке 3.19 выделены наиболее выраженные представители районов Крайнего Севера (Бодайбинский, Мамско-Чуйскому, Катангскому).

Районы, обладающие наилучшими показателями должны быть приближены к области, выделенной на рисунке 3.20. Объекты в ней - это Иркутский район в разные периоды времени.

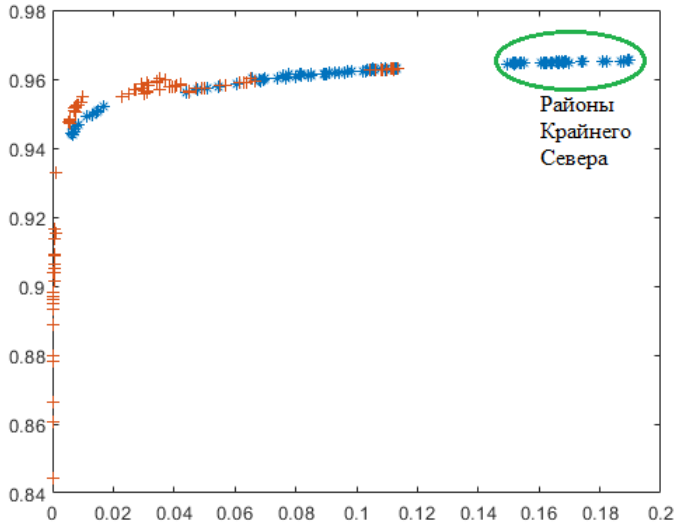

Рисунок 3.19 – Визуализация исследуемых объектов

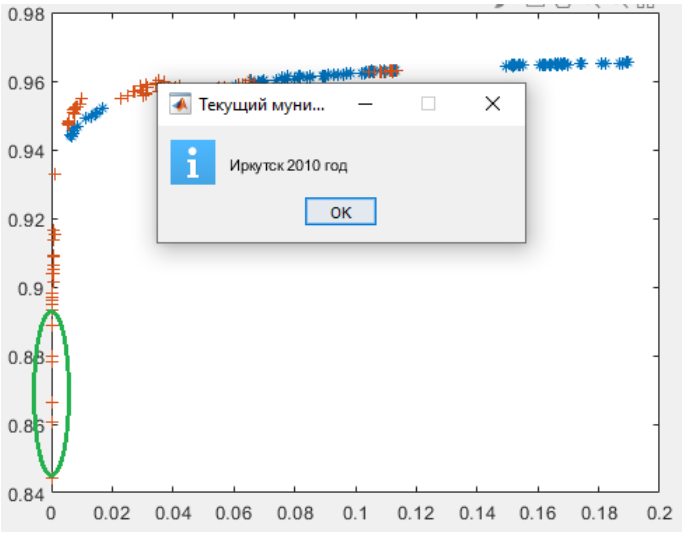

Рисунок 3.20 – Визуализация исследуемых объектов

Из северных районов наиболее эффективным для производства картофеля является Братский район, скопление синих «*» среди красных «+». Также стоит отметить наименее эффективный среди южных – Слюдянский район.

На втором этапе исследования был применен генетический алгоритм для нахождения начальной матрицы весов слоя Кохонена. В отличие от исследования, связанного с банковскими банкнотами, в данном случае генетический алгоритм применяется к данным, прошедшим редукцию с помощью автокодировщика.

Один параметр генетического алгоритма (число особей в одном поколении) был изменен в силу специфики данных и числа кластеров, и был принят равным 10.

Второй этап также включает 25 экспериментов для более объективных результатов. Результаты экспериментов приведены в таблице 3.12. С целью сравнения в 1 строке таблицы присутствует наилучшее решение задачи кластеризации из первого этапа.

Таблица 3.12 – Сравнение результатов второго этапа анализа

Метод кластерного анализа	Среднее значение RI по экспериментам	Среднее значение ARI по экспериментам
Слой Кохонена (автокодировщик)	0.6654	0.3308
Слой Кохонена (автокодировщик + генетический алгоритм)	**0.6815**	**0.3629**

Рисунок 3.21 служит для отображения разделенных на кластеры анализируемых объектов.

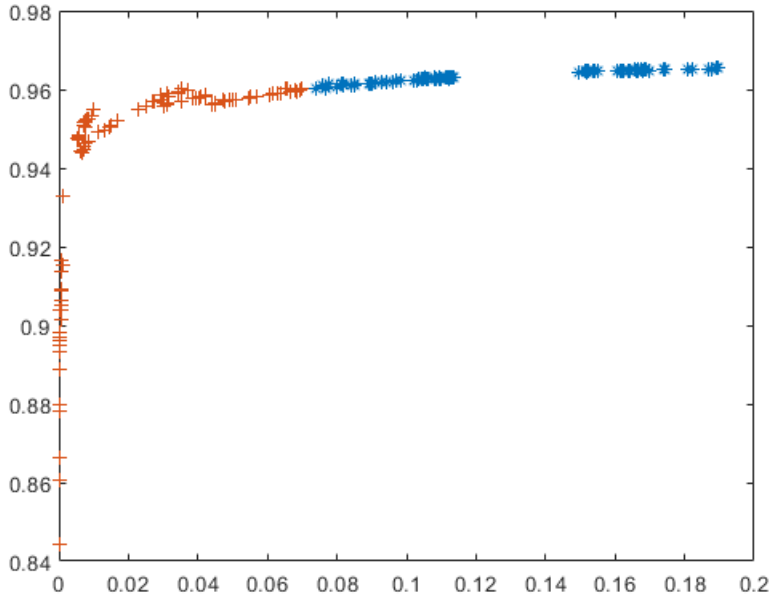

Рисунок 3.21 – Разделение объектов на кластеры (представлено наилучшее из полученных решений)

Легко заметить, что Братский район был отнесен к южным, поскольку его показатели по производству картофеля высоки. Причиной этого могут служить достаточно высокие, для северных районов, показатели: число населения и его плотность, уровень экономического развития. В то же время, Слюдянский район был отнесен к северным, что может говорить о низких показателях этого района.

Данное исследование показало, что разработанная методика может быть успешно применена для повышения качества кластерного анализа урожайности картофеля, что позволяет выделить районы области, где возможно повышение урожайности в силу их географического положения.

Выводы по главе 3

1. Программно реализована СППР в системе MATLAB. Разработанная система позволяет с помощью интуитивно понятного интерфейса использовать нейронные сети для решения задач классификации и кластеризации технических объектов.

2. Разработанная СППР прошла апробацию при решении задач классификации и кластеризации технических объектов, в частности были сформулированы и решены три задачи, связанные с классификацией ДТП по уровню их влияния на пропускную способность дорожного участка. Все исследования показали успешное применение нейронных сетей прямого распространения для классификации и кластеризации ДТП.

3. Для апробации разработанной методики было проведено два исследования, в ходе которых решалась задача кластеризации объектов других отраслей промышленности. Выявлено, что ее применение повышает качество результатов кластеризации.

ЗАКЛЮЧЕНИЕ

В диссертационной работе были проведены исследования по применению различных методов интеллектуального анализа данных для классификации и кластеризации технических объектов, в частности ДТП. Методы редукции данных, а также генетический алгоритм были применены совместно для решения задач классификации и кластеризации; в одном случае это позволило улучшить качество кластерного анализа, в другом – сократить затраты времени на решение задачи классификации.

Основными результатами работы являются следующие:

1. Разработана методика, использующая совместно алгоритм кластеризации К-средних и генетический алгоритм, позволяющая инициализировать матрицу весов слоя Кохонена, обеспечивающая повышение качества решения задачи кластеризации. Данная методика прошла апробацию на различных предметных областях (включая регулирование характеристик дорожных участков). Установлено, что при ее использовании возрастают значения индексов Рэнда.

2. Проведены исследования по кластерному анализу с помощью сети Кохонена на различных предметных областях, в ходе которых сравнивалось качество анализа с использованием данных, нормализованных общепринятыми статистическими подходами, и данных, редуцированных с помощью автокодировщика. При использовании последних значения индексов Рэнда в большинстве случаев выше.

3. Разработана модель функционирования системы «Анализ данных» путем использования математического аппарата сетей Петри, что явилось подтверждением корректности спроектированной архитектуры системы.

4. Основываясь на результатах моделирования, была реализована СППР в виде программного комплекса «Анализ экспериментальных данных на основе нейронных сетей», используемая для построения классификаторов с использованием нейросетевых и эвристических методов.

5. Произведена апробация реализованной системы при классификации и кластеризации технических объектов, в частности было определено, что при классификации ДТП по степени их влияния на пропускную способность дорожного участка в среднем общая доля правильно классифицированных объектов составляет 75%. Дополнительно показана возможность применения разработанной методики при решении задач классификации и кластеризации в других предметных областях (сельское хозяйство, биология, банковское дело).

Разработанная система использовалась при моделировании транспортных потоков в компании ООО «Центр транспортных технологий»; компанией ООО НПО ССЦ «Ангара» при анализе массивов данных в различных районах Иркутской области; в учебном процессе Института высоких технологий Иркутского национального исследовательского технического университета при организации учебного курса «Технологии обработки информации».

Результаты исследования подтверждаются наличием соответствующих актов о внедрении.

СПИСОК ЛИТЕРАТУРЫ

1. Steinbach Michael. Introduction to Data Mining. 2nd Edition / Michael Steinbach, Pang-Ning Tan, Vipin Kumar. – Pearson Education, 2018. – 864 p.

2. Witten Ian H. Data Mining: Practical Machine Learning Tools and Techniques. 3rd Edition / Ian H. Witten, Eibe Frank, Mark A. Hall. – Morgan Kaufmann, 2011. – 664 p.

3. Прикладная статистика: классификация и снижение размерности / С. А. Айвазян [и др.]. – М.: Финансы и статистика, 1989. – 607 с.

4. Tryon R. C. Cluster analysis / R. C. Tryon. – London: Ann Arbor Edwards Bros, 1939. – 139 p.

5. Мандель, И. Д. Кластерный анализ / И. Д. Мандель. – М.: Финансы и статистика, 1988. – 176 с.

6. Миркин, Б. Г. Методы кластер-анализа для поддержки принятия решений: обзор / Б. Г. Миркин. – М.: Высшая школа экономики, 2011. – 88 с.

7. Жамбю, М. Иерархический кластер-анализ и соответствия / М. Жамбю. – М.: Финансы и статистика, 1988. – 345 с.

8. Wierzchon S. T. Modern Algorithms of Cluster Analysis / S. T. Wierzchon, M. A. Klopotek. – Springer, 2018. – 421 p.

9. Steinhaus H. Sur la division des corps materiels en parties / H. Steinhaus // Bulletin de l’academie polonaise des sciences. – 1956. – Vol. 4(12). – P. 801-804.

10. Lloyd S. P. Least Squares Quantization in PCM / S. P. Lloyd // Transactions on information theory. – 1982. – Vol. 28(2). – P. 129-137.

11. Kaufman L. Finding Groups in Data: An Introduction to Cluster Analysis / L. Kaufman, P. Rouseeuw. – Hoboken, NJ: John Wiley & Sons, Inc., 1990. – 369 p.

12. MATLAB Documentation [Электронный ресурс]. – Режим доступа: https://www.mathworks.com/help/stats/kmeans.html, свободный.

13. Rand W. M. Objective criteria for the evaluation of clustering methods / W. M. Rand // Journal of the American Statistical Association. – 1971. – Vol. 66. – P. 846–850.

14. Santos J. On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification / J. Santos, M. Embrechts // Springer-Verlag Berlin Heidelberg. – 2009. – Vol. 2. – P. 175–184.

15. McCulloch W. A logical calculus of the ideas immanent in nervous activity / W. McCulloch, W. Pitts // Bulletin of Mathematical Biology. – 1943. – Vol. 5(4). – P. 115-133.

16. Rosenblatt F. The perceptron a perceiving and recognizing automaton (project para) / F. Rosenblatt // Buffalo, NY: Cornell aeronautical laboratory, Inc. – 1957.

17. Хайкин, С. Нейронные сети: полный курс, 2-е издание / С. Хайкин. – М.: Издательский дом «Вильямс», 2006. – 1104 с.

18. Pham D. Training multilayered perceptrons for pattern recognition: a comparative study of four training algorithms / D. Pham, S. Sagiroglu // International Journal of Machine Tools and Manufacture. – 2001. – Vol. 41. – P. 419-430.

19. Sharma B. Comparison of neural network training functions for hematoma classification in Brain CT Images / B. Sharma, K. Venugopalan // Journal of Computer Engineering. – 2014. – Vol. 16. – P. 31-35.

20. Alsmadi M. K. Back propagation algorithm: The best algorithm among the multi-layer perceptron algorithm / M. K. Alsmadi, K. B. Omar, S. A. Noah // International Journal of Computer Science and Network Security. – 2009. – Vol. 9(4). – P. 378-383.

21. Karras D. A. Comparison of learning algorithms for feedforward networks in large scale networks and problems / D. A. Karras, S. J. Perantonis // Proceedings of International Joint Conference on Neural Networks. – 1993. – P. 532-535.

22. Solano Y. A comparative study of eight learning algorithms for artificial neural networks based on a real application / Y. Solano, H. Ikeda // IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences. – 1998. – P. 355-357.

23. Hagan M. T. Training feedforward networks with the Marquardt algorithm / M. T. Hagan, M. B. Menhaj // IEEE Transactions on Neural Networks. – 1994. – Vol. 5(6). – P. 989-993.

24. Cömert Z. A study of artificial neural network training algorithms for classification of cardiotocography signals / Z. Cömert, A. Kocamaz // Bitlis Eren University Journal of Science and Technology. – 2017. – Vol. 7(2). – P. 93-103.

25. Stager F. Three methods to speed up the training of feedforward and feedback perceptrons / F. Stager, M. Agarwal // Neural Networks. – 1997. – Vol. 10(8). – P. 1435-1443.

26. RoyChowdhury P. Dynamic tunneling technique for efficient training of multilayer perceptrons / P. RoyChowdhury, Y. Singh, R. Chansarkar // IEEE Transactions on Neural Networks. – 1999. – Vol. 10(1). – P. 48-55.

27. Zhou G. Advanced neural-network training algorithm with reduced complexity based on Jacobian deficiency / G. Zhou, J. Si // IEEE Transactions on Neural Networks. – 1998. – Vol. 9(3). – P. 448-453.

28. Kim C. Training two-layered feedforward networks with variable projection method / C. Kim, J. Lee // IEEE Transactions on Neural Networks. – 2008. – Vol. 19(2). – P. 371-375.

29. Hannan J. M. Comparison of fast training algorithms over two real problems / J. M. Hannan, J. M. Bishop // Proceedings of the 5th International Conference on Artificial Neural Networks. – 1997. – P. 1-6.

30. Du Y. Levenberg-Marquardt neural network algorithm for degree of arteriovenous fistula stenosis classification using a dual optical photoplethysmography sensor / Y. Du, A. Stephanus // Sensors (Basel). – 2018. – Vol. 18(7). – P. 1-18.

31. Медведев, В. С. Нейронные сети. MATLAB 6 / В. С. Медведев, В. Г. Потемкин. – М.: ДИАЛОГ-МИФИ, 2002. – 496 с.

32. MATLAB Documentation [Электронный ресурс]. – Режим доступа: https://www.mathworks.com/help/deeplearning/ref/patternnet.html, свободный.

33. MATLAB Documentation [Электронный ресурс]. – Режим доступа: https://www.mathworks.com/help/deeplearning/ref/softmax.html, свободный.

34. Moller M. F. A scaled conjugate gradient algorithm for fast supervised learning / M. F. Moller // Neural Networks. – 1993. – Vol. 6. – P. 525-533.

35. MATLAB Documentation. // URL: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.softmaxlayer.html (дата обращения: 11.12.2018).

36. MATLAB Documentation [Электронный ресурс]. – Режим доступа: https://www.mathworks.com/help/deeplearning/ref/competlayer.html, свободный.

37. MATLAB Documentation [Электронный ресурс]. – Режим доступа: https://www.mathworks.com/help/deeplearning/ref/selforgmap.html, свободный.

38. Yao Xiao. Clustering Based on Continuous Hopfield Network / Xiao Yao, Zhang Yashu, Dai Xiangguang, Yan Dongfang // Mathematics. – 2022. – Vol. 10(6). – 11 p.

39. Харман, Г. Современный факторный анализ / Г. Харман. – М.: Статистика, 1972. – 484 с.

40. Hinton G.E. Autoencoders, Minimum Description Length and Helmholtz Free Energy / G.E. Hinton // Advances in Neural Information Processing Systems. – 1993. – Vol. 6. – P. 3-10.

41. MATLAB Documentation [Электронный ресурс]. – Режим доступа: https://www.mathworks.com/help/deeplearning/ref/trainautoencoder.html, свободный.

42. Deep Learning – Different Types of Autoencoders [Электронный ресурс]. – Режим доступа: https://medium.com/datadriveninvestor/deep-learning-different-types-of-autoencoders-41d4fa5f7570, свободный.

43. Олдендерфер, М. С. Кластерный анализ. Факторный, дискриминантный и кластерный анализ / М. С. Олдендерфер, Р. К. Блэшфилд. – М.: Финансы и статистика, 1989. – 215 с.

44. Ким, О. Дж. Факторный, дискриминантный и кластерный анализ / О. Дж. Ким, Ч. У. Мьюллер, У. Р. Клекка. – М.: Финансы и статистика, 1989. – 215 с.

45. Лоули, Д. Факторный анализ как статистический метод / Д. Лоули, А. Мансвелл. – М.: Мир, 1967. – 141 с.

46. MATLAB Documentation [Электронный ресурс]. – Режим доступа: http:// mathworks.com/help/stats/factoran.html, свободный.

47. Kaiser H. F. The varimax criterion for analytic rotation in factor analysis / H. F. Kaiser // Phychometrika. – 1958. – Vol. 23. – P. 187–200.

48. Power D. J. Web-based and model-driven decision support systems: concepts and issues / D. J. Power // Americas Conference on Information Systems, Long Beach, California. – 2000. – P. 352-355.

49. Блауберг, И. В. Становление и сущность системного подхода / И. В. Блауберг, Э. Г. Юдин. – М.: Наука, 1974. – 271 с.

50. Губанов, В. А. Введение в системный анализ / В. А. Губанов, В. В. Захаров, А. Н. Коваленко. – Л.: Изд-во Ленинградского университета, 1988. – 232 с.

51. Перегудов, Ф. И. Основы системного анализа / Ф. И. Перегудов, Ф. И. Тарасенко. – Томск: Изд-во НТЛ, 1997. – 396 с.

52. Гейн, К. Структурный системный анализ: средства и методы. Часть 1 / К. Гейн. – М.: Эйгекс, 1993. – 188 с.

53. Hathaway T. Data Flow Diagrams - Simply Put!: Process Modeling Techniques for Requirements Elicitation and Workflow Analysis (Advanced Business Analysis Topics) / T. Hathaway, A. Hathaway. – BA-Experts, 2015. – 75 p.

54. Гудман, С. Введение в разработку и анализ алгоритмов / С. Гудман, С. Хидетниеми. – М.: Мир, 1981. – 368 с.

55. Колмогоров, А. Н. Теория информации и теория алгоритмов / А. Н. Колмогоров. – М.: Наука, 1987. – 304 с.

56. Марков, А. А. Теория алгорифмов. 2-е издание / А. А. Марков, Н. М. Нагорный. – М.: ФАЗИС, 1996. – 432 с.

57. Goldberg D.E. Genetic Algorithms in Search, Optimization and Machine Learning / D.E. Goldberg. – Addison-Wesley Publishing Company, Inc., 1989. – 412 p.

58. Демиденко, Е. З. Оптимизация и регрессия / Е. З. Демиденко. – М.: Наука, 1989. – 296 с.

59. Гладков, Л. А. Генетические алгоритмы / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик. – М.: ФИЗМАТЛИТ, 2006. – 320 с.

60. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия – Телеком, 2006. – 452 с.

61. Booker L. B. Classifier systems and genetic algorithms / L. B. Booker, D. E. Goldberg, J. H. Holland // Machine learning paradigms and methods. – 1990. – P. 235-282.

62. Yao X. Evolving artificial neural networks / X. Yao // Proceedings of the IEEE. – 1999. – Vol. 87(9). – P. 1423- 1447.

63. Yao X. A review of evolutionary artificial neural networks / X. Yao // International Journal of Intelligent Systems. – 2007. – Vol. 8(4). – P. 539-567.

64. Montana D. Training feedforward neural networks using genetic algorithm / D. Montana, L. Davis // Proceedings of IJCAI. – 1989. – P. 762-767.

65. Ding S. Using genetic algorithms to optimize artificial neural networks / S. Ding, L. Xu, C. Su, H. Zhu // Journal of Convergence Information Technology. – 2010. – Vol. 5(8). – P. 54-62.

66. Maulik Ujjwal. Multiobjective Genetic Algorithms for Clustering / Ujjwal Maulik, Sanghamitra Bandyopadhyay, Anirban Mukhopadhyay. – Springer-Verlag Berlin Heidelberg, 2011. – 281 p.

67. Котов, В. Е. Сети Петри / В. Е. Котов. – М: Наука, 1984. – 160 с.

68. Питерсон Дж. Теория сетей Петри и моделирование систем / Дж. Питерсон. – М.: Мир, 1984. – 264 с.

69. Марков, А. В. Анализ сетей Петри при помощи деревьев достижимости / А. В. Марков, А. А. Воевода // Журнал «Сборник научных трудов НГТУ». – 2013. – № 1. – С. 78-95.

70. Официальный сайт разработчиков среды CPN Tools [Электронный ресурс]. – Режим доступа: http://cpntools.org/, свободный.

71. Girault C. Petri Nets for Systems Engineering: A Guide to Modeling, Verification and Applications / C. Girault, R. Valk. – Springer, 2003. – 628 p.

72. Reisig W. Petri Nets: An Introduction / W. Reisig. – Springer-Verlag, 1985. – 172 p.

73. Романников, Д. О. Об использовании программного пакета CPN Tools для анализа сетей Петри / Д. О. Романников, А. В. Марков // Журнал «Сборник научных трудов НГТУ». – 2012. – № 2. – С. 105-116.

74. Харахинов, В. А. Использование сетей Петри при проектировании архитектуры программного продукта для анализа данных с помощью нейронных сетей / В. А. Харахинов, С. С. Сосинская // Научный вестник НГТУ. – 2018. – № 4(73). – С. 91-100.

75. Кетков, Ю. Л. MATLAB 7: программирование, численные методы / Ю. Л. Кетков, А. Ю. Кетков, М. М. Шульц. – СПб.: БХВ-Петербург, 2005. – 752 с.

76. Дьяконов, В. П. MATLAB 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики / В. П. Дьяконов, В. В. Круглов. – М.: СОЛОН-Пресс, 2009. – 456 с.

77. Paluszek Michael. MATLAB Machine Learning / Michael Paluszek, Stephanie Thomas. – Apress, 2016. – 335 p.

78. Kim Phil. MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence / Phil Kim. – Apress, 2017. – 162 p.

79. Paluszek Michael. MATLAB Machine Learning Recipes: A Problem-Solution Approach / Michael Paluszek, Stephanie Thomas. – Apress, 2019. – 340 p.

80. MATLAB Documentation [Электронный ресурс]. – Режим доступа: https://www.mathworks.com/help/gads/ga.html, свободный.

81. Свидетельство о государственной регистрации программы для ЭВМ 2017617294 Российская Федерация. Программный комплекс «Анализ

экспериментальных данных на основе нейронных сетей» / В. А. Харахинов, С. С. Сосинская ; заявитель и правообладатель Федеральное государственное бюджетное образовательное учреждение высшего образования «Иркутский национальный исследовательский технический университет». - № 2017614163 ; заявл. 04.05.2017 ; опубл. 04.07.2017. – 1 с.

82. Официальный сайт научно-производственной компании «ИТЕЛДОР» [Электронный ресурс]. – Режим доступа: https://iteldor.ru/products/arken/, свободный.

83. Официальный сайт IT-компании «Яндекс» [Электронный ресурс]. – Режим доступа: https://yandex.ru/company/technologies/yaprobki/, свободный.

84. Блог компании «Яндекс» [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/yandex/blog/153631/, свободный.

85. Gibbens R.J. Road traffic analysis using MIDAS data: journey time prediction / R.J. Gibbens, Y. Saacti // Computer Laboratory, University of Cambridge. – 2006. – 35 p.

86. Min Wanli. Road Traffic Prediction with Spatio-Temporal Correlations / Wanli Min, Laura Wynter, Yasuo Amemiya // IBM Research Report. Thomas J. Watson Research Center. Yorktown Heights. – RC24275 (W0706-018). – 2007. – 12 p.

87. Zheng Weizhong. Short-Term Freeway Traffic Flow Prediction: Bayesian Combined Neural Network Approach / Weizhong Zheng, Der-Horng Lee, Qixin Shi // Journal of Transportation Engineering. – 2006. – Vol. 132(2). – P. 114–121.

88. Набор данных «A countrywide Traffic Accident Dataset» [Электронный ресурс]. – Режим доступа: https://kaggle.com/sobhanmoosavi/us-accidents, свободный.

89. Электронный фонд правовых и нормативно-технических документов. Отраслевой дорожный методический документ. Методические рекомендации по оценке пропускной способности автомобильных дорог (ОДМ 218.2.020-2012) [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/1200092512, свободный.

90. Calvert J. The sun altitude, azimuth, declination, zenith and sunrise and sunset for each day of the year for the city and state and immediate surrounding area listed below: City, Frankfort, state, Kentucky Unknown Binding / J. Calvert. – University of Kentucky, 1976. – 183 p.

91. US Car Accidents Severity Analysis [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/art12400/us-car-accidents-severity-analysis, свободный.

92. Харахинов, В. А. Исследование способов кластеризации деталей машиностроения на основе нейронных сетей / В. А. Харахинов, С. С. Сосинская // Журнал «Программная инженерия». – 2017. – № 4. – С. 170–176.

93. Харахинов, В. А. Классификация деталей машиностроительного производства с использованием нейронных сетей / В. А. Харахинов, С. С. Сосинская // Винеровские чтения. – 2016. – С. 19-23.

94. Харахинов, В. А. Влияние сокращения размерности пространства признаков на результаты классификации листьев различных видов растений / В. А. Харахинов, С. С. Сосинская // Журнал «Программная инженерия». – 2018. – № 2. – С. 82–90.

95. Харахинов, В. А. Генетический алгоритм как альтернатива обучения слоя Кохонена / В. А. Харахинов // Журнал «Информационные технологии». – 2018. – № 10. – С. 642-648.

96. Харахинов, В. А. Разработка инструмента для анализа результатов методической и научно-исследовательской работы преподавателей с помощью нейронных сетей / В. А. Харахинов, С. С. Сосинская // Вестник БГТУ им. В. Г. Шухова. – 2015. – № 3. – С. 110-114.

97. Митрофанов, С. П. Научная организация машиностроительного производства / С. П. Митрофанов. – Л.: Машиностроение, 1976. – 712 с.

98. Янчуковский, В. Н. Использование параллельных вычислений в кластерном анализе для формирования комплексных деталей / В. Н. Янчуковский // Вестник ИрГТУ. – 2012. – №6. – С. 25-31.

99. Янчуковский, В. Н. Параллельный итерационный кластерный анализ на основе алгоритма FOREL / В. Н. Янчуковский, С. С. Сосинская // Современные технологии. Системный анализ. Моделирование. – 2013. – № 3(39). – С. 94-99.

100. Репозиторий реальных и модельных задач машинного обучения [Электронный ресурс]. – Режим доступа: https://archive.ics.uci.edu/ml/datasets/banknote+authentication, свободный.

101. Астафьева, Н. М. Вейвлет-анализ: Основы теории и примеры применения / Н. М. Астафьева // Успехи физических наук. – 1996. – Т.166. – № 11. – С. 1145-1170.

102. Федеральная служба государственной статистики (Росстат). Сельское хозяйство в России 2019. – М., 2019. – 91 с.

103. База данных показателей муниципальных образований / Федеральная служба государственной статистики (Росстат) [Электронный ресурс]. – Режим доступа: https://gks.ru/dbscripts/munst/munst25/DBInet.cgi, свободный.

104. Харахинов, В. А. Нейросетевой подход к оценке пропускной способности дорожного участка по различным характеристикам ДТП / В. А. Харахинов // Молодая наука Сибири. – 2021. - №4(14).

105. Харахинов, В. А. Оценка времени обучения нейронной сети с предварительной редукцией данных в задаче классификации листьев различных видов растений / В. А. Харахинов // Винеровские чтения: материалы XI Всероссийской молодежной научно-практической конференции, 6-7 июня 2019 г., г. Иркутск. – Иркутск: Изд-во ИРНИТУ, 2019.

106. Kharakhinov, V. A. Information technology of accounting the impact of data reduction methods on the results of classification of plant leaves / V. A. Kharakhinov, S. S. Sosinskaya, R. S. Dorofeev, A. S. Dorofeev, R. I. Bazhenov // Applied Physics, Information Technologies and Engineering: conference proceeding (Krasnoyarsk, 25-27 September 2019) / Krasnoyarsk Science & Technology City Hall of the Russian Union of Scientific and Engineering Associations. – Krasnoyarsk, 2019. - DOI: 10.1088/1742-6596/1399/3/033006.

107. Харахинов, В. А. Оценка урожайности картофеля в различных районах Иркутской области с применением методов интеллектуального анализа данных / В. А. Харахинов // Информационные технологии интеллектуальной поддержки принятия решений: сборник трудов VIII Всероссийской научной конференции (с приглашением зарубежных ученых), 6-9 октября 2020 г., г. Уфа. – Уфа: Изд-во УГАТУ, 2021. - Т. 2. – С. 21-27.

ПРИЛОЖЕНИЕ А

Программный код основных модулей подсистемы анализа данных СППР.

```
function varargout = Main(varargin)
 gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Main_OpeningFcn, ...
                   'gui_OutputFcn',  @Main_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Main_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
clc
movegui('center')
set(handles.DataTable,'Data',{})
set(handles.FValueTable,'Data',{})
global F
F = [];
global ghmform
ghmform = guihandles(hObject);
global SelectedMethod

function varargout = Main_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function Factorrun_Callback(hObject, eventdata, handles)
global Data4Fac
Data4Fac = getappdata(handles.LoadData,'Data');
fanalfig=open('Factor_analysis.fig');
global ghfanal
ghfanal = guihandles(fanalfig);

function LoadData_Callback(hObject, eventdata, handles)
[filename]=uigetfile({'*.mat'});
    %set(handles.text7,'String','asdad');
if filename~=0
    vardata = load (filename);
    %varlist = who('-file', filename);
    varlist = fieldnames(vardata)';
    for i=1:length(varlist)
        MatSize{i} =  size(vardata.(varlist{i}));
        MinMatSize(i) = min(MatSize{i});
    end
    [val pos] = max(MinMatSize);
    setappdata(handles.LoadData,'Data',vardata.(varlist{pos}))
    set(handles.DataTable,'Data',vardata.(varlist{pos}));
    set(handles.DataTable,'Visible','on');
    set(handles.DataName2,'String',varlist{pos});
    set(handles.UseOriginal,'UserData',vardata.(varlist{pos}))
    [val pos] = min(MinMatSize);
    setappdata(handles.LoadData,'Target',vardata.(varlist{pos}));
    set(handles.StartClusterization,'Enable','on');
    set(handles.Factorrun,'Enable','on');
```

```
    set(handles.AutoEncRun,'Enable','on');
    set(handles.CVpartrun,'Enable','on');
    set(handles.GArun,'Enable','on');
end

function IterationData_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function ClusterCountData_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function ClusterCountData_KeyPressFcn(hObject, eventdata, handles)
check_user_input(hObject,eventdata.Key)

function UseKLayer_Callback(hObject, eventdata, handles)
set(handles.ClustersCount,'String','Число кластеров')

function UseOriginal_Callback(hObject, eventdata, handles)
set(handles.UseNormalizeData,'Enable','on');
get(hObject,'UserData');
set(handles.CVpartrun,'Label','Разделить исходные данные')

function UseFactorsValue_Callback(hObject, eventdata, handles)
set(handles.UseNormalizeData,'Enable','off');
global F
set(handles.FValueTable,'Data',F);
if strcmp(handles.FactorName1.UserData,'None') == 1
    set(handles.FactorName1,'String',strcat('Фактор. знач. (без вращения)'));
else
    set(handles.FactorName1,'String',strcat('Фактор. знач. (',...
        handles.FactorName1.UserData,' вращение)'));
end
set(handles.CVpartrun,'Label','Разделить факторные значения')

function UseKMap_Callback(hObject, eventdata, handles)
set(handles.ClustersCount,'String','Число кластеров')

function UseLVQ_Callback(hObject, eventdata, handles)
set(handles.ClustersCount,'String','Число кластеров')

function StartClusterization_Callback(hObject, eventdata, handles)
if get(handles.UseNormalizeData,'Value') ~= 0
    Data4train = normc(Data4train);
End

global t
ClusterCount = get(handles.ClusterCountData,'String');
ClusterCount = str2num(ClusterCount);
set(handles.ClusterCountData,'UserData',ClusterCount);
global IterCount
IterCount = get(handles.IterationData,'String');
IterCount = str2num(IterCount);
if (any(ClusterCount) == 0) || (any(IterCount) == 0)
    errordlg('Incorrect network settings');
else
    gacheck = get(handles.GAcheck,'Value');
    global selected_net
    global andatatype
    global exparam4net
    global train_time
    flag4ex = isempty(hObject.UserData);
    net_type = get(get(handles.GroupForMethod,'SelectedObject'),'Tag');
    traindata_type = get(handles.usePartOfTrainData,'Value');
    testdata_type = get(handles.usePartOfTestData,'Value');
```

```
if traindata_type == 1
    Data4train = get(handles.usePartOfTrainData,'UserData');
    Target4train = Data4train{2};
    Data4train = Data4train{1};
else
    Target4train = getappdata(handles.LoadData,'Target');
    switch get(get(handles.GroupForData,'SelectedObject'),'Tag')
        case 'UseOriginal'
            Data4train = getappdata(handles.LoadData,'Data');
        case 'UseFactorsValue'
            Data4train = get(handles.FValueTable,'Data');
            size(Data4train)
        case 'UseEncodedData'
            Data4train = get(handles.FValueTable,'Data');
    end
end
Target4train = Target4train';
Target4train = full(ind2vec(Target4train));
if testdata_type == 1
    Data4sim = get(handles.usePartOfTestData,'UserData');
    Target4sim = Data4sim{2};
    Data4sim = Data4sim{1};
else
    Target4sim = getappdata(handles.LoadData,'Target');
    switch get(get(handles.GroupForData,'SelectedObject'),'Tag')
        case 'UseOriginal'
            Data4sim = getappdata(handles.LoadData,'Data');
        case 'UseFactorsValue'
            Data4sim = get(handles.FValueTable,'Data');
            size(Data4sim)
        case 'UseEncodedData'
            Data4sim = get(handles.FValueTable,'Data');
    end
end
transt = Target4sim;
t = transt;
Data4sim = Data4sim';
switch get(get(handles.GroupForData,'SelectedObject'),'Tag')
    case 'UseOriginal'
        andatatype = 'Original';
        Data4train = Data4train';
    case 'UseFactorsValue'
        andatatype = 'Factor';
        global nowrot
        nowrot = get(handles.FactorName1,'UserData');
        FValue = handles.FValueTable.Data;
        FValue = FValue';
        Data4train = Data4train';
    case 'UseEncodedData'
        andatatype = 'Encoded';
        EncodedData = handles.FValueTable.Data;
        EncodedData = EncodedData';
        Data4train = Data4train';
end
[simm,trnetwork,train_time,WNonTNet,WTNet,net_name] =...
    net_initialization(net_type,Data4train,...
        Target4train,Data4sim,ClusterCount,IterCount,gacheck);
selected_net = net_name;
ResVect = vec2ind(simm);
ResVect = transpose(ResVect);
for i=1:length(Data4sim)
    Res(i,1)=i;
    Res(i,2)=ResVect(i);
end
setappdata(handles.StartClusterization,'R',Res)
guihandles;
global erro
global misss
global miss2cl
[err, mis, m2cl] = comperr(transt,ResVect,ClusterCount);
```

```
    erro = err; misss = mis; miss2cl = m2cl;
    set(handles.UseShowRes,'Enable','on');
    global nowf
    nowf = size(WTNet,2);
end

function UseShowRes_Callback(hObject, eventdata, handles)
Res = getappdata(handles.StartClusterization,'R');
clear FValue;
global F
FValue = F;
global gp1;
gp1 = Res;
global GAData
dat = get(handles.DataTable,'Data');
h=open('Results_on_table.fig');
h = guihandles(h);
set(h.uitable1,'visible','on');
set(h.uitable2,'visible','on');
if (size(dat,2)) <= 3
    if (size(dat,2)) == 3
        scrsz = get(groot,'ScreenSize');
        mdot = {'.','+','*','o','x'};
        mcolor = {'y','m','r','g','b','k'};
        figure('Name','Результаты кластерного анализа в координатах факторных значений',...
        'Position',[10 scrsz(4)/17 scrsz(3)/2 scrsz(4)/1.2]);
        for i=1:length(Res)
            ii = num2str(i);
            str1 = strcat('\leftarrow',' ',ii);
            inddot = 0;
            j = 1;
            while j<=max(Res(:,2))
                if Res(i,2) == j
                    inddot = fix((j-1) / length(mcolor));
                    str2 = strcat(mdot(inddot+1),mcolor(j-inddot*length(mcolor)));
                    l1 = plot3(dat(i,1),dat(i,2),dat(i,3),str2{1},'markersize',6);
                    text(dat(i,1),dat(i,2),dat(i,3),str1,'FontSize',6);
                    fl{j} = [l1];
                    break
                else
                    j = j+1;
                end
            end
            hold on
        end
    end
end
if (isempty(FValue)~=1)
    scrsz = get(groot,'ScreenSize');
    mdot = {'.','+','*','o','x'};
    mcolor = {'y','m','r','g','b','k'};
    if (size(FValue,2) == 2)
        figure('Name','Результаты кластерного анализа в координатах факторных значений',...
        'Position',[10 scrsz(4)/17 scrsz(3)/2 scrsz(4)/1.2]);
        for i=1:length(Res)
            ii = num2str(i);
            str1 = strcat('\leftarrow',' ',ii);
            inddot = 0;
            j = 1;
            while j<=max(Res(:,2))
                if Res(i,2) == j
                    inddot = fix((j-1) / length(mcolor));
                    str2 = strcat(mdot(inddot+1),mcolor(j-inddot*length(mcolor)));
                    l1 = plot(FValue(i,1),FValue(i,2),str2{1},'markersize',6);
                    text(FValue(i,1),FValue(i,2),str1,'FontSize',6);
                    fl{j} = [l1];
                    break
                else
                    j = j+1;
                end
```

```
            end
            hold on
        end
    elseif (size(FValue,2) == 3)
         figure('Name','Результаты кластерного анализа в координатах факторных
значений',...
         'Position',[10 scrsz(4)/17 scrsz(3)/2 scrsz(4)/1.2]);
         for i=1:length(Res)
             ii = num2str(i);
             str1 = strcat('\leftarrow',' ',ii);
             inddot = 0;
             j = 1;
             while j<=max(Res(:,2))
                 if Res(i,2) == j
                     inddot = fix((j-1) / length(mcolor));
                     str2 = strcat(mdot(inddot+1),mcolor(j-inddot*length(mcolor)));
                     l1                                                          =
plot3(FValue(i,1),FValue(i,2),FValue(i,3),str2{1},'markersize',6);
                     text(FValue(i,1),FValue(i,2),FValue(i,3),str1,'FontSize',6);
                     fl{j} = [l1];
                     break
                 else
                     j = j+1;
                 end
             end
             hold on
         end
    else
        warndlg({'Заданное число факторов невозможно отобразить.';'Визуализация результатов
анализа невозможна.'}'','Внимание! ')
    end
end
if (size(FValue,2) == 2) || (size(FValue,2) == 3) || (size(GAData,2) == 3)
         grid on
         flmark = [];
         fllabel = {};
         for i=1:length(fl)
             flmark = [flmark fl{i}];
             fllabel{i} = strcat('Класс № ',num2str(i));
         end
         legend(flmark,fllabel)
         xlabel('F1');
         ylabel('F2');
         zlabel('F3');
end
if (size(dat,2)) > 3 && (isempty(FValue)==1)
    warndlg({'Не проведен факторный анализ.';'Визуализация результатов анализа
невозможна.'}'','Внимание! ')
end

function NoUseExSet_Callback(hObject, eventdata, handles)
flag1 = handles.Mainfig.UserData
set(handles.StartClusterization,'UserData',flag1)

function GArun_Callback(hObject, eventdata, handles)
genfig=open('GeneticModule.fig');
global ghgen
ghgen = guihandles(genfig);

function AutoEncRun_Callback(hObject, eventdata, handles)
Data4AE = getappdata(handles.LoadData,'Data');
autoencfig=open('Autoenc.fig');
ghaefig = guihandles(autoencfig);
set(ghaefig.fAE,'UserData',Data4AE);
set(ghaefig.eNumberNeurons,'String',fix(min(size(Data4AE))/2));

function UseEncodedData_Callback(hObject, eventdata, handles)
set(handles.UseNormalizeData,'Enable','off');
set(handles.FValueTable,'Data',hObject.UserData);
set(handles.FactorName1,'String','Кодированные данные');
```

```
set(handles.CVpartrun,'Label','Разделить кодированные данные')

function CVpartrun_Callback(hObject, eventdata, handles)
cvpartfig = open('CVpart.fig');
ghcvpfig = guihandles(cvpartfig);
switch get(get(handles.GroupForData,'SelectedObject'),'Tag')
    case 'UseOriginal'
        data{1} = getappdata(handles.LoadData,'Data');
    case 'UseFactorsValue'
        data{1} = get(handles.FValueTable,'Data');
    case 'UseEncodedData'
        data{1} = get(handles.FValueTable,'Data');
end
data{2} = getappdata(handles.LoadData,'Target');
set(ghcvpfig.CVpartfig,'UserData',data);

function GroupForData_SelectionChangedFcn(hObject, eventdata, handles)
set(handles.usePartOfTrainData,'Value',0)
set(handles.useAllTrainData,'Value',1)
set(handles.usePartOfTrainData,'Enable','off')
set(handles.usePartOfTestData,'Value',0)
set(handles.useAllTestData,'Value',1)
set(handles.usePartOfTestData,'Enable','off')

function Select_count_factors_Callback(hObject, eventdata, handles)
global ghfanal
global Data4Fac;
d = size(Data4Fac,2);
hObject.Value;
for i=1:d
    for j=1:hObject.Value
        colsiz(i,j) = {''};
    end
end
for i=1:hObject.Value
    istr = num2str(i);
    CN{1,i} = strcat(istr,' фактор');
end
set(ghfanal.Loadings_table,'Data',colsiz,'ColumnName',CN);

function Select_count_factors_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global Data4Fac;
d = size(Data4Fac,2);
m=1;
while ((d+m)<=(d-m)^2)
    mposfact(m,1)=m;
    m=m+1;
end
set(hObject,'String',mposfact);

function Select_rotation_Callback(hObject, eventdata, handles)
selected_rotation = hObject.String{get(hObject,'Value')};
set(hObject,'UserData',selected_rotation);

function Select_rotation_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'UserData','None');

function Start_factor_analysis_Callback(hObject, eventdata, handles)
global Data4Fac
global ghfanal
global ghmform
clc
```

```
m = get(ghfanal.Select_count_factors,'Value');
rotation = get(ghfanal.Select_rotation,'UserData');
global F
rot = '';
rot = rotation;
Data4Fac
[Loadings,specVar,rotation,stats,F] = factoran(Data4Fac,m,'rotate',rotation);
Loadings2 = Chaddok_scale(Loadings);
specVar
stats
set(ghfanal.Loadings_table,'UserData',Loadings);
set(ghfanal.Loadings_table,'Data',Loadings2);
set(ghmform.UseFactorsValue,'Enable','On');
set(ghmform.FValueTable,'Data',F);
set(ghmform.FValueTable,'Visible','on');
set(ghmform.FactorName1,'UserData',rot);
if strcmp(rot,'None') == 1
    set(ghmform.FactorName1,'String',strcat('Фактор. знач. (без вращения)'));
else
    set(ghmform.FactorName1,'String',strcat('Фактор. знач. (',rot,' вращение)'));
end
set(ghmform.usePartOfTrainData,'Value',0)
set(ghmform.useAllTrainData,'Value',1)
set(ghmform.usePartOfTrainData,'Enable','off')
set(ghmform.usePartOfTestData,'Value',0)
set(ghmform.useAllTestData,'Value',1)
set(ghmform.usePartOfTestData,'Enable','off')

function Loadings_table_CreateFcn(hObject, eventdata, handles)
global Data4Fac;
d = size(Data4Fac,2);
m=1;
while ((d+m)<=(d-m)^2)
    mposfact(m,1)=m;
    m=m+1;
end
for i=1:d
    istr = num2str(i);
    RN{1,i} = strcat(istr,' признак');
    colsiz(i,1) = {''};
end
set(hObject,'RowName',RN,'Data',colsiz,'ColumnName','1 фактор');

function Corr_table_CreateFcn(hObject, eventdata, handles)
global Data4Fac;
d = size(Data4Fac,2);
for i=1:d
    istr = num2str(i);
    tname{1,i} = strcat(istr,' признак');
end
CorMat = corrcoef(Data4Fac);
CorMat2 = Chaddok_scale(CorMat);
set(hObject,'RowName',tname,'ColumnName',tname,'Data',CorMat2,'UserData',CorMat);

function figure1_CreateFcn(hObject, eventdata, handles)
movegui('center')

function Start_export_Callback(hObject, eventdata, handles)
global F
global ghfanal
Correlation_matrix = ghfanal.Corr_table.UserData;
Factor_values = F;
Factor_loadings = ghfanal.Loadings_table.UserData;
if iscell(Factor_loadings) == 1
    errordlg('Отсутствуют данные для экспорта','Ошибка');
else
    filename1 = get(ghfanal.Select_rotation,'UserData');
    filename2 = num2str(get(ghfanal.Select_count_factors,'Value'));
    filename = strcat(filename1,'_on_',filename2,'_factors');
    ws = cd;
```

```
    expdir = 'Export Data';
    expsubdir = filename1;
    expws1 = strcat(ws,'\',expdir);
    expws2 = strcat(expws1,'\',expsubdir);
    InfoDir = dir(cd);
    i=1;
    while i<=length(InfoDir)
        namedir1 = InfoDir(i).name;
        i=i+1;
        check1 = strcmp(namedir1,expdir);
        if check1 == 1
            i=length(InfoDir)+1;
            cd(expws1);
            InfoSubDir = dir(cd);
            j=1;
            while j<=length(InfoSubDir)
                namedir2 = InfoSubDir(j).name;
                j=j+1;
                check2 = strcmp(namedir2,expsubdir);
                if check2 == 1
                    cd(expws2);

uisave({'Correlation_matrix','Factor_values','Factor_loadings'},filename);
                    j=length(InfoSubDir)+1;
                    cd(ws);
                end
            end
            if check2 == 0
                mkdir(expsubdir);
                waitfor(msgbox({'Создана директория: ' expws2},'Сообщение','warn','modal'))
                cd(expws2);
                uisave({'Correlation_matrix','Factor_values','Factor_loadings'},filename);
                cd(ws);
            end
        end
    end
    if check1 == 0
        mkdir(expdir);
        waitfor(msgbox({'Создана директория: ' expws1},'Сообщение','warn','modal'))
        cd(expws1);
        InfoSubDir = dir(cd);
        j=1;
        while j<=length(InfoSubDir)
            namedir2 = InfoSubDir(j).name;
            j=j+1;
            check2 = strcmp(namedir2,expsubdir);
            if check2 == 1
                cd(expws2);
                uisave({'Correlation_matrix','Factor_values','Factor_loadings'},filename);
                j=length(InfoSubDir)+1;
                cd(ws);
            end
        end
        if check2 == 0
            mkdir(expsubdir);
            waitfor(msgbox({'Создана директория: ' expws2},'Сообщение','warn','modal'))
            cd(expws2);
            uisave({'Correlation_matrix','Factor_values','Factor_loadings'},filename);
            cd(ws);
        end
    end
end

function varargout = Autoenc(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Autoenc_OpeningFcn, ...
                   'gui_OutputFcn',  @Autoenc_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
```

```
                  'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Autoenc_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = Autoenc_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pbRunAE_Callback(hObject, eventdata, handles)
aehandles = guihandles;
data = aehandles.fAE.UserData;
if size(data,1) > size(data,2)
    data = data';
    set(aehandles.fAE,'UserData',data);
end
EncoderTF = get(aehandles.ppmEncoderTF,'UserData');
DecoderTF = get(aehandles.ppmDecoderTF,'UserData');
TrainingAl = get(aehandles.ppmTrainingAlg,'UserData');
if (strlength(aehandles.eNumberNeurons.String) ~= 0) && ...
        (strlength(aehandles.eMaxIterations.String) ~= 0)
    autoencoder = trainAutoencoder(data,'hiddensize',...
        str2num(aehandles.eNumberNeurons.String),'MaxEpochs',...
        str2num(aehandles.eMaxIterations.String),...
        'EncoderTransferFunction',EncoderTF,...
        'DecoderTransferFunction',DecoderTF,...
        'TrainingAlgorithm',TrainingAl);
    encoded_data = encode(autoencoder,data);
    global ghmform
    set(ghmform.FValueTable,'Data',encoded_data');
    set(ghmform.UseEncodedData,'UserData',encoded_data');
    set(ghmform.FValueTable,'Visible','on');
    set(ghmform.FactorName1,'String','Кодированные данные');
    set(ghmform.UseEncodedData,'Enable','On');
else
    errordlg('Введены не все данные','Ошибка ввода данных');
end
set(ghmform.usePartOfTrainData,'Value',0)
set(ghmform.useAllTrainData,'Value',1)
set(ghmform.usePartOfTrainData,'Enable','off')
set(ghmform.usePartOfTestData,'Value',0)
set(ghmform.useAllTestData,'Value',1)
set(ghmform.usePartOfTestData,'Enable','off')

function ppmEncoderTF_Callback(hObject, eventdata, handles)
set(hObject,'UserData',hObject.String{get(hObject,'Value')});

function ppmEncoderTF_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'UserData',hObject.String{get(hObject,'Value')});

function ppmDecoderTF_Callback(hObject, eventdata, handles)
set(hObject,'UserData',hObject.String{get(hObject,'Value')});

function ppmDecoderTF_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
set(hObject,'UserData',hObject.String{get(hObject,'Value')});

function ppmTrainingAlg_Callback(hObject, eventdata, handles)
set(hObject,'UserData',hObject.String{get(hObject,'Value')});

function ppmTrainingAlg_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'UserData',hObject.String{get(hObject,'Value')});

function eNumberNeurons_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function eMaxIterations_CreateFcn(hObject, eventdata, handles)
if               ispc               &&               isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function eNumberNeurons_KeyPressFcn(hObject, eventdata, handles)
aehandles = guihandles;
check_user_input(aehandles.eNumberNeurons,eventdata.Key)

function eMaxIterations_KeyPressFcn(hObject, eventdata, handles)
aehandles = guihandles;
check_user_input(aehandles.eMaxIterations,eventdata.Key)

function varargout = GeneticModule(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GeneticModule_OpeningFcn, ...
                   'gui_OutputFcn',  @GeneticModule_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function GeneticModule_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = GeneticModule_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
global ghgen
e5=get(ghgen.edit5,'String');
e5=str2num(e5);
if ~isempty(e5) && ~isempty(get(ghgen.edit1,'String')) &&...
        ~isempty(get(ghgen.edit2,'String')) && ~isempty(get(ghgen.edit4,'String'))...
        &&                      ~isempty(get(ghgen.edit6,'String'))                    &&
~isempty(get(ghgen.pushbutton2,'UserData'))
    global ghmform
    data=get(ghmform.DataTable,'Data');
    k=get(ghgen.edit6,'String');
    save data
    fun = eval(['@' get(ghgen.pushbutton2,'UserData')]);
```

```
    options                                                                                    =
gaoptimset('PopulationSize',str2num(get(ghgen.edit1,'String')),'Generations',str2num(get(gh
gen.edit2,'String')),...

'StallGenLimit',str2num(get(ghgen.edit2,'String'))/4,'TimeLimit',str2num(get(ghgen.edit4,'S
tring')),...

'StallTimeLimit',str2num(get(ghgen.edit5,'String')),'PlotFcns',{@gaplotbestf,@gaplotbestind
iv});

    [x F]=ga(fun,size(data,2)*k,options)

function pushbutton2_Callback(hObject, eventdata, handles)
current_dir = cd;
current_dir = strcat(current_dir,'\');
[file,path] = uigetfile('*.m');
if ~isequal(file,0) && isequal(current_dir,path)
   fun = erase(file,".m");
   set(hObject,'UserData',fun)
   global ghgen
   set(ghgen.text10,'String',strcat('Выбрана функция: ',fun))
   set(ghgen.text10,'ForegroundColor','black')
end

function varargout = Results_on_table(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Results_on_table_OpeningFcn, ...
                   'gui_OutputFcn',  @Results_on_table_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Results_on_table_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject
guidata(hObject, handles);

function varargout = Results_on_table_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function figure1_CreateFcn(hObject, eventdata, handles)
movegui('center')

function uitable1_CreateFcn(hObject, eventdata, handles)
global gp1;
global t
RCl = gp1;
RCl= Class_valuation1(RCl,t);
set(hObject,'ColumnName',{'Object №','Cluster №'},'RowName','');
set(hObject,'Data',RCl);

function uitable2_CreateFcn(hObject, eventdata, handles)
global erro
global misss
global miss2cl
global ghmform
ClusterCount = ghmform.ClusterCountData.UserData;
RN = {'Total'};
for i=2:ClusterCount+1
    ii = i-1;
    ii = num2str(ii);
    RN{1,i} = strcat(ii,' cluster');
```

```
end
set(hObject,'ColumnName',{'Count errors','% errors'},'RowName',RN);
ut = hObject;
d(1,1) = misss;
d(1,2) = erro;
for i=1:ClusterCount
    d(i+1,1) = miss2cl(i,1);
    d(i+1,2) = miss2cl(i,2);
end
d = Class_valuation2(d);
ut.Data = d;

function SaveXLSOut_Callback(hObject, eventdata, handles)
Multisave('XLS','NWOUT',selected_net,andatatype,gp1,nowrot,nowf)

function SaveXLSPercentTable_Callback(hObject, eventdata, handles)
Multisave('XLS','PercTable',selected_net,andatatype,{misss erro miss2cl},nowrot,nowf)

function SaveXLSAll_Callback(hObject, eventdata, handles)
Multisave('XLS','All',selected_net,andatatype,{misss erro miss2cl gp1},nowrot,nowf)

function SaveMATOut_Callback(hObject, eventdata, handles)
Multisave('MAT','NWOUT',selected_net,andatatype,gp1,nowrot,nowf)

function SaveMATPercentTable_Callback(hObject, eventdata, handles)
Multisave('MAT','PercTable',selected_net,andatatype,{misss erro miss2cl},nowrot,nowf)

function SaveMATAll_Callback(hObject, eventdata, handles)
global selected_net
global andatatype
global nowf
global erro
global misss
global miss2cl
global gp1
global nowrot
nowrot
Multisave('MAT','All',selected_net,andatatype,{misss erro miss2cl gp1},nowrot,nowf)

function PerfomTable_CreateFcn(hObject, eventdata, handles)
global train_time
global selected_net
global ghmform
ClusterCount = ghmform.ClusterCountData.UserData;
global IterCount
if (strcmp(selected_net,'Слой_Кохонена') == 1) || (strcmp(selected_net,'Карта_Кохонена') ==
1)
    colname3 = 'Кол-во кластеров';
else
    colname3 = 'Кол-во классов';
end
set(hObject,'RowName',{'Кол-во итераций на обучение сети',colname3,...
    'Время на обучение сети, с'},'ColumnName',selected_net);
hObject.Data = [IterCount; ClusterCount; train_time];
```

ПРИЛОЖЕНИЕ Б

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2017617294

ПРОГРАММНЫЙ КОМПЛЕКС «АНАЛИЗ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ»

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Иркутский национальный исследовательский технический университет» (ФГБОУ ВО «ИРНИТУ») (RU)***

Авторы: ***Харахинов Владимир Александрович (RU), Сосинская Софья Соломоновна (RU)***

Заявка № **2017614163**

Дата поступления **04 мая 2017 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***04 июля 2017 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

ПРИЛОЖЕНИЕ В

Утверждаю
Директор института
высоких технологий
ИРНИТУ
_____Е. А. Анциферов
«07» июня 2019 г.

АКТ

О внедрении программного продукта «Анализ экспериментальных данных на основе нейронных сетей» (автор Харахинов Владимир Александрович)

Комиссия в составе: С. В. Бахвалов, к.т.н, доцент, зав. Кафедрой автоматизированных систем ИРНИТУ (председатель),Е.А. Осипова, к.т.н., доцент кафедры автоматизированных систем, С.С. Сосинская, к.т.н.,профессор кафедры вычислительной техники составила настоящий акт о том, что программный продукт «Анализ экспериментальных данных на основе нейронных сетей», автором которого является Харахинов Владимир Александрович, внедрен в учебный процесс и использовался на кафедре автоматизированных систем в учебном курсе «Технологии обработки информации», а также при подготовке выпускных квалификационных работ.

Использование созданной Харахиновым Владимиром Александровичем программы для ЭВМ «Анализ экспериментальных данных на основе нейронных сетей» (свидетельство о государственной регистрации №2017617294 от 04.07.2017) позволяет решать задачи классификации и кластеризации данных с использованием искусственных нейронных сетей различных типов. Программный продукт поддерживает процесс редукции анализируемых экспериментальных данных. Реализована возможность детальной настройки нейронной сети, в том числе при помощи генетического алгоритма. Результат работы программы представляет собой сводную таблицу, содержащую детальную информацию о проведенном анализе. Разработанное приложение помогает студентам в изучении различных алгоритмов и методов анализа данных, изучаемых в курсе «Технологии обработки информации».

Председатель комиссии С. В. Бахвалов

Члены комиссии Е.А. Осипова

С.С. Сосинская

ПРИЛОЖЕНИЕ Г

ООО «Центр транспортных технологий»
Адрес (почтовый/фактический): 664074, г. Иркутск, ул. Лермонтова 81/6-25
ИНН/КПП: 3812156617 / 381201001
E-mail: transport@istu.edu
URL: transport.istu.edu
Тел.: +79148805378

Исх.№ 4/001 от 24.08.2021 г.

АКТ

об использовании результатов диссертационной работы «Оценка и прогнозирование дорожной обстановки при наличии ДТП на основе нейросетевых и эвристических технологий», представленной Харахиновым Владимиром Александровичем на соискание ученой степени кандидата технических наук по специальности: 05.13.01 – системный анализ, управление и обработка информации.

Настоящим актом удостоверяется, что результаты диссертационной работы «Оценка и прогнозирование дорожной обстановки при наличии ДТП на основе нейросетевых и эвристических технологий» были использованы компанией ООО «Центр транспортных технологий» в научно-исследовательской и проектной деятельности в процессе моделирования транспортных потоков города Иркутска.

Генеральный директор
ООО «Центр транспортных
Технологий» А.Г. Левашев

ПРИЛОЖЕНИЕ Д

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
НАУЧНО-ПРОИЗВОДСТВЕННОЕ ОБЪЕДИНЕНИЕ
СЕЛЕКЦИОННО СЕМЕНОВОДЧЕСКИЙ ЦЕНТР «АНГАРА»
(ООО НПО ССЦ «АНГАРА»

ИНН/КПП: 3827035485/382701001
Адрес: 664038, Иркутская область, Иркутский район, п. Молодежный,
ул. Молодежная, дом 1/1, помещение 208. Тел.: +79526123358

АКТ

об использовании результатов научной диссертационной работы
Харахинова Владимира Александровича

Настоящим актом удостоверяется, что результаты диссертационной работы представленной Харахиновым Владимиром Александровичем на соискание ученой степени кандидата технических наук по специальности: 2.3.1 – системный анализ, управление и обработка информации были использованы ООО НПО ССЦ «АНГАРА» в научно-исследовательской и производственной деятельности.

Разработанная методика повышения качества кластерного анализа использована для анализа массивов данных по урожайности сельскохозяйственных культур в различных районах Иркутской области.

Использование полученных результатов научной работы позволяет ускорить процесс выявления территорий с низкими показателями урожайности.

Директор ООО НПО ССЦ «АНГАРА» — Футорный А.В

подпись — Фамилия И.О.