ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ТВЕРСКОЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Храмов Игорь Сергеевич

ГЕОИНФОРМАЦИОННЫЕ МОДЕЛИ И МЕТОДЫ ПРЕДСТАВЛЕНИЯ И ОЦЕНКИ ОБСТАНОВКИ В БЛИЖНЕЙ МОРСКОЙ ЗОНЕ С ИСПОЛЬЗОВАНИЕМ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Специальность 25.00.35 - Геоинформатика

Диссертация на соискание учёной степени кандидата технических наук

Научный руководитель:
доктор технических наук,
профессор, Биденко С.И.

Санкт-Петербург - 2020

Оглавление

Введение

Актуальность. Ближняя морская зона (БМЗ) характеризуется высокой интенсивностью территориальной активности (грузовые и пассажирские перевозки, добыча углеводородов и полезных ископаемых, исследования, оборонная деятельность), множеством навигационных опасностей (сложный рельефом дна, малые глубины, лед, течения, влияние суши), изменчивостью гидрометеорологических условий.

Хозяйственная и иные виды деятельности в прибрежной акватории оказывают значительное влияние на экологическое состояние региона.

Обстановка в БМЗ меняется достаточно быстро и требует постоянной оценки для обеспечения безопасности хозяйственной деятельности и экологической ситуации.

Традиционно для отображения и анализа территориальной ситуации используются различные геоинформационные средства. Особенностью профессиональных ГИС является их ориентация на широкий круг различных пользователей. В связи с этим узкие приложения требуют создание дополнительных программных оболочек ГИС для решения конкретных задая территориального анализа.

В связи с тем, что обстановка в БМЗ содержит большое количество разнородных объектов и явлений, является высоко динамичной, представляется целесообразным рассмотреть возможность использования в ГИС-анализе модельно-методического аппарата искусственных нейронных сетей (ИНС), так как ИНС содержат значительный аналитический потенциал по классификации и оценке больших массивов высоко динамических данных.

Использование аппарата ИНС в пространственном анализе призвано устранить такие недостатки традиционных алгоритмов оценки обстановки, как:

При анализе текущего состояния методов оценки обстановки в морской зоне выявлены следующие проблемы оценке обстановки, как: эффективный учет большого количества разнородных факторов обстановки; обеспечение высокой скорости обработки пространственной информации; возможность быстрого обучения и перенастройки алгоритмов анализа (человеческий фактор).

Следовательно, актуальной является задача внедрения моделей и методов ИНС в пространственный ГИС-анализ, в том числе и в процедуры оценки обстановки и выработки рекомендаций в БМЗ.

Объект исследований: обстановка в ближней морской зоне.

Предмет исследований: модели и методы представления и анализа геопространственной ситуации в ближней морской зоне.

Цель работы: Создание методики оценки обстановки и выработки рекомендаций в ближней морской зоне на основе искусственных нейронных сетей.

Научные задачи:

- Проанализировать предметную области оценки территориальной обстановки, модели и методы ИНС в аспекте процедур геопространственного регионального анализа.
- Создать и описать модель обстановки в ближней морской зоне, оптимальной для обозначенной цели
- Разработать математическую модель для последующей реализации методики оценки обстановки в ближней морской зоне
- Произвести отбор и последующее сравнение нескольких реализаций архитектур ИНС для выявления оптимальной для обозначенной цели
- Провести анализ эффективности предложенной методики
- Предложить варианты практического применения методики в алгоритмах построения пути судна в ближней морской зоне

Научные результаты, выносимые на защиту:

1. Топологическая модель представления обстановки в ближней морской зоне, основанная на анаморфировании и оптимизированная для работы с искусственными нейронными сетями.

2. Методика оценки обстановки в ближней морской зоне, основанная на работе искусственных нейронных сетей и анаморфированном представлении территориальной обстановки.

3. Методика построения оптимального маршрута перехода на основании оценки обстановки в ближней морской зоне, реализованная с применением каскада настраиваемых искусственных нейронных сетей.

Новизна научных результатов

1. Модель геосреды (обстановки) отличается топологическим переходом от географически конкретного представления территориальной ситуации к пространственно-абстарктной анаморфозе (картоиду), что позволяет формировать наборы исходных геоданных, применимых для работы (обучения) ИНС.

2. Методика оценки обстановки в ближней морской зоне отличается:

- применением специально спроектированных и обученных на оригинально сформированных априорных наборах геоданных ИНС, что позволяет повысить быстродействие процедур анализа и снизить нагрузку на аппаратные ресурсы;

- топологизацией результатов территориальных оценок, что позволяет более наглядно отображать проблемные зоны геосреды и упрощать процессы

оптимизации решений на конкретной геоситуации (за счёт снижения размерности пространства обстановки);

3. Методика построения оптимального маршрута перехода в БМЗ отличается наличием дополнительных процедур топологизации для поиска вариантов решений в пространственно абстрактной среде и детопологизации первичного решения для адаптации его в географически конкретной обстановке с применением аппарата ИНС, что позволяет наглядно отображать опасные зоны, избегать потери общей обстановки в регионе при переходе к более крупным масштабам геоизображений районов, а также обеспечивает непрерывный контроль оператором процессов преобразования геоинформации при оценке территориальной обстановки и выработки рекомендаций.

Соответствие диссертации паспорту специальности. Полученные научные результаты соответствуют пунктам 3, 6, 7, 8, 9 паспорта специальности 25.00.35 – «Геоинформатика».

Ценность полученных результатов

Теоретическая значимость полученных результатов состоит в разработке модели представления обстановки в ближней морской зоне, оптимизированной как для работы с аппаратом ИНС, так и для визуального представления. Кроме того, была создана и апробирована новая математическая модель НС, оптимизированная для решения поставленной задачи.

Практическая ценность полученных результатов заключается в том, что предложенные методики показывают прирост быстродействия при обработке больших массивов входных данных в сравнении с традиционными алгоритмами за счет обученных ИНС, а также нивелируют воздействие субъективных факторов при оценке обстановки в ближней морской зоне и построении безопасных маршрутов

Публикации. Основные научные результаты диссертации опубликованы в 21 статье в научно-технических изданиях, в том числе 12 из рекомендованного ВАК перечня. Получено авторское свидетельство № 2018665037 о регистрации программы для ЭВМ (2018).

Апробация. Основное содержание диссертации опубликовано в научных журналах РИНЦ, в том числе в пяти изданиях, рекомендованных ВАК РФ. Результаты работы были доложены на конференциях: «Актуальные направления научных исследований XXI века: теория и практика» (г. Воронеж) в 2015 и 2017 гг., Образование в XXI веке, 2017 г. (г. Тверь) Научно-практическая конференция «Современные проблемы гидрометеорологии и устойчивого развития Российской Федерации», (Российский государственный гидрометеорологический университет, Санкт-Петербург, 2019).

Внедрение. Полученные в диссертации научные результаты внедрены в учебный процесс ФГБОУ ВО «Тверской Государственный университет», ФГБОУ ВО «Государственный морской университет имени адмирала Ф.Ф. Ушакова» и ФГБОУ ВО ««Российский государственный гидрометеорологический университет», а также НИР «Грифон-8-ТвГУ».

Структура и объём диссертации. Диссертационная работа состоит из введения, четырёх глав, заключения, двух приложений, объем 139 листов, включая 42 рисунков, 5 таблиц, в списке литературы 117 наименований.

Глава 1. Анализ предметной области предоставления и отображения обстановки ГИС-средствами и постановка задач исследования

1.1 Анализ данных и требований к представлению, отображению и оценке территориальной ситуации в ближней морской зоне

В соответствии с источниками [47, 77] ближняя морская зона (БМЗ) – это прилегающая к береговой линии акватория шириной около 200 морских миль (примерно 400 км).

В широко смысле слова территориальная ситуация (обстановка) – это совокупность природных, социальных и технических объектов и явлений, расположенных, протекающих и функционирующих в конкретном географическом районе.

Применительно к ситуации в БМЗ это водная среда, корабли и суда, навигационные опасности (отмели, острова, подъёмы морского дна (малые глубины), затонувшие суда) элементы морской хозяйственной активности (нефтяные и газовые вышки, оборудование для добычи донных полезных ископаемых), навигационное оборудование района, системы связи, береговая инфраструктура, лед, течения, ветер, трубопроводы и др. [35]

Обстановка в БМЗ характеризуется следующими территориальными особенностями:

- широкий пространственный охват;
- высокая интенсивность судоходства (грузовые и пассажирские перевозки) на подходах к портам;
- наличие многочисленных навигационных опасностей (малые глубины, сложный рельеф дна, лед, течения);
- высокая экономическая активность (добыча углеводородов, полезных ископаемых);
- наличие исследовательской и рекреационной активности
- влияние суши на изменчивость гидрометеорологического режима в регионе;
- наличие различных ограничений (санитарные, таможенные, контрольные, пограничные и др. зоны);
- навигационные разграничения и др (см. рисунок 1).

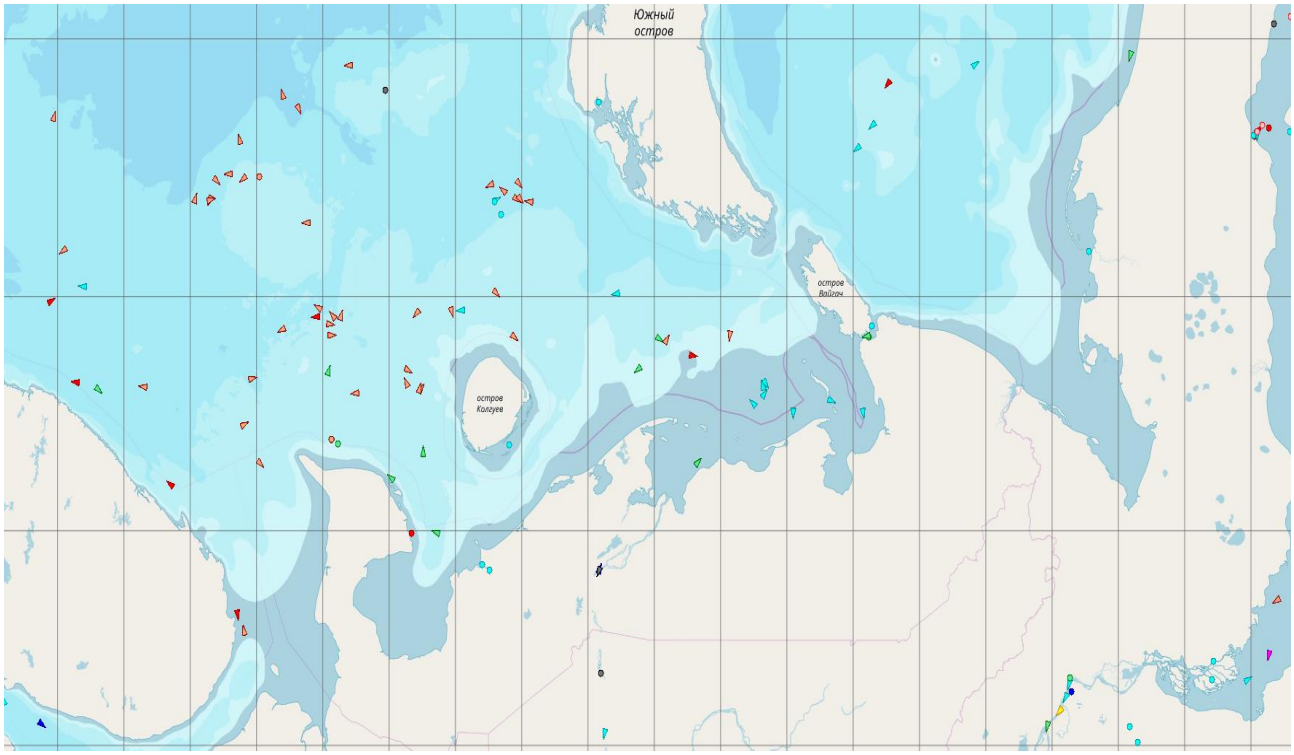

Рис. 1. Карта обстановки в ближней морской зоне. (Цветными треугольниками выделены суда, цветными кругами – стационарные объекты антропогненного происхождения, глубины выделены различными цветами)

Данные об обстановке в БМЗ характеризуются:

- большими объемами ситуационной информации;
- географичностью – координатной привязкой к поверхности Земли;
- высокой изменчивостью;
- предметно-содержательной разнородностью (физико-географические условия, хозяйственно-географические условия, навигационная информация, правовая информация и др.);
- типовой разнородностью:

по носителям (бумажные, электронные),

по источникам (карты и книги (лоции, описания, пособия, таблицы, рекомендации), от средств наблюдения (локация, средства определения места, лаг,) данные дистанционного зондирования (ДДЗ), данные от систем оповещения и связи),

по форме представления (текст, графика),

по геопространственности (геоинформация (имеет привязку к поверхности Земли), обычная информация),

по моменту получения (априорная, оперативная),

по масштабам (глобальная, региональная, локальная),

по точности,

по актуальности и т.д.).

Традиционно обстановка в акваториях, районах морской территориальной активности (МТА) отображается, ведется и анализируется с помощью карт [74, 82]. Карта – это математически определенное, уменьшенное, генерализованное отображение земной (или иной поверхности) или геопространства, показывающее посредством условных знаков размещение и свойства картируемых объектов [69, 75]. То есть это – модель земной поверхности, характеризующаяся: математической основой (проекция, масштаб, сетка); отбором отображаемых объектов и явлений; легендой (правилами) изображения реальной действительности.

В качестве модели карта служит для анализа изображенных на ней процессов и явлений, их связей и отношений, их пространственно-временной динамики [80].

С точки зрения процедур управления МТА (оценка обстановки, выработка рекомендаций, реализация решения) морская карта характеризуется следующими полезными свойствами [32,33]:

1) пространственно-временное подобие (геометрическое, временное, реляционное (связи, отношения)) объекта обстановки и его картографического изображения;

2) содержательное соответствие, обуславливающее передачу главных сущностных особенностей объектов обстановки с учетом генезиса, внутренней и внешней структуры, иерархии картируемых объектов местности (геообъектов);

3) наглядность и обзорность – оперативная одномоментная передача (параллельный код) расположения всех отображаемых на карте объектов, их пространственных и содержательных характеристик, связей и отношений;

4) высокая информативность – наряду с размещением большого количества знаков и характеристик, дополнительная информация содержится во взаимном расположении и соподчинении объектов, процедурах математико-картографического моделирования, анализа и синтеза территориальной обстановки.

5) метричность – возможность выполнять прямые измерения количественных (картометрических) показателей и определять качественные характеристики объектов и отношений реального мира (с учетом особенностей математической основы (проекция, масштаб) карты);

6) синтетичность – возможность получать отсутствующие на первичной карте реальности параметры и характеристики географической действительности;

7) генерализованность карты, подразумевающая: переход от индивидуальных понятий к классификационно-обобщающим; установление и отображение типичных (с точки зрения существа решаемой с помощью карты

задачи, ее назначения (типа)) характеристик объектов; устранение второстепенных деталей, что сообщает карте свойство пространственно-содержательной абстрактности (в противоположность географической конкретности);

7) избирательность и обобщенность – способность, с одной стороны, вычленять и раздельно представлять те факторы и аспекты геообъекта, которые в реальной действительности проявляются совместно (форма без содержания, содержание без формы), а, с другой стороны, давать целостное изображение явлений и процессов, которые в реальных условиях проявляются изолированно (тип климата, тип гидрологии, режим судоходства и др.);

8) универсальность – при составлении карты ориентируются на удовлетворение потребностей различных пользователей;

9) непрерывность – картографическое изображение присутствует во всех точках карты, на ней нет пустот и разрывов (за исключением разрывов, обусловленных некоторыми картографическими проекциями);

10). однозначность – свойство иметь единственное значение картируемого параметра в каждой точке геопространства в пределах принятой системы условных обозначений.

С точки зрения управления территориальными объектами и системами, картам свойственны следующие характерные сущностные недостатки [20,32]:

1. Субъективность карты, обусловленная предпочтениями картографа при выборе математической основы, составлении карты, выполнении процедур генерализации. То есть, невозможность участия пользователя (судоводителя) в формировании требуемого для конкретных задач картографического изображения (КИз).

2. Отсутствие возможности использовать априорные первичные данные полевых наблюдений, данных дистанционного зондирования. Это снижает степень доверия пользователя к отображенным на карте объектам и явлениям.

3. Низкая оперативность и высокая трудоемкость выполнения измерений по карте, выполнения преобразований КИз (проекции, масштаба, нарезки, легенды и др.), нанесения данных обстановки, совместной обработки априорных (карта) и оперативных (от системы наблюдения) данных, графической и описательной информации.

4. Статичность КИз, показ только одного временного среза, невозможность передачи системной динамики обстановки и «проигрывания» вариантов действий, низкие мультимедийные качества.

5. Отсутствие возможности анализа взаимосвязей между различными феноменами, если они не отображены на карте.

6. Потеря общей ситуации в районе МТА в целом при необходимости перехода к крупным масштабам КИз. За детализацию (увеличение масштаба КИз) обстановки в конкретном подрайоне платим потерей отображения общей ситуации в регионе.

7. Устаревание КИз, необходимость трудоемкой ручной корректуры крты.

8. Низкая контроллинговая активность [4, 5, 13], возможность использования КИз в системах территориального управления только на уровне карты- подложки.

В аспекте снижения размерности пространства признаков МТА, применения в территориальном анализе и управлении методов теории графов, искусственных нейронных сетей карта имеет еще один недостаток: сохраняет неравномерность передачи пространственного распространения картируемых содержательных параметров (плотности, напряженности, потенциалы геофизических, географических, морфометрических, социальных, экономических, логистических и др. характеристик (полей)). КИз не обеспечивает изотропность (равномерность) распределения картируемого параметра в пространстве,

Для обеспечения изотропности представления и отображения анализируемого параметра геоситуации могут быть применены процедуры топологизации или анаморфирования [9, 10, 11, 14, 17] картографических изображений реальной географической действительности. В этом случае географически конкретное изображение территориальной обстановки (карта) перейдет в абстрактное (с точки зрения географической конкретики) изображение местности (картоид, анаморфоза) [7, 9, 30]. Но при этом будет обеспечено свойство изотропности картируемых параметров.

Еще одно полезное свойство топологизации (анаморфирования) реальной тактической геоситуации – снижение размерности пространства решений [33, 19], что, в свою очередь, повышает оперативность выработки вариантов решения.

Важным аспектом информационно-аналитических систем является дружественность интерфейса доведения информации об обстановке оператору (лицу, принимающему решение) для адекватного восприятия, оперативого анализа и принятия решения для широкого спектра практических задач через графическое представление информации.

Эта информация должна всесторонне учитывать различные факторы и характеристики исследуемого процесса или явления, а для того, чтобы ее восприятие и обработка привели к резкому сокращению времени ее анализа и, соответственно, времени принятия обоснованного варианта решения, носить в основном, визуальный характер – в виде визуальных информационных образов.

Исходя из требований к аппаратно-программным средствам представления и отображения данных обстановки [55], они должны обеспечивать:

1. Адекватное представление больших объемов разнородной и разнотипной быстроменяющейся информации об обстановке.

2. Наглядное непрерывное отображение больших объемов разнородной и разнотипной быстроменяющейся информации об обстановке.

3. Оперативное преобразование ГИ об обстановке – масштаба, нарезки, проекции, легенды, нагрузки.

4. Отображение системной динамики геоситуации.

5. Автоматическая актуализация данных обстановки.

6. Реализация адаптивной информационной модели обстановки.

7. Обеспечивать сохранение общей ситуации в районе при локальном увеличении масштаба геоизображения подрайона карты обстановки.

Оценка обстановки (ОО) – это формально-логико-эвристическая процедура получения количественных и качественных характеристик факторов хозяйственной активности, а также физико-географических и социально-географических условий района плавания (БМЗ) с точки зрения решаемой (судном, кораблем, объектом хоз. деятельности) задачи [68]. А именно: в какой мере факторы обстановки способствуют или препятствуют решению поставленной перед кораблем (судном) задачи.

В состав ОО входят:

1) оценка своих объектов (корабли, суда, объекты хозяйственной деятельности);

2) оценка противодействующих или ограничивающих факторов (терроризм, пиратство, эпидемии, карантины, экологические угрозы и т.д.);

3) оценка самого района плавания, а именно оценка физико-географических (ФГУ) условий района плавания (рельеф дна, глубины, ветер, течения, лед, видимость), а также оценка инфраструктурно-географических условий (ИСГУ) района (навигационно-гидрографическая оборудованность, причальная инфраструктура, система связи и оповещения, система спасения).

Содержание ОО включает выполнение частных оценок обстановки и получение общей (обобщенной) оценки. Частные оценки определяют территориальное расположение и содержательные характеристики опасных явлений (лед, мелководье, течения, социальные угрозы). Общая оценка призвана определить пространственную (территориальную) зону, где судно безопасно может решать поставленную задачи, а также обобщенную количественную или качественную оценку успешности решения судном поставленной задачи.

Процедуры оценки обстановки характеризуются:

1. Как формальным, так и логико-эвристическим характером обработки информации в процессе оценки данных обстановки. Часть алгоритмов могут выполняться автоматически, часть – человеком-оператором или лицом, принимающим решение.

2. Требованием оперативной обработки больших массивов разнородной информации об остановке (лед, течения, ветер, видимость, навигационные опасности и др.).

3. Необходимостью оперативной адаптации (обучения) алгоритмов ОО.

4. Потребностью доступа к первичной полевой информации (данные наблюдений).

5. Требованием совместного анализа разнородной ГИ об обстановке.

6.Учетом взаимного влияния макро- и микро-уровней представления геоситуации.

7. Необходимостью реализации автоматической картометрии и др.

1.2 Анализ ГИС-средств моделирования территориальной ситуации

1.2.1 Обзор особенностей современных отечественных и зарубежных ГИС

Анализ современных ГИС-средств и продуктов [32] показывает следующее. Разработки ГИС-проектов последних лет отличаются явно выраженной тенденцией использования различных ГИС-продуктов на платформах персональных компьютеров с ориентацией на подключение их в локальные и глобальные информационные сети. Телекоммуникационные возможности, очевидно, существенно усиливают мощность и гибкость разрабатываемых ГИС-проектов, объединяя возможности технических и программных средств разных систем и наращивая информационное обеспечение посредством объединения распределенных баз данных, функционирующих на различных вычислительных платформах.

Вычислительные платформы, на которых функционируют современные программные средства ГИС-технологий разделяются (в довольно условных границах) на большие ЭВМ (Main Frame), минимикрокомпьютеры (МС – PDP,

VAX, Apo11o), рабочие (WS – Hewlet-Packerd, Sun SPARCStation) и многочисленное семейство персональных компьютеров ПК (PC - Intel и Macintosh).

Последние находят все большее и широкое применение в области обработки больших и сверх больших объемов текстовой и пространственной информации, в том числе и в ГИСпроектах. Это объясняется тем, что рынок аппаратных средств для инструментальной поддержки информационных систем на базе ГИС технологий все более расширяется. Развитие средств вычислительной техники привело к ситуации, когда мощные многопроцессорные персональные компьютеры на базе SISCпроцессоров Intel Pentium и их аналогов стали доступными по цене массовому потребителю, а по своим технологическим, возможностям «подпирают» снизу рабочие станции на RISC процессорах, но по совокупным накладным расходам значительно выигрывают перед последними. Это обстоятельство, по существу, стирает грань, разделявшую до недавнего времени эти два класса вычислительных платформ и обеспечило «прорыв» ГИС-технологиям на массовый потребительский рынок.

Можно выделить ряд основных технических требований к вычислительным платформам, работающим со сверхбольшими БД, каковыми являются БД геоинформационных систем. Требования непосредственно к компьютерам определяются исходя из: размеров оперативной (RAM) и дисковой (HDD) памяти; числа и общего быстродействия центрального процессора(ов) (CPU); возможности гибкой настройки комплекса на конкретную задачу за счет расширения конфигурации системы; характеристик видеосистемы (прежде всего видеоадаптера), тапа, размеров, разрешения экранов мониторов и их числа, подсоединенных к компьютеру; набора подключаемых периферийных устройств сканеров, дигитайзеров, плоттеров, принтеров, модемов и т. д.; эргономичности, надежности и безопасности для операторов и пользователей.

Практически все эти требования достижимы современными ПК на базе процессоров Intel Pentium и их аналогов с частотой достигающей сегодня 2,5 – 3

ГГц, имеющих оперативную память от 128 до 1500 и более мегабайт (Мб), а дисковую память до нескольких сотен гигабайт.

Уже остались в прошлом операционная система MS DOS. Основное ее достоинство небольшой объем занимаемой ее ядром в оперативной памяти, достаточно открытая архитектура. Главным же недостатком этой операционной системы (и ее аналогов PCDOS, DRDOS) было небольшое доступное системе адресное пространство в оперативной памяти (не более 640КЬ) и невозможность поддержки многозадачных режимов. Тем не менее, в этих операционных средах нашли реализацию многие версии ГИС-пакетов, где и сегодня решаются разнообразные задачи комплексной обработки пространственной информации. [14].

В настоящее время на платформах PC используется в подавляющем большинстве случаев 64х разрядные операционные системы семейства WINDOWS, которые, обеспечивают вытесняющую многозадачность, многопотоковость и, что очень важно, развитую поддержку сетевых функций, что раньше было прерогативой ОС OS/2 и UNIX.

Не надо забывать, что геоинформационные технологии производственного масштаба ориентированы на работу в сетевой среде с размещением данных и прикладного ПО на серверах различного назначения – файловых, принтсерверах, серверах БД и пр. И здесь основной платформой для серверов различного назначения все чаще стали использоваться INTEL платформы в серверной конфигурации. [14,22,45,57].

На платформах UNIX до сих пор применяются "тяжелые" полнофункциональные инструментальные ГИС, использующие как свою внутреннюю СУБД, так и большие корпоративные внешние СУБД (ORACLE, DB2, Informix и др.), которые в свою очередь видят эти ГИС в качестве своего клиента. На российском рынке в этой "весовой категории" нашли свое применение такие ГИС как SPANS GIS (Intera Tydec, Canada), ARC/INFO (ESRI, USA) и ряд других. [45].

Говоря о ГИС, обычно следует уточнять, о чем именно идет речь.

Информационные технологии (в общем случае) - это совокупность информационных процедур и операций, выполняемых в определенной последовательности и определяющих процесс преобразования некоторой исходной информации в информационный продукт.

Инструментальные ГИС — это программные средства общего назначения, выполняющие различные сервисные функции для создания пользовательских ГИС-приложений.

ГИС-технологии — это информационные технологии, обеспечивающие организацию и преобразование пространственноопределенных данных с использованием инструментальных ГИС и автоматизированных средств получения и обработки данных.

Прикладные ГИС (проблемноориентированные ГИС, ГИС-приложения) это средства решения конкретных задач, определяемых требованиями проблемной области.

Методика создания прикладных ГИС проблемно зависима, т.е. на ее положения оказывает существенное влияние специфика проблемной области.

Требования к ГИС необходимо разделить на требования к ГИС вообще и требований к ГИС в возможности решать целевые задачи.

Еще один вопрос, интересный для обсуждения в части общих тенденций развития рынка информационных систем в навигации, это сближение существующих программных продуктов судовождения по обеспечению навигационной информацией с интегрированными информационными навигационными системами и полноценными ГИС.

Осознав возникшую потребность, многие производители электронной аппаратуры (прежде всего систем радиолокационной прокладки САРП и систем управления движением судов СУДС) приступили к созданию собственных электронных картографических систем. Первые системы использовали растровые электронные карты сканированные копии бумажных источников. Основными недостатками растровых карт являются: невозможность автоматического контроля безопасности и существенные ограничения по

настройке отображения картографической информации на экране компьютера (презентации электронной карты).

Вместе с тем, необходимость использования электронных карт и соответствующих навигационных систем, основанных на единых стандартах, становилась все более очевидной. В 1987 году была учреждена Гармонизационная группа ИМО/МГО (IMO/IHO), которая занялась разработкой единого эксплуатационного стандарта на систему отображения электронных карт и информации (ECDIS Electronic Chart Display and Information System), основывающегося на использовании векторных электронных карт.

Электроннокартографическая система может быть признана ECDIS системой, только если входящие в ее состав программное обеспечение и аппаратура, а также используемые электронные карты соответствуют всем ниже перечисленным требованиям:

IHO Special Publication S57 стандарт МГО (Международного Гидрографического Общества) на обмен гидрографическими данными. Описывает структуру данных и формат обмена между Гидрографическими Службами, производителями ECDIS, судовладельцами. В настоящий момент осуществляется переход от версии 2 документа (DX90) к редакции 3 (S57 edition 3 от 3 ноября 1996 г.)

IHO Special Publication S52 спецификация на информацию, содержащуюся в электронных картах, а также перечисление требований по отображению данных. Редакция 5, декабрь 1996 г.

IMO Resolution A/817 рабочий стандарт ИМО на ECDIS системы, перечисляет требования к программному и аппаратному обеспечению, электронным картам и цифровым корректурам. Дата издания: декабрь 1995 г.

IEC International Standard 61174 требования к морскому навигационному и радиокоммуникационному оборудованию ECDISсистем. Содержит описание методов проверки соответствия систем требованиям стандарта. Дата издания: 1998 г. [78,79].

Помимо соответствия перечисленным выше требованиям, электроннокартографическая система должна пройти процедуру официальной проверки в уполномоченной организации. [29].

Электроннокартографические системы, полностью удовлетворяющие требованиям ECDIS и получившие официальное подтверждение, могут стоить несколько десятков тысяч долларов. Более дешевой альтернативой являются электроннокартографические системы (ЭКС или ECS), которые либо не полностью соответствуют требованиям ECDIS, либо не прошли процедуру официального подтверждения соответствования. В отличие от ECDIS, ЭКС не могут служить официальной заменой традиционных бумажных карт, однако предоставляемые ими возможности могут оказать существенное подспорье судоводителю и повысить безопасность мореплавания. [22,88].

Представляемые сегодня на рынке программные средства ГИС мы будем условно разделять на три основных категории по степени их функциональности, как это представлено в таблице 1.

Таблица 1

Классификация программных средств ГИС-технологий

ГИС-средства	**Ввод и формирование пространственных данных**	**Создание и ведение атрибутивных БД**	**Запросы к БД**	**Пространственный анализ**
ГИС-конструкторы	да	да	да	да
ГИС-аналитики	ограничен	Да	да	да
ГИС-вьюверы	нет	ограничен	да	ограничен

ГИС-конструкторы (инструментальные полнофункциональные ГИС) - системы с наиболее широкими возможностями, включающие ввод,

редактирование, хранение (как пространственной, так и атрибутивной информации), а также сложные процедуры пространственного анализа и моделирования геоданных. Все это реализуется при помощи встроенного универсального инструментария или с помощью специальных языков для разработки приложений. Продукты этого класса изначально проектировались для функционирования в операционной системе UNIX на рабочих станциях, но по причинам, указанным выше, сегодня переходят на платформы INTEL.

Отдельным классом сегодня выделилась группа ГИС-продуктов для анализа данных готовых проектов - настольные ГИС-аналитики, обладающие многими свойствами своих полнофункциональных прототипов, но при этом требующих минимальных вычислительных ресурсов. Главная задача этого класса программного обеспечения ясна уже из их названия - это анализ информации, содержащейся в БД. ГИС-аналитики также, как и ГИС-конструкторы могут включать язык программирования, существенно расширяющий их возможности. Практически все из них позволяет организовать высококачественный вывод карт и таблиц на твердый носитель

Многие тяжелые инструментальные ГИС сопровождаются средствами для конечного пользователя - ГИС-вьюверы. Они предназначаются в основном для просмотра ранее введенной и структурированной по правам доступа информации.

Во все ГИС-вьюверы включается инструментарий запросов к локальным и удаленным базам данных. Как правило, ГИС- вьюверы предоставляют пользователю (если предоставляют вообще) крайне ограниченные возможности пополнения баз данных и предназначены в основном для просмотра и поиска необходимой информации. Естественно, такие «вьюеры» всегда входят составной частью в средние и крупные проекты, позволяя сэкономить затраты на создание части рабочих мест, не наделенных правами пополнения базы данных. Сегодня этот класс продуктов становится основным инструментом в геоинформационных WEB-технологиях [13,116], позволяя извлекать и визуализировать данные из удаленных информационных узлов сети INTERNET.

1.2.2 Анализ возможностей «классических» средств геоинформатики для представления отображения обстановки.

Функциональные особенности программных средств ГИС определяются их ориентацией на обработку и анализ атрибутивной информации. Это программы сбора, ввода в машинную среду, обработки (манипулирования, анализа, моделирования, визуализации) и представления пространственно-координированных данных в форме различных (табличных, графических, картографических) выходных документов. [2,15,42].

Структурно программные средства ГИС включают базовые программные средства, модули приложения и вспомогательные средства (утилиты), обеспечивающие решение всей совокупности перечисленных задач.

Наряду с этим имеются и облегченные программные продукты, предназначенные для просмотра информации в картографическом виде и решения простейших геоинформационных задач.

Большое распространение сегодня в судовождении получили разнообразные программные комплексы электронных картографических систем (ЭКДИС), основное назначение которых направлено на информационное обеспечение объектов судовождения с частичной информационной увязкой с береговыми службами безопасности. [1,2,21,25,117].

Базовые программные средства позволяют осуществить связь пространственной и атрибутивной информации, отображение пространственной и атрибутивной информации, организацию запросов для выбора или поиска необходимых пространственных объектов, редактирование атрибутивной информации и т. п. Базовые программные средства могут быть универсальными и специализированными.

Модули приложения работают вместе с базовыми средствами и позволяют решать специализированные задачи. Модули приложения работают с данными, передаваемыми от ГИС, но используют собственные машинные ресурсы.

Вспомогательные средства (утилиты) используются для выполнения необходимых операций без использования более дорогих базовых средств.

Вспомогательные программные средства зачастую являются более производительными за счет их меньшей универсальности. Практически всегда вспомогательные средства обладают более низким соотношением цена/производительность.

К вспомогательным средствам можно отнести конвертеры, векторизаторы, растеризаторы и др. [115].

Если попытаться описать множество функций, реализованных в наиболее популярных ГИС, модулях, приложениях и утилитах, то можно выделить следующие группы основных функций.

1. Первичная обработка растровых изображений.
2. Трансформация и привязка растровых изображении к системе координат (преобразования плоскости).
3. Преобразование растровых изображений в заданные картографические проекции.
4. Цветоделение растровых изображений.
5. Векторизация растровых изображений.
6. Преобразование векторных данных в заданные картографические проекции.
7. Преобразования векторных данных в заданные системы координат.
8. Конвертации данных в заданные форматы и из заданных форматов.
9. Анализ и редактирование топологии.
10. Ввод и предварительная обработка данных геодезических измерении.
11. Ввод и предварительная обработка данных дистанционного зондирования.
12. Визуализация векторных данных.
13. Визуализация растровых данных.
14. Настройка визуализации данных (выбор значков, штриховок рисунков заполнения, стилей рисования линий, выбор отображаемых слоев или

классов объектов, настройка диапазонов масштабов, в пределах которых отображается слой или класс объектов и др.)

15. Изменение масштаба визуализации (суммирования), смешения изображений (скроллинг).

16. Создание и редактирование структуры информации и самой информации в базах данных.

17. Организация запросов на отбор объектов по атрибутивной и пространственной информации.

18. Выполнение теоретико-множественных операций с полигонами и планарными разбиениями (создание нового планарного разбиения на базе двух или более исходных), определение общей части полигона, внешней части одного полигона по отношению к другому и т.д.

19. Построение картографического отображения информации (методы картограмм, картодиаграмм, локализованных диаграмм, значков, точечный, знаков движения, изолиний и др.). Задания параметров методов, выбор и расчет шкал, задания значений графических переменных визуализации.

20. Обработка растровой информации: операции с одним растром; сглаживание фильтром, построение буферных зон, вычисление значений на растре (кратчайшее расстояние от заданного множества точек); операции с несколькими растрами; поэлементное сравнение (больше, меньше, равно), поэлементные математические операции (сложение, вычитание и др.); определение взаимосвязи (вычисления коэффициентов корреляции); кластеризация; переклассификация.

21. Анализ информации в сетевом (графовом) представлении (поиск кратчайшего пути, поиск оптимального пути с учетом стоимости и ограничений движения и др.).

22. Геокодироваие (точки по координатам, объекты по адресу, точки по пикетажу на линейных объектах).

23. Вывод карт на печать, с разрезкой на листы цветоделением и т.п.

Это далеко не полный перечень. Он не включает специализированных функций моделирования, например, экологических расчетов в область мониторинга ЧС (прогноз паводковых явлений, прогноз схода снежных лавин, прогноз распространения пожаров и др.). [3,18].

Большинство упомянутых модулей являются предметно-ориентированными, создаются для анализа специфических явлений и процессов и не предназначены для решения универсальных задач пространственного анализа.

Основу любой интеллектуальной ГИС составляет ее программная составляющая. Именно она определяет основные свойства ГИС. Поэтому при анализе известных методов разработки интеллектуальных ГИС будем основной акцент делать на нее. Разработка любой ГИС, как правило, состоит из ряда этапов:

- анализ требований, предъявляемых к ГИС;
- определение спецификаций;
- проектирование системы;
- кодирование;
- тестирование.

Остановимся на первых трех из них.

Первый этап предусматривает анализ требований предъявляемых к разрабатываемой системе, которые сосредоточены, прежде всего, в интерфейсе между этой системой и источниками информации, пользователями. В анализ включаются такие вопросы как время, вероятность ошибки, стоимость обработки информации, число и виды решаемых прикладных задач, пространственно-временные показатели и другие. На этом этапе также определяется концепция ГИС СОО: основные понятия, которые будут лежать в основе ГИС, а также объекты и процедуры обработки информации, на основе которых будет строиться система. Подходить к решению этой задачи необходимо очень ответственно, так как именно концепция будущей системы и совершенство модели данных определят её успех. При этом разработчику

приходится учитывать множество факторов – достоинства и недостатки концепций уже существующих систем, постоянно изменяющиеся требования со стороны прикладных задач, которые должна будет решать система, изменения в информационных технологиях и многое другое.

Для обоснования требований к интеллектуальным ГИС используют специальные методы. С формальной точки зрения они предусматривающие решение обратных задач анализа вышестоящих процессов. В нашем случае это процессы управления освещением обстановки в ближней морской зоне. Именно из их анализа предъявляются требования к интеллектуальным ГИС, обеспечивающим данные процессы. К таким методам относятся положения классической теории вероятностей, математический аппарат марковских и полумарковских процессов, сети Петри, Е-сети, сетевое планирование и другие [80]. Посредством их формализуют физические процессы и получают количественные характеристики интересующих требований, которым должна удовлетворять интеллектуальная ГИС. Несмотря на относительную развитость таких методов в явном виде нет моделей позволяющих сравнивать традиционные и интеллектуальные ГИС обеспечения освещения обстановки и предъявлять требования к последним по эффективности.

На этапе определения спецификаций осуществляется точное описание функций системы, задается структура входных и выходных данных, решается комплекс вопросов, имеющих отношение к структуре файлов, организации доступа к данным, обновлению и удалению последних. Спецификации указывают только те функции, которые система должна выполнять, не указывая, каким образом это достигается. Составление подробных алгоритмов на данном этапе не осуществляется. Однако должны использоваться концептуальные обобщенные модели и алгоритмы функционирования ГИС, формулируются математические и информационные задачи, которые должны быть решены. В качестве таких моделей используются функциональные и структурные схемы, архитектуры самой ГИС [21, 65, 66, 70, 72, 79], концептуальные модели предметных областей, систем управления данными и другие. Например, в

работах [21, 65, 66] варианты архитектур интеллектуальных ГИС представлены соответственно в виде схем, представленных на рисунках 1 – 3. На рис. 3: *Spatial DB* – база пространственно соотнесенных данных; *KB* – база знаний; *ES Shell* – оболочка экспертной системы.

Общей особенностью этих схем является интеграция традиционных ГИС с интеллектуальными элементами. Для интеграции экспертных систем с традиционными ГИС могут применяться также подходы, изложенные в [68]. Общий недостаток этих архитектур в том, что в них изначально интеллектуализация процессов осуществляется в узком смысле с опорой на экспертные системы с целью автоматизированной выработки рекомендаций или решений прикладных задач. Эти подходы не предусматривают самообучение ГИС и их саморазвитие, что существенно ограничивает их возможности.

Для организации взаимосвязи между географической и атрибутивной информацией используют четыре подхода взаимодействия.

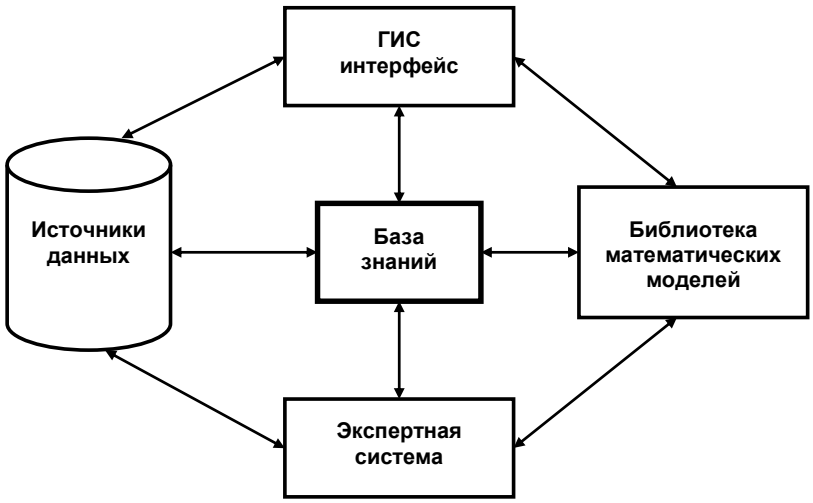

Рис. 2 – Пример архитектуры интеллектуальных ГИС (вариант а)

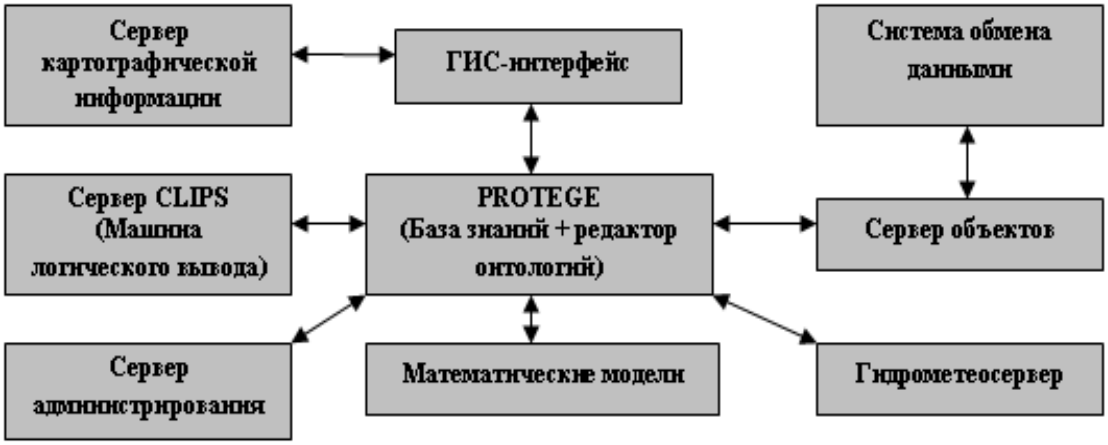

Рис. 3 – Пример архитектуры интеллектуальных ГИС (вариант б)

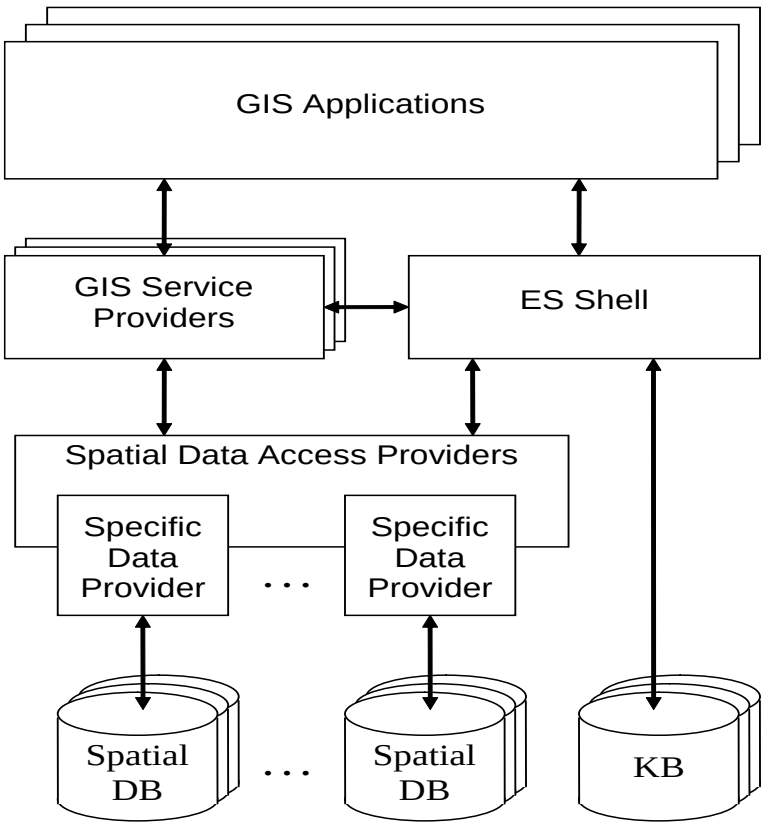

Рис. 4 – Пример архитектуры интеллектуальных ГИС (вариант с)

Первый подход – геореляционный (гибридный). При таком подходе географические и атрибутивные данные организованы по-разному. Между двумя типами данных связь осуществляется посредством идентификатора объекта. Геоинформация хранится отдельно от атрибутивной в своей базе данных. Атрибутивная информация организована в таблицы пол управлением реляционной системы управления базой данных (СУБД).

Следующий подход называется интегрированным. При этом подходе предусматривается использование средств реляционных СУБД как для хранения пространственной, так и атрибутивной информации. В этом случае ГИС выступает в качестве надстройки над СУБД. Третий подход называют объектным. Плюсы этого подхода в легкости описания сложных структур данных и взаимоотношений между объектами. Объектный подход позволяет выстраивать иерархические цепочки объектов и решать многочисленные задачи моделирования. В последнее время самое широкое распространение получил объектно-реляционный подход, являющийся синтезом первого и третьего подходов.

Организация пространственных данных в ГИС может осуществляться в соответствии со слоевой, векторно-топологической, векторно-нетопологической моделями. Наряду со слоевой моделью используют объектно-ориентированную модель [65, 72].

Интеллектуальные подсистемы ГИС, в которые, в частности, входят экспертные системы, в настоящее время могут разрабатываться в соответствии с положениями теории формальных систем и теории эволюционных вычислительных систем. К теории формальных систем относятся логические и эмпирические методы представления знаний и выводов. Элементами теории эволюционных вычислительных систем являются нейронные сети, генетические алгоритмы, методы иммунокомпьютинга и другие.

Несмотря на достигнутые результаты в области применения этих методов для обработки геоинформации в ГИС, вопросы текущего прогнозирования анализируемой обстановки БМЗ в интеллектуальных подсистемах не решены. Нет тесной связи применяемых машин логического вывода с моделями из библиотеки прикладных задач.

Несомненно, если ряд важных функций интеллектуальных ГИС не проработан на этапах обоснования требований к ним и разработки спецификаций, то на этапе проектирования их это сделать довольно сложно.

На этапе проектирования разрабатываются алгоритмы, задаваемые спецификациями, и формируется общая структура информационной системы. При этом одни и те же задачи сбора, обработки, отображения, распределения геоинформации и манипулирования ею могут решаться с использованием различных алгоритмов в рамках сформулированных задач. В ряде случаев поиск эффективных алгоритмов на этом этапе осуществляется путем их заимствования или синтеза по интересующим показателям с применением известных методов [80]. При решении новых задач приходится разрабатывать также и новые алгоритмы их решения.

В целом анализ известных методов разработки интеллектуальных ГИС показывает, что они во многом несовершенны, как в части обоснования требований к этим системам, так и расширения их функций по текущему прогнозированию обстановки в БМЗ на электронных картах, обучению и саморазвитию.

Требуется поиск новых подходов к разработке интеллектуальных ГИС нового поколения, обладающих интеллектуальностью в широком смысле.

1.3 Анализ особенностей возможности применения искусственных нейронных сетей в современных ГИС

1.3.1 Особенности ИНС, находящие применение в ГИС

Модели НС могут быть программного и аппаратного исполнения. В дальнейшем речь пойдет в основном о первом типе.

Несмотря на существенные различия, отдельные типы НС обладают несколькими общими чертами.

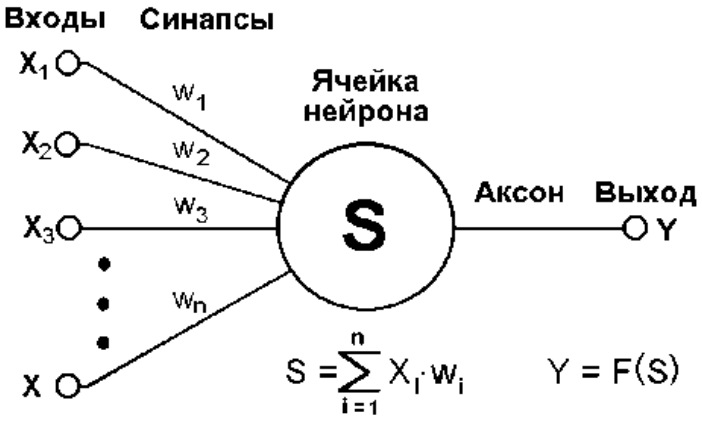

Рис.5. Пример искусственного нейрона

Во-первых, основу каждой НС составляют относительно простые, в большинстве случаев – однотипные, элементы (ячейки), имитирующие работу нейронов мозга. Далее под нейроном будет подразумеваться искусственный нейрон, то есть ячейка НС. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Он обладает группой синапсов – однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон – выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Общий вид нейрона приведен на рисунке 5. Каждый синапс характеризуется величиной синаптической связи или ее весом w_i, который по физическому смыслу эквивалентен электрической проводимости[14,15,24].

Текущее состояние нейрона определяется, как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} x_i \cdot w_i \qquad (1)$$

Выход нейрона есть функция его состояния:

$$y = f(s) \qquad (2)$$

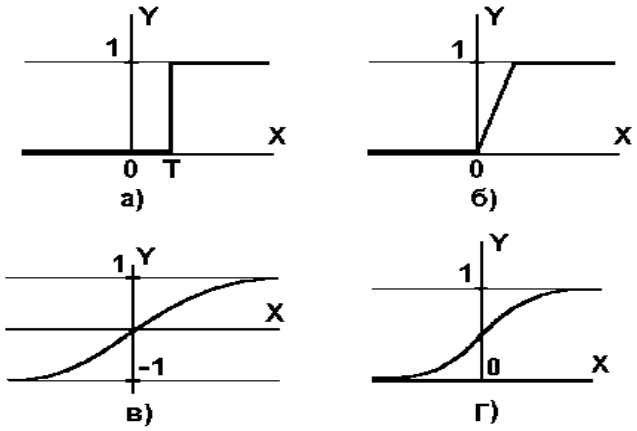

Рис.6. а) функция единичного скачка; б) линейный порог (гистерезис); в) сигмоид – гиперболический тангенс; г) сигмоид – формула (3)

Нелинейная функция f называется активационной и может иметь различный вид, как показано на рисунке 6. Одной из наиболее распространеных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоид (т.е. функция S-образного вида)[2,17]:

$$f(x) = \frac{1}{1+e^{-\alpha x}}. \qquad (3)$$

При уменьшении α сигмоид становится более пологим, в пределе при α=0 вырождаясь в горизонтальную линию на уровне 0.5, при увеличении α сигмоид приближается по внешнему виду к функции единичного скачка с порогом T в точке x=0. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне [0,1]. Одно из ценных свойств сигмоидной функции – простое выражение для ее производной, применение которого будет рассмотрено в дальнейшем.

$$f'(x) = \alpha \cdot f(x) \cdot (1 - f(x)). \qquad (4)$$

Следует отметить, что сигмоидная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон.

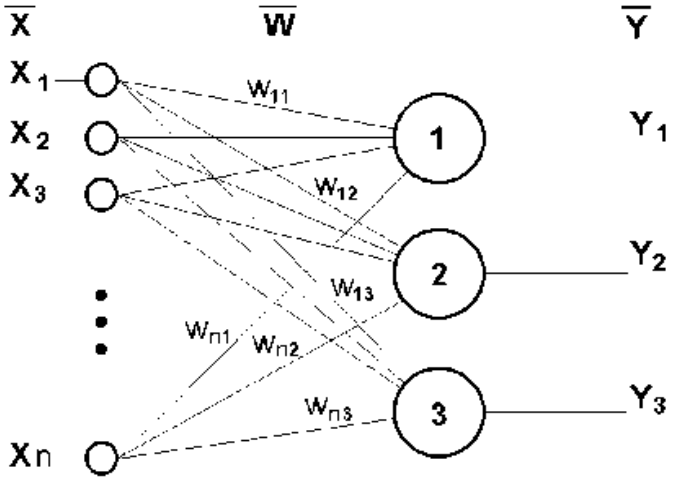

Рис.7. Пример однослойного перцептрона

Возвращаясь к общим чертам, присущим всем НС, отметим, во-вторых, принцип параллельной обработки сигналов, который достигается путем объединения большого числа нейронов в так называемые слои и соединения определенным образом нейронов различных слоев, а также, в некоторых конфигурациях, и нейронов одного слоя между собой, причем обработка взаимодействия всех нейронов ведется послойно.

В качестве примера простейшей НС рассмотрим трехнейронный перцептрон (рис.7), то есть такую сеть, нейроны которой имеют активационную функцию в виде единичного скачка. На n входов поступают некие сигналы, проходящие по синапсам на 3 нейрона, образующие единственный слой этой НС и выдающие три выходных сигнала:

$$y_j = f\left[\sum_{i=1}^{n} x_i \cdot w_{ij}\right], \quad j=1...3. \tag{5}$$

Очевидно, что все весовые коэффициенты синапсов одного слоя нейронов можно свести в матрицу **W**, в которой каждый элемент w_{ij} задает величину i-ой

синаптической связи j-ого нейрона. Таким образом, процесс, происходящий в НС, может быть записан в матричной форме:

$$Y=F(XW), \quad (6)$$

где X и Y – соответственно входной и выходной сигнальные векторы, F(V) – активационная функция, применяемая поэлементно к компонентам вектора V.

Теоретически число слоев и число нейронов в каждом слое может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуется НС. Чем сложнее НС, тем масштабнее задачи, подвластные ей[9,26].

Выбор структуры НС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные, на сегодняшний день, конфигурации. Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом он руководствуется несколькими основополагающими принципами: возможности сети возрастают с увеличением числа ячеек сети, плотности связей между ними и числом выделенных слоев; введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети; сложность алгоритмов функционирования сети (в том числе, например, введение нескольких типов синапсов – возбуждающих, тромозящих и др.) также способствует усилению мощи НС. Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза НС сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора.

Очевидно, что процесс функционирования НС, то есть сущность действий, которые она способна выполнять, зависит от величин синаптических связей, поэтому, задавшись определенной структурой НС, отвечающей какой-либо

задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными).

Этот этап называется обучением НС, и от того, насколько качественно он будет выполнен, зависит способность сети решать поставленные перед ней проблемы во время эксплуатации. На этапе обучения кроме параметра качества подбора весов важную роль играет время обучения. Как правило, эти два параметра связаны обратной зависимостью и их приходится выбирать на основе компромисса.

Обучение НС может вестись с учителем или без него. В первом случае сети предъявляются значения как входных, так и желательных выходных сигналов, и она по некоторому внутреннему алгоритму подстраивает веса своих синаптических связей. Во втором случае выходы НС формируются самостоятельно, а веса изменяются по алгоритму, учитывающему только входные и производные от них сигналы.

Существует великое множество различных алгоритмов обучения, которые однако делятся на два больших класса: детерминистские и стохастические. В первом из них подстройка весов представляет собой жесткую последовательность действий, во втором – она производится на основе действий, подчиняющихся некоторому случайному процессу[13,31].

Развивая дальше вопрос о возможной классификации НС, важно отметить существование бинарных и аналоговых сетей. Первые из них оперируют с двоичными сигналами, и выход каждого нейрона может принимать только два значения: логический ноль ("заторможенное" состояние) и логическая единица ("возбужденное" состояние). К этому классу сетей относится и рассмотренный выше перцептрон, так как выходы его нейронов, формируемые функцией единичного скачка, равны либо 0, либо 1. В аналоговых сетях выходные значения нейронов способны принимать непрерывные значения, что могло бы иметь место после замены активационной функции нейронов перцептрона на сигмоид.

Еще одна классификация делит НС на синхронные и асинхронные. В первом случае в каждый момент времени свое состояние меняет лишь один нейрон. Во втором – состояние меняется сразу у целой группы нейронов, как правило, у всего слоя. Алгоритмически ход времени в НС задается итерационным выполнением однотипных действий над нейронами. Далее будут рассматриваться только синхронные НС..

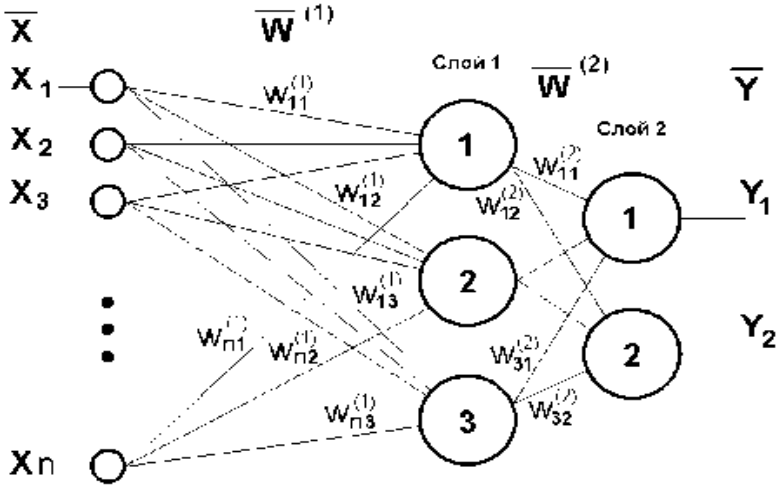

Рис.8. Пример двухслойного перцептрона

Сети также можно классифицировать по числу слоев. На рисунке 4 представлен двухслойный перцептрон, полученный из перцептрона с рисунка 3 путем добавления второго слоя, состоящего из двух нейронов. Здесь уместно отметить важную роль нелинейности активационной функции, так как, если бы она не обладала данным свойством или не входила в алгоритм работы каждого нейрона, результат функционирования любой p-слойной НС с весовыми матрицами $\mathbf{W}^{(i)}$, i=1,2,...p для каждого слоя i сводился бы к перемножению входного вектора сигналов **X** на матрицу

$$W^{(\Sigma)}=W^{(1)}\cdot W^{(2)}\cdot\ldots\cdot W^{(p)}\ , \qquad (7)$$

то есть фактически такая p-слойная НС эквивалентна однослойной НС с весовой матрицей единственного слоя $W^{(\Sigma)}$:

$$Y=XW^{(\Sigma)}\ . \qquad (8)$$

Продолжая разговор о нелинейности, можно отметить, что она иногда вводится и в синаптические связи. Большинство известных на сегодняшний день

НС используют для нахождения взвешенной суммы входов нейрона формулу (1), однако в некоторых приложениях НС полезно ввести другую запись, например:

$$s = \sum_{i=1}^{n} x_i^2 \cdot w_i \,, \tag{9}$$

или

$$s = \sum_{i=1}^{n} x_i \cdot x_{((i+1) \bmod n)} \cdot w_i \,. \tag{10}$$

Вопрос в том, чтобы разработчик НС четко понимал, для чего он это делает, какими ценными свойствами он тем самым дополнительно наделяет нейрон, и каких лишает. Введение такого рода нелинейности, вообще говоря, увеличивает вычислительную мощь сети, то есть позволяет из меньшего числа нейронов с "нелинейными" синапсами сконструировать НС, выполняющую работу обычной НС с большим числом стандартных нейронов и более сложной конфигурации.

Многообразие существующих структур НС позволяет отыскать и другие критерии для их классификации, но они выходят за рамки данной работы.

Теперь рассмотрим один нюанс, преднамеренно опущенный ранее. Из рисунка функции единичного скачка видно, что пороговое значение T, в общем случае, может принимать произвольное значение. Более того, оно должно принимать некое произвольное, неизвестное заранее значение, которое подбирается на стадии обучения вместе с весовыми коэффициентами. То же самое относится и к центральной точке сигмоидной зависимости, которая может сдвигаться вправо или влево по оси X, а также и ко всем другим активационным функциям. Это, однако, не отражено в формуле (1), которая должна была бы выглядеть так:

$$s = \sum_{i=1}^{n} x_i \cdot w_i - T \,. \tag{11}$$

Дело в том, что такое смещение обычно вводится путем добавления к слою нейронов еще одного входа, возбуждающего дополнительный синапс каждого из нейронов, значение которого всегда равняется 1. Присвоим этому входу номер 0. Тогда

$$s = \sum_{i=0}^{n} x_i \cdot w_i \,, \qquad (12)$$

где $w_0 = -T$, $x_0 = 1$.

Очевидно, что различие формул (1) и (12) состоит лишь в способе нумерации входов.

Из всех активационных функций, изображенных на рис. 2, одна выделяется особо. Это гиперболический тангенс, зависимость которого симметрична относительно оси X и лежит в диапазоне [-1,1]. Забегая вперед, скажем, что выбор области возможных значений выходов нейронов во многом зависит от

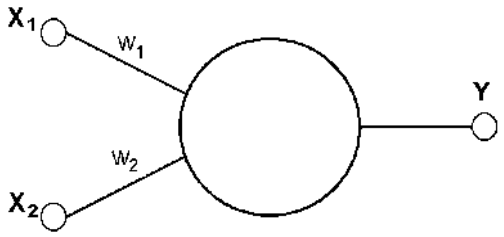

Рис.9. Однонейронный перцептрон

конкретного типа НС и является вопросом реализации, так как манипуляции с ней влияют на различные показатели эффективности сети, зачастую не изменяя общую логику ее работы. Пример, иллюстрирующий данный аспект, будет представлен после перехода от общего описания к конкретным типам НС.

Работа всех сетей сводится к классификации (обобщению) входных сигналов, принадлежащих n-мерному гиперпространству, по некоторому числу классов. С математической точки зрения это происходит путем разбиения гиперпространства гиперплоскостями (запись для случая однослойного перцептрона)

$$\sum_{i=1}^{n} x_i \cdot w_{ik} = T_k \,, \quad \text{k=1...m} \,. \qquad (13)$$

Каждая полученная область является областью определения отдельного класса. Число таких классов для одной НС перцептронного типа не превышает 2^m, где m – число выходов сети. Однако не все из них могут быть разделимы данной НС[6].

Например, однослойный перцептрон, состоящий из одного нейрона с двумя входами, представленный на рисунке 9, не способен разделить плоскость (двумерное гиперпространство) на две полуплоскости так, чтобы осуществить классификацию входных сигналов по классам A и B [28,37].

Уравнение сети для этого случая

$$x_1 \cdot w_1 + x_2 \cdot w_2 = T \qquad (14)$$

является уравнением прямой (одномерной гиперплоскости), которая ни при каких условиях не может разделить плоскость так, чтобы точки из множества входных сигналов, принадлежащие разным классам, оказались по разные

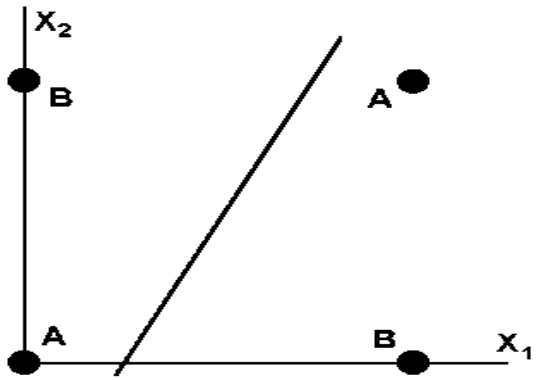

Рис.10 Визуальное представление работы НС, представленной на рис. 8
В данном случае оси несут в себе произвольные координаты для удобства рассмотрения

стороны от прямой (см. рису. 8).

Если рассматривать табл. 1, можно заметить, что данное разбиение на классы реализует логическую функцию исключающего ИЛИ для входных сигналов. Невозможность реализации однослойным перцептроном этой функции получила название проблемы исключающего ИЛИ.

Функции, которые не реализуются однослойной сетью, называются линейно неразделимыми. Решение задач, подпадающих под это ограничение, заключается в применении 2-х и более слойных сетей или сетей с нелинейными синапсами, однако и тогда существует вероятность, что корректное разделение некоторых входных сигналов на классы невозможно.

1.3.2 Основные задачи, решаемые ИНС в применении к ГИС

Опишем круг задач, требующих решения в ГИС, для которых могут быть использованы нейросетевые технологии.

Построение (дополнение) слоя

Основная задача, к которой, так или иначе, относятся остальные, описанные ниже, это построение слоя. Она означает заполнение его недостающих частей (или построение слоя полностью) по информации, имеющейся в других слоях, на основе нахождения некоторой функциональной зависимости между параметрами, полученными эмпирическим путем, и скрытыми теоретическими параметрами, определяющими сущностные характеристики каждой конкретной точки.

Даны слои качественных характеристик одной и той же территории. Слой, который необходимо восстановить, известен частично. Для восстановления слоя при обучении нейросети используется только та информация из слоев, которая покрывает известные участки слоя с пробелами. После обучения можно распространить знания о зависимости между слоями на отсутствующие области карты. Получившиеся знания обладают переносимостью за рамки данной территории. Все описанные ниже задачи можно рассматривать как частный случай данной.

Классификационные задачи. Поскольку при сборе информации для БД приходится иметь дело с результатами измерений, определим по этому показателю три типа задач классификации.

К задачам классификации первого типа относятся те, в которых исходные измерения требуется разделить на устойчивые группы. Их называют задачами классификации без учителя, кластеризации, таксономии, типизации [56-62]. Этот тип классификации применяется для обработки опытных данных.

Задачи классификации второго типа характеризуются тем, что исходные данные уже сгруппированы и требуется оценить их информативность (значимость) относительно совокупности известных эталонов. Такого рода

задачи встречаются при распознавании образов [63,65], дешифрировании снимков и т.д.

Задачи классификации третьего типа - задачи разбиения. В них исходные измерения или их функции требуется разбить на устойчивые группы в зависимости от их величины (типичный пример - зонирование) [23, 30, 66].

В ГИС задачи классификации первого типа возникают и решаются при разработке классификаторов, т.е. при организации информационной основы, задачи второго типа - при сборе первичных данных и при использовании ГИС для экспертных решений или оценок. Задачи классификации третьего типа возникают в приложениях ГИС для решения проблем в области экологии, землепользования, статистики и т.п.

Восстановление легенды слоя

Вторая решаемая задача - восстановление легенды. Классификация с учителем - генерация объектов слоя по заданным классификационным правилам. Правила задаются во время обучения нейросети и остаются скрытыми от пользователя. Пользователь имеет возможность задавать, по его мнению, полезные для классификации признаки, выбрав слои, участвующие в обучении. Типичная задача поиск полезных ископаемых по косвенным признакам. Эта задача решается на основе информации об уже разведанных месторождениях и полевых съемках косвенных признаков. Знания, полученные при обучении, переносимы на другую территорию с известными косвенными признаками[58].

Районирование и типология

Зонирование. Основное назначение функций этой группы состоит в построении новых объектов - зон до того на карте не существовавших, т.е. участков территорий, однородных в смысле некоторого критерия или группы критериев. Границы зон могут либо совпадать с границами ранее существовавших объектов (задача определения "нарезки" избирательных округов по сетке квартального деления), либо строиться в результате различных видов моделирования (зоны экологического риска). Типичные задачи этого типа: выделение зон градостроительной ценности территорий, зон экологического

риска, зонирование урбанизированных территорий по транспортной доступности, построение зон обслуживания поликлиник и т.д. Работа может производиться как с растровыми, так и с векторными изображениями.

В сущности, зонирование это - классификация без учителя. Задан набор объектов, каждому объекту сопоставлен вектор значений признаков (строка таблицы). Требуется разбить эти объекты на классы эквивалентности.

Отнесение объекта к классу проводится путем его сравнения с типичными элементами разных классов и выбора ближайшего. Простейшая мера близости объектов - квадрат евклидового расстояния между векторами значений их признаков (чем меньше расстояние, тем ближе объекты). Соответствующее определение признаков типичного объекта - среднее арифметическое значение признаков по выборке, представляющей класс. Другая мера близости, естественно возникающая при обработке сигналов, изображений и т.п. - квадрат коэффициента корреляции (чем он больше, тем ближе объекты). Возможны и иные варианты - все зависит от задачи. Для каждого нового объекта нужно выполнить два действия:

1) найти класс, к которому он принадлежит;

2) использовать новую информацию, полученную об этом объекте, для исправления (коррекции) правил классификации.

В результате классификации как бы появляются новые имена и правила их присвоения.

Создание моделей поверхностей

Создание моделей поверхностей - это и построение моделей изолинейных изображений по регулярным и нерегулярным сеткам и создание модели трехмерной визуализации, например, построение панорамы города в аксонометрической или иной проекции. Расчет производится по содержащимся в базах данных численным характеристикам. Моделироваться могут, как изображения действительного рельефа или непрерывного поля, современного или с учетом динамических изменений, так и воображаемые поверхности,

построенные по одному или нескольким показателям, например, поверхность цен на землю, плотность дорожной сети или населения и т.п.

Интерполяция и прогнозное картирование

Задача - интерполяция пространственно распределенных данных. Сводиться к задаче построения функции по конечному набору значений и как следствие к задаче заполнения пробелов. Цель - извлечение максимума информации из набора данных, учитывая возможные ошибки измерений, неравномерную плотность сетки мониторинга, и прочие помехи, встречающиеся при реальных измерениях. Данные по окружающей среде обладают неоднородностью как на крупных, так и на мелких масштабах, что затрудняет анализ. Нейросетевая обработка обладает рядом преимуществ перед детерминистическими моделями.

Временной анализ

Временной анализ растровых изображений. В качестве таких изображений в ГИС обычно выступают снимки или растеризованные векторные изображения. Преимущество снимков - в их современности и достоверности, поэтому часто встречающийся вид анализа в этой группе - временной. Сравниваются и ищутся различия между снимками различной давности, таким образом, оценивается динамика произошедших изменений. Не менее часто анализируются пространственные взаимосвязи двух или нескольких явлений.

Анализ временных рядов содержит комплекс задач, которые сводятся к построению функций по конечным наборам значений и заполнению пробелов в таблицах. Временные ряды представляют собой специальный вид таблиц и заслуживают отдельного рассмотрения. Для каждого типа объектов выделяется набор постоянных признаков (констант) и множество свойств, меняющихся со временем (переменных признаков). Предполагается, что в любой момент времени для каждого объекта существуют свои значения переменных признаков. Вот, например, три задачи, специфичные для обработки временных рядов:

а) определение констант (всех или части) по известным значениям переменных в разные моменты времени;

б) предсказание значений части переменных в некоторый момент времени по известным значениям констант, переменных в нескольких предшествовавших моментах времени и части переменных в текущий момент;

в) определение объема данных о прошлом, достаточных для предсказания будущего на конкретное время и с заданной точностью.

Обычная задача при временном анализе - получение прогноза. Легко заметить, что решение такой задачи немногим отличается от решения задачи по восполнению пробелов в слое на основе информации, заключенной в других слоях. Единственное концептуальное отличие состоит в том, что слои вместо разных пространственных признаков содержат изменение во времени одного и того же слоя. Для примера, упростим описание анализа временных рядов. Возьмем за основу известные слои одной и той же территории в количестве N-1 (без последнего), а в качестве восстанавливаемого слой с номером N. Произведем обучение нейросети. Для прогноза в качестве входных параметров возьмем слои в количестве N-1 (без первого) и подадим на вход нейросети. На выходе получим прогнозируемый слой N+1[11,24].

Выбор значимых признаков

Анализ значимости. Как уже было сказано, основной задачей является восполнение пробелов в данных и решается она применительно к данной области построением слоя по слою (или нескольким слоям). При этом исследуется вопрос, какие из входных сигналов являются доминирующими, (значимыми) при принятии нейросетью решения, а какие нет. Другими словами, насколько каждый слой участвующий в построении влияет на восполнение пробелов. Такая информация дает знание, например, о том какие признаки можно убрать из рассмотрения, а какие оставить. То есть решается задача нахождения оптимального набора исходных показателей, которые полностью описывают изучаемые явления. Это может помочь в понимании сущности географического комплекса.

Значимость по слою складывается из значимости точек сетки. Благодаря такой информации можно видеть, какие области из слоев участвующих в

качестве входов были значимы при построении. Таким образом, получаем представление о территориальном распределении значимости.

1.3.3 Сравнение и анализ моделей ИНС, применяемых в работе с ГИС-системами

В литературе заметное внимание уделяется вопросам архитектуры технических нейронных сетей, приведем вариант соответствующей классификации схем:

- однослойные;
- многослойные;
- полносвязные.

Более принципиальным является разбиение нейроалгоритмов на два класса – Supervised (обучающиеся по образцу, с Учителем) и Unsupervised (обучающиеся без образца, без Учителя). В первом случае обучение организовано как воспроизведение набора правильных образцов (обучающей выборки), после чего сеть может адекватно реагировать и на примеры, которых не было в обучающей выборке, во втором случае образцы правильной реакции исходно отсутствуют. В части русской литературы утвердились термины обучение с учителем и обучение без учителя, что не является точным переводом с английского, и не вполне точно по нормам русского языка. Видимо, нейросети, обучающиеся по образцу, неплохо воспроизводят рефлекторное поведение. Нейросети, обучающиеся без образца, быть может, иногда моделируют более интересную вещь – мышление, однако, делают это несравненно менее успешно.

Нейросети, обучающиеся по образцу, произошли от персептронов, и в современной трактовке могут рассматриваться как варианты и модификации сетей с обратным распространением ошибки (иногда как результат примитивизации такого рода сетей, с целью упрощения реализации). К этому классу можно отнести, например, однослойный и многослойный перцептрон, машину Больцмана, сети, обучающиеся по правилу Хебба, рекуррентные слоистые и полносвязные сети обратного распространения ошибки, сети,

использующие радиальные базисные функции. Различия между указанными системами порой достаточно велики, но всегда есть немало общего, а детали классификации различаются у разных авторов[80].

Нейросети, обучающиеся без образца, пожалуй, более разнообразны, хотя стоящая за ними теория математически порой более примитивна – это карты Кохонена, системы с множественными локально устойчивыми состояниями, такие как сеть Хопфилда, сети, настраивающиеся на основе адаптивного резонанса. Прямые аналогии между данными классами не просматриваются, хотя часто исходно имеются необработанные данные, а в итоге возникают их образы, построенные в ходе работы нейросети, либо сама сеть, меняя свою структуру, моделирует образы данных. В живой природе есть аналоги и этому – строились карты возбуждения участков коры мозга в зависимости от возбуждения участков тела, получившие названия «гомункулусов», оттого, что на этих картах формируется узнаваемый образ человека, только ладони, например, получаются увеличенными, а спина - уменьшенной.

Наконец, разрабатываются, комбинированные подходы. Идеология такого комбинирования порой заставляет вспомнить лозунг «человека создал труд». За основу берется алгоритм обучения по образцу, произвольно устанавливающий некое первоначальное, можно сказать «абстракционистское», соответствие между «сырыми данными» и «обработанными данными». Затем «обработанные данные» меняются в ходе внешней «трудовой деятельности» с учетом «свойств материала» так что соответствие улучшается. Устанавливается новое соответствие между исходными и обработанными данными, вновь меняются обработанные данные, и так далее. В итоге нейросеть порождает с одной стороны «художественный образ» ситуации, а с другой стороны – собственный навык быстрого, как бы рефлекторного, соотнесения реальных данных и их образов. Например, так можно проверять связность образа данных – если «природа материала» не позволяет ему изменять связность, а выше обозначенный подход работает, то значит и у образа данных связность та же. Данный пример приводится для напоминания о кризисе нейросетевого подхода,

имевшим место до восьмидесятых годов, как раз в связи с пессимизмом по части возможностей использования нейроалгоритмов в задачах топологии.

Под именем нейросетевых алгоритмов сегодня объединяется несколько подходов к обработке данных, которые их авторы, не согласовывая друг с другом, сочли напоминающими принципы организации биологических нейронных сетей. Видимо, сыграла роль привлекательность названия, вместе с тем обстоятельством, что по-настоящему принципы работы таких сложных биологических систем, как мозг человека, никому не известны, и в этом смысле все равны и свободны. Это несколько нарушает существующие в математике традиции логически обоснованной классификации алгоритмов, но поскольку некоторые нейроалгоритмы достаточно эффективны, приходится считаться с установившейся практикой. Здесь мы обсудим два типа нейроалгоритмов, наиболее часто используемых в приложениях – алгоритмы обратного распространения ошибки (back error propagation algo-rithms; BackProp; в российских публикациях восьмидесятых годов использовалось математически более культурное название: алгоритмы двойственного функционирования; АДФ) и карты Кохонена (самоорганизующиеся карты, self-organization maps, SOM).

1.3.4. Текущее состояние применения модулей ИНС в ГИС

На рынке программного обеспечения в настоящее время имеется множество самых разнообразных программ для моделирования нейронных сетей. Поиск в Интернете дает сотни ссылок на зарубежные и российские сайты. Можно выделить несколько основных функций, которые реализованы во всех этих программах:

- формирование, конструирование нейронной сети;
- обучение нейронной сети;
- имитация функционирования (тестирование) обученной нейронной сети.

С точки зрения компьютерной технологии и программных интерфейсов они

опираются на современные стандарты - от простых программ, ориентированные на платформу Unix с текстовым интерфейсом, до сложных модульных продуктов, базирующихся на последних технологических решениях от Microsoft.

Интегрированные решения на основе ГИС и нейронных сетей пока представлены слабо, несмотря на то что повышение функциональной мощности геоинформационных пакетов за счет интеграции специальных модулей расширения или ГИС-приложений - одна из важнейших черт современных геоинформационных систем. Проблема интеграции нейронных сетей и ГИС может быть решена по крайней мере тремя способами:

- интеграция (встраивание) нейросетевых моделей в ГИС с использованием специализированных средств геоинформационной системы (программирование на встроенных языках типа Avenue, MapBasic и т.п.);
- развитие интерфейса между отдельными приложениями нейросетевого анализа и ГИС, как самостоятельными системами (например, через обмен данными по DDE);
- создание прикладного программного обеспечения нейросетевых систем с элементами ГИС (например, с использованием библиотек классов типа MapObjects, GeoConstructor, MapX и проч.).

Выбор конкретного варианта связан с требованиями и постановкой задачи, имеющимися ресурсами и опытом работы. Особенности реализации определяются возможностями используемого ПО - как со стороны нейронной сети, так и со стороны ГИС, например, встроенным языком программирования, средствами OLE и DDE, функциональными возможностями DLL и ActiveX. На основе выбранной технологии создается соответствующее нейросетевое ГИС-приложение[75].

Ниже приводятся два конкретных примера - программных продукта, которые уже созданы на основе нейросетей и ГИС.

Программа NeRIS предназначена для тематической интерпретации пространственных данных, в первую очередь данных дистанционного

зондирования Земли. Основным инструментом, реализованным в программе, являются нейронные сети Кохонена, используемые для ординации, классификации и тематической интерпретации. Являясь одним из методов классификации многомерных данных, нейронные сети Кохонена обладают важными дополнительными свойствами, на которых основана значительная часть используемых в программе алгоритмов[17].

Возможности пакета тематической обработки растровых изображений в программе ScanEx-NeRIS:

- оценка количества классов, требуемых для описания тематики и составления тематической карты;
- оценка внутренней дробности, неоднородности тематических объектов (контуров);
- оценка распределения свойств экспертных объектов в признаковом поле дистанционной модели;
- оценка вероятностей присутствия тематических объектов, заданных экспертом в поле признаков снимка (выделение на изображении областей с различным уровнем оценки: оптимистическим, реалистическим, пессимистическим);
- построение иерархических классификаций с оценкой близости классов между собой (удобный для специалистов по тематической картографии интерфейс обработки «карты расстояний» («дерева расстояний») с целью уточнения и корректировки строения легенды и географического содержания тематической карты;
- создание тематически ориентированных нейронных сетей для последующей обработки растра с целью выявления тематических объектов;
- автотрассировка (векторизация) результатов поклассовой обработки;
- поддержка системы координат наиболее распространенных отечественных и зарубежных картографических проекций;
- экспорт растровых покрытий и векторных слоев в обменных

форматах наиболее распространенных географических информационных систем;

- представление результаты классификации для всех видов нейронных сетей как присвоением индекса класса каждому классифицированному пикселу, так и созданием растровых слоев "вероятности" (possibility) принадлежности пиксела одному конкретному классу (создание нескольких таких слоев с последующей их визуализацией позволяет наглядно представлять результаты классификации, например, выявлять "белые пятна" (неклассифицированные области пространства) и представлять данные для окончательной классификации традиционными методами).

Модуль Arc-SDM - одно из свободно доступных расширений ArcView для моделирования в ГИС на основе алгоритмов нечеткой логики и нейронных сетей. С точки зрения пользователя ГИС процесс пространственного моделирования с использованием этого модуля состоит в построении нового тематического слоя на основе нескольких уже существующих. Arc-SDM использует два нейросетевых алгоритма, которые вынесены в самостоятельный программный модуль DataXplore. Первый построен на основе нейросети, использующей радиальные базисные функции, второй на основе кластеризации в нечеткой логике. Нейросеть, использующая радиальные базисные функции должна пройти этап обучения, в результате которого будет сгенерирован набор параметров, определяющий взаимосвязь между входными слоями данных и выходным (результирующим) слоем. После этого для классификации данных можно использовать обученную нейросеть[44].

Например, в геологической задаче исследования полезных ископаемых, результирующий слой содержит сведения о наличии или отсутствии месторождений. Входные данные, используемые в процессе обучения, можно разделить на два типа - местоположения известных месторождений и участки территории, про которые известно, что там полезных ископаемых нет. На основе исходных векторных тематических слоев геоинформационной системы

создается grid-тема, далее подготовленный набор данных передается в программный модуль DataXplore. Результат вычислений отображается в виде нового тематического слоя.

1.4 Основные результаты анализа и сравнения ГИС-продуктов и остановка задачи исследования

Анализ литературы показал фактически полное отсутствие использование искуственных нейронных сетей (ИНС) среди огромного количества работ, связанных с ГИС-тематикой. Исключение представляют работы, затрагивающие решение частных задач. Например, в работах, ведущихся в институте проблем безопасного развития атомной энергетики, нейросети используются для решения задачи интерполяции [17]. Есть работы, посвященные использованию искусственных нейронных сетей в географических информационных системах для оценки устойчивости сельскохозяйственных земель [31]. Есть попытки описать взаимодействие систем искусственного интеллекта с ГИС [12]. Несмотря на то, что существует некоторое количество работ, в которых высказывается пожелание использования нейросетевых технологий [3, 6, 12, 13], общей методологии использования нейросетей в ГИС до сих пор не создано. Также нет общего описания и классификации задач, для которых возможно использование нейронных сетей. Предположительно, такое положение вызвано отсутствием удобного средства для решения задач ГИС нейронными сетями.

1.4.1 Выявление преимуществ и недостатков ИНС в сравнении с «классическими» схемами (постановка проблематики исследования)

Существующие проблемы при оценке обстановки традиционными алгоритмами:

- большое количество учитываемых факторов;
- скорость вычислений существующих алгоритмов;
- человеческий фактор.

Однако следующие аспекты архитектуры ИНС позволяют решить данные проблемы при оценке обстановки.

1. Решение задач при неизвестных закономерностях

Используя способность обучения на множестве примеров, нейронная сеть способная решать задачи, в которых неизвестны закономерности развития ситуации и зависимости между входными и выходными данными. Традиционные математические методы и экспертные системы в таких случаях пасуют.

2. Устойчивость к шумам во входных данных

Возможность работы при наличии большого числа неинформативных, шумовых входных сигналов. Нет необходимости делать их предварительный отсев, нейронная сеть сама определит их малопригодность для решения задачи и отбросит их.

3. Адаптирование к изменениям окружающей среды

Нейронные сети обладают способностью адаптироваться к изменениям окружающей среды. В частности, нейронные сети, обученные действовать в определенной среде, могут быть легко переучены для работы в условиях незначительных колебаний параметров среды. Более того, для работы в нестационарной среде (где статистика изменяется с течением времени) могут быть созданы нейронные сети, переучивающиеся в реальном времени. Чем выше адаптивные способности системы, тем более устойчивой будет ее работа в нестационарной среде. При этом следует заметить, что адаптивность не всегда ведет к устойчивости; иногда она приводит к совершенно противоположному результату. Например, адаптивная система с параметрами, быстро изменяющимися во времени, может также быстро реагировать и на посторонние возбуждения, что вызовет потерю производительности. Для того чтобы использовать все достоинства адаптивности, основные параметры системы должны быть достаточно стабильными, чтобы можно было не учитывать внешние помехи, и достаточно гибкими, чтобы обеспечить реакцию на существенные изменения среды.

4. Потенциальное сверхвысокое быстродействие

Нейронные сети обладают потенциальным сверхвысоким быстродействием за счет использования массового параллелизма обработки информации.

5. Отказоустойчивость при аппаратной реализации нейронной сети

Нейронные сети потенциально отказоустойчивы. Это значит, что при неблагоприятных условиях их производительность падает незначительно. Например, если поврежден какой-то нейрон или его связи, извлечение запомненной информации затрудняется. Однако, принимая в расчет распределенный характер хранения информации в нейронной сети, можно утверждать, что только серьезные повреждения структуры нейронной сети существенно повлияют на ее работоспособность. Поэтому снижение качества работы нейронной сети происходит медленно.

Однако стоит отметить и недостатки ИНС.

1. Длительное обучение нейронной сети. Как показывает практика, ИНС неработоспособны без обучения, причем точность их работы во многом зависит от обучающих наборов, а соответственно от алгоритма обучения и экспертов, составляющих данные наборы.

2. Неуниверсальность. ИНС справляются только с теми задачами, для которых создавались. Например, спроектированная под задачу оценки обстановки в ближней морской зоне нейронная сеть не способна анализировать сезонные миграции жаворонков.

Однако в решении узкоспециализированных задач преимущества нейронных сетей берут верх над недостатками и аппарат ИНС является тем самым инструментом, который позволит преодолеть возникшие проблемные моменты.

Выводы по главе 1

По результатам проведенного анализа требований к представлению обстановки и возможностей средств геоинформатики для ее отображения и моделирования, сделаны выводы:

1. Представление информации об обстановке должно предусматривать как планарную, так и трехмерную сущность географических объектов и явлений, а также обеспечить в составе атрибутивной части отражение динамических характеристики и принципиальные структурные взаимосвязи.

2. Атрибутивная составляющая, принадлежащая объектам должна быть минимальной по своей емкости, но достаточной для представления всех необходимых свойств объектов, для построения как географически зависимой модели объектов и явлений, так и организационной структуры взаимодействий и управления.

3. По совокупности удовлетворяемых требований с хорошим запасом лидирует отечественная ГИС «ИнГЕО», обеспечивающая самое высокое отношение «функциональность/стоимость». Из иностранных ГИС следует отметить ARCGis, как обладающую наибольшим функционалом.

4. Существующие ГИС отвечают большому числу существующих потребностей в геоинформационном обеспечении, являются многофункциональными, достаточно интегрированными между собой, но в то же время обладают разноэффективными инструментариями. Однако ни одна из существующих ГИС не может в полной мере быть взята в качестве полноценной ГИС для обеспечения безопасности мореплавания. Разработанное в работе техническое задание на разработку целевой ГИС может служить рабочим прототипом для ее реализации.

5. Для удовлетворения потребностей полнофункциональных систем освещения морской обстановки требуется разработка специальных геоинформационных моделей и методов представления и использования ГИ о территориальной ситуации.

6. ИНС как модельно-программный инструмент для проведения анализа массивов данных на основе обучающих алгоритмов и методик является перспективным инструментом обработки ГИ об обстановке в БМЗ.

7. Для обработки ГИ с помощью ИНС требуется определенное представление ГИ в виде, приемлемом для входов нейронной сети.

8. ИНС обладает свойством неуниверсальности, что не дает возможности применения одной и той же ИНС для различных задач.

Глава 2. Модель геоданных для представления обстановки в ближней морской зоне

2.1 Подходы к построению моделей представления обстановки (принципы геомоделирования)

Для приема-передачи информации от удаленных терминалов определяющим требованием является объем передаваемой информации и устойчивость к возможным искажениям или частичным потерям. Для чего предлагается передавать оперативную информацию в формализованном текстовом виде последовательного типа.

Формализованность информации об объектах обеспечивается ограниченным числом типов объектов, имеющих в своих описаниях, также формально определенные свойства.

Последовательность в передаваемых данных позволяет при пропуске части информации в процессе передачи не терять и не искажать информацию об объектах, следующих в описании далее.

В предлагаемой модели представления информации об объектах, с учетом особенностей задач, решаемых с применением ИНС в ГИС необходимо обеспечить следующие требования:

- объект может описываться двухмерном пространстве;
- объект может иметь динамические характеристики;
- объект может иметь несколько распределенных в пространстве характеристик;
- объект может иметь сложную форму как совокупность простых примитивов;
- объект имеет в характеристиках дополнительную описательную часть;
- объекты должны сопровождаться текстовой пояснительной информацией;
- объекты могут составлять иерархическую структуру логической подчиненности, т.е. быть не только обособленными элементами обстановки,

а входить в иерархию по какому-либо логическому признаку, например ведомственной принадлежности;

На удаленном терминале необходимо обеспечить формирование файлов формализованного текстового сообщения об объектах, передаваемых в удаленные ГИС на основании данных, вносимых соответствующими операторами. Данные в сообщения должны формироваться согласно единых правил с помощью специализированных оверлейных модулей в клиентской части ГИС. Данные об объектах, полученные из разнородных источников должны логически обрабатываться, исходя из логики получаемых данных. К примеру элементы движения судов могут быть получены по оперативным данным, данным радиолокации с объектов слежения, по данным разведки, по данным стационарных пеленгаторных станций и разнесенным по времени. Результатом решения задачи определения в любом из перечисленных случаях, для внесения в формализованное сообщение, должны стать данные о времени, курсе, скорости, глубине (высоте) и месте цели с учетом перевода размерностей данных.

Принимающие оверлейные программы ГИС должны в соответствии с полученными в файлах сообщений данными вносить объекты с описанными характеристиками в соответствующие базы данных для их последующей обработки прикладными задачами и визуализации в соответствующие слои ЭК. Учитывать и отображать изменение динамических характеристик объектов в соответствии с их описаниями в реальном времени. Иметь процедуры прогнозирования изменения динамических объектов на заданный интервал времени. Вести журнал учета поступающих сообщений и базу данных текущих объектов, входящих в перечень типовых[53].

ГИС должна поддерживать работу с дополнительных слоев для ЭК числом не менее 30, которые должны отображаться по указанию оператора. В число этих дополнительных слоев будут входит как тематические слои общего плана так дополнительные военные слои (уровни ВУ). В число обязательных дополнительных слоев (уровней) ЭК целесообразно внести следующие:

дополнительная гидрографическая информация, необходимая исключительно для морской навигации, в том числе оперативная информация от НАВТЕКС или ИНМАРСАТ; океанографическая климатологическая информация; метеорологическая климатологическая информация.

Информация о слое содержит: наименование слоя (текст, которым слой будет обозначен в меню слоев КИС); идентификатор слоя; наименование приложения (прикладной задачи) обеспечивающего формирование и формирование объектов слоя; тип приложения (прикладной задачи) (activeX, DLL); имя интерфейсного файла данных; перечень функций работы со слоем в целом (со всеми объектами слоя): перечень разрешенных оверлейных модулей по работе с объектом слоя (условным знаком или графическим примитивом); другие необходимые данные, перечень которых определяется в процессе разработки ГИС.

Логическая иерархия объектов осуществляется на основе применения семейства идентификаторов, которые используются в базе данных ГИС для построения такой иерархии[54].

2.2. Формальная модель геоданных для представления обстановки

2.2.1 Примитивы модели и семантика их описания

ГИС преодолевает основные недостатки обычных карт - их статичность и ограниченная емкость как носителя информации. В последние десятилетия бумажные карты из-за перегруженности информацией становятся нечитабельными. ГИС же обеспечивает управление визуализацией информации. Появляется возможность выводить (на экран, на твердую копию) только те объекты или их множества, которые интересуют нас в данный момент. Фактически осуществляется переход от сложных комплексных карт к серии взаимоувязанных частных карт. При этом улучшается структурированность информации, а следовательно, повышается эффективность ее обработки и анализа. В ГИС карта оживает и становится действительно динамическим объектом в смысле:

• изменяемости масштаба;

• преобразования картографических проекций;

• варьирования объектным составом карты;

• возможности опрашивать через карту в режиме реального времени многочисленные базы данных;

• изменения способа отображения объектов (цвет, тип линии и т.п.), в том числе и определения символогии через значения атрибутов, то есть синхронизации визуализации с изменениями в базах данных;

• легкости внесения любых изменений.

Рассмотрим основные понятия ГИС, в том или ином виде присутствующие во всех современных геоинформационных системах.

Данные

В ГИС данные делятся на две категории:

• пространственные (местоположение);

• непространственные (атрибуты).

Объекты

Пространственные данные включают географические объекты, представляемые:

• точками;

• линиями;

• полигонами.

Дугами описываются те реальные объекты, которые можно рассматривать как линии. Дуга состоит из отрезков линий и дуг окружностей.

Полигоны - замкнутые области, которые представляют однородные по некоторым критериям участки.

Атрибутивные данные могут включать идентификатор объекта, любую описательную информацию из баз данных, изображение и многое другое.

•Слой

Слои в карте подразделяются на два основных вида - растровые и векторные.

Векторные слои - это совокупность простых геометрических объектов (точка, дуга, полигон), которые представляют те или иные объекты на местности. Векторные слои могут также хранить топологию, т.е. информацию о взаимном расположении объектов.

Растровые слои представляют из себя сплошные изображения. Они не могут содержать объекты. Однако они могут служить фоном для векторных слоев

•Объект слоя

Каждому объекту векторного слоя может соответствовать запись в базе данных, чем обеспечивается привязка информации к местности. Это соответствие может обеспечиваться в частности назначением каждому объекту соответствующего идентификатора.

•Легенда карты

Легенда карты - свод условных обозначений, использованных на карте, с текстовыми пояснениями к ним. Обычно, легенды создаются на основе классификаций изображаемых объектов и явлений, они становятся их графической моделью и часто служат для построения классификаторов.

•Карта

Представляет собой набор географических слоев, каждый из которых привносит в карту информацию по какой-либо определенной теме. например, на слой границ некоторой территории может быть нанесен слой рек, затем слой, отображающий количество атмосферных осадков в процентном отношении и т.д.

Электронную карту в ГИС можно рассматривать как многокомпонентную модель реальности. Основными целями ее создания являются:

• графическая коммуникация пространственных отношений и распределений;

• улучшение возможности анализа, обработки и отображения геоинформационных данных;

• визуальное отображение цифровых моделей явлений, невидимых для человеческого глаза;

• автоматизация отображения и картографического анализа в системах управления; исследование объектов, явлений и процессов с учетом динамики их развития и возможного использования;

• получение аналитических решений в графическом виде в режимах реального и разделенного времени и т.д.

Основой визуального представления данных при помощи ГИС- технологий служит так называемая графическая среда. Основу графической среды и соответственно визуализации базы данных ГИС составляют векторные и растровые модели.

Рассматриваемая модель будет считаться векторной слаботопологизированной.

Большое количество графических данных в ГИС со специфическими взаимными связями требует топологического описания объектов и групп объектов, которое зависит от "связанности" (простой или сложной). Оно определяет совокупность топологических моделей.

Напомним, что топологические свойства фигур не изменяются при любых деформациях, производимых без разрывов или соединений. Топологически родственные фигуры: прямоугольный четырехугольник, замкнутый контур произвольной формы, окружность, треугольник. Эти объекты (фигуры) имеют одинаковую топологию - одинаковые топологические свойства. Другим примером топологически родственных фигур могут служить арифметические знак” сложения " + " и умножения " х ".

В геоинформационных системах применение термина топологический не такое строгое, как в топологии. В ГИС топологическая модель определяется наличием и хранением совокупностей взаимосвязей, таких, как соединенность дуг на пересечениях, упорядоченный набор звеньев (цепей), образующих границу каждого полигона, взаимосвязи смежности между ареалами и т.п.

В общем смысле слово топологический означает, что в модели объекта хранятся взаимосвязи, которые расширяют использование данных ГИС для различных видов пространственного анализа.

Топологическими характеристиками графические модели ГИС существенно отличаются от моделей САПР. Соответственно это различие просматривается в программно-технологическом обеспечении этих систем.

Например, вплоть до настоящего времени много разработок ГИС выполняется с использованием средств Автокада, версий от 10 до 13 Однако в нем не предусмотрены ни работа с покрытиями, ни оверлейные процедуры, ни обработка топологических данных. Принципиально такие операции в системах CAD (Computer-Aided Desing) возможны, но путем доработки программного обеспечения, что требует достаточно высокой квалификации пользователя и, естественно, ограничивает их круг.

В системах ГИС названные выше процедуры являются встроенными и делают доступным анализ картографической информации широкому кругу пользователей без всякой доработки.

Элементы топологии, входящие в описание моделей данных ГИС, в простейшем случае определяются связями между элементами основных типов координатных данных. Например, в логическую структуру описания данных могут входить указания о том, какие линии входят в район, в каких точках эти линии пересекаются.

Топологические модели позволяют представлять элементы карты и всю карту в целом в виде графов. Площади, линии и точки описываются границами и узлами (дуговая/узловая структура). Каждая граница идет от начального к конечному узлу, и известно, какие площади находятся слева и справа.

Теоретической основой моделей служат алгебраическая топология и теория графов. В соответствии с алгебраической топологией координатные типы данных: площади, линии и точки называются 2-ячейками, 1-ячейками и 0-ячейками соответственно. Карта рассматривается как ориентированный двухмерный ячеечный комплекс.

Двойственность между теорией графов и алгебраической топологией позволяет применять теоретические положения графов, а также топологический подход.

Топологическое векторное представление данных отличается от нетопологического наличием возможности получения исчерпывающего списка взаимоотношений между связанными геометрическими примитивами без изменения хранимых координат пространственных объектов.

Необходимая процедура при работе с топологической моделью - подготовка геометрических данных для построения топологии. Этот процесс не может быть полностью автоматизирован уже на данных средней сложности и реализуется только при дополнительных затратах труда (обычно значительных). Таким образом, данные, хранимые в системе, не предусматривающей поддержки топологии, не могут быть надежно преобразованы в топологические данные другой системы чисто автоматическим алгоритмом.

Топологические характеристики должны вычисляться в ходе количественных преобразований моделей объектов ГИС, а затем храниться в базе данных совместно с координатными данными.

Основные топологические характеристики моделей ГИС

Топологические модели в ГИС задаются совокупностью следующих характеристик:

• связанность векторов - контуры, дороги и прочие векторы должны храниться не как независимые наборы точек, а как взаимосвязанные друг с другом объекты;

• связанность и примыкание районов - информация о взаимном расположении районов и об узлах пересечения районов;

• пересечение - информация о типах пересечений позволяет воспроизводить мосты и дорожные пересечения. Так Т-образное пересечение (3 линии) является трехвалентным, а Х-образное (4 линии сходятся в точке пересечения) называют четырехвалентным;

• близость - показатель пространственной близости линейных или ареальных объектов, оценивается числовым параметром, в данном случае символом 5.

Топологические характеристики линейных объектов могут быть представлены визуально с помощью связанных графов. Граф сохраняет структуру модели со всеми узлами и пересечениями. Он напоминает карту с искаженным масштабом. Примером такого графа может служить схема метрополитена. Разница между картой метро и схемой метро показывает разницу между картой и графом.

Узлы графа, описывающего картографическую модель, соответствуют пересечениям дорог, местам смыкания дорог с мостами и т.п. Ребра такого графа описывают участки дорог и соединяющие их объекты. В отличие от классической сетевой модели в данной модели длина ребер может не нести информативной нагрузки.

Топологические характеристики ареальных объектов могут быть представлены с помощью графов покрытия и смежности. Граф покрытия топологически гомоморфен контурной карте соответствующих районов. Ребра такого графа описывают границы между районами, а его узлы (вершины) представляют точки смыкания районов. Степень вершины такого графа - это число районов, которые в ней смыкаются. Граф смежности это как бы вывернутый наизнанку граф покрытия. В нем районы отображаются узлами (вершинами), а пара смыкающихся районов ребрами. На основе такого графа ГИС может выдать ответ на вопрос, является ли проходимой рассматриваемая территория, разделенная напроходимые или непроходимые участки.

Топологические характеристики сопровождаются позиционной и описательной информацией. Вершина графа покрытия может быть дополнена координатными точками, в которых смыкаются соответствующие районы, а ребрам приписывают левосторонние и правосторонние идентификаторы.

После введения точечных объектов при построении линейных и площадных объектов необходимо "создать" топологию. Эти процессы включают вычисление и кодирование связей между точками, линиями и ареалами.

Пересечения и связи имеют векторное представление. Топологические характеристики заносятся при кодировании данных в виде дополнительных

атрибутов Этот процесс осуществляется автоматически во многих ГИС в ходе дигитализации (картографических или фотограмметрических) данных.

Объекты связаны множеством отношений между собой. Это определяет эффективность применения реляционных моделей и баз данных, в основе которых используется понятие *отношения.* В свою очередь, отношения задают множества связей. Простейшие примеры таких связей : "ближайший к .. ", "пересекает", "соединен с ...".

Каждому объекту можно присвоить признак, который представляет собой идентификатор ближайшего к нему объекта того же класса; таким образом кодируются связи между парами объектов.

В ГИС часто кодируются два особых типа связей: связи в сетях и связи между полигонами.

Топологически сети состоят из объектов двух типов: линий (звенья, грани, ребра, дуги) и узлов (вершины, пересечения, соединения).

Простейший способ кодирования связей между звеньями и узлами заключается в присвоении каждому звену двух дополнительных атрибутов - идентификаторов узлов на каждом конце (входной узел и выходной узел).

В этом случае при кодировании геометрических данных будут иметь место два типа записей:

1) координаты дуг*;*

2) атрибуты дуг - входной узел, выходной узел, длина, описательные характеристики.

Такая структура позволяет, перемещаясь от звена к звену, определять те из них, у которых перекрываются номера узлов.

Более сложная, но и более совершенная структура имеет список всех звеньев для каждого узла. Это может быть выполнено добавлением к первым двум записи третьего типа;

3) узел: (x, *y),* смежные дуги (со знаком "+" для входного угла и со знаком "-" - для выходного).

Чтобы избежать неудобств, связанных с хранением неодинакового количества идентификаторов дуг, используют два отдельных файла:

1) простой упорядоченный список, в котором файл узлов сжат до ряда идентификаторов дуг;

2) таблицу, в которой для каждого узла хранится информация о положении первой дуги списка.

Используемое в настоящее время математическое обеспечение ГИС почти исключительно основано на топологических моделях, дающих хорошее формализованное представление о пространственных соотношениях между основными объектами карты. Однако, если требуется установить более сложные соотношения, например, включение или порядок, нужны дополнительные средства.

2.2.2 Область применения модели

В данной модели рассматриваются карта ближней морской зоны, представленная в векторном виде. По сути рассматриваемая нами модель представляет географическую карту как совокупность территорий, с каждой из которых связана соответствующая таблица данных.

Базовым элементом модели будет являться территория, то есть некоторая структура, графически на изображении ограниченная ломанной линией и имеющая связанные числовые данные, являющиеся характеристиками данного объекта.

В частности, рассматривая конкретную задачу представления обстановки в ближней морской зоне, можно разбить рассматриваемую зона на территории (как правило, по однородности рассматриваемых парметров). При этом стоит учитывать, что чем мельче по площади данные территории, тем точнее получится оценка обстановки, однако будет затрачено больше ресурсов машинного времени на обработку. В качестве данных, связанных с данной территорией можно взять конкретные данные, по которым будет производиться оценка, например, величину ледяного покрова, ресурсы данной территории либо иные числовые данные, связанные с рассматриваемой местностью.

Данная модель не учитывает особенностей рельефа и реального масштаба в рассматриваемой зоне. Но сделанные упрощения позволяют оптимизировать формат хранения данных для их дальнейшего использования с помощью искусственных нейронных сетей. Основным массивом данных в нашем случае являются таблицы числовых данных, связанные с картой, а не сама карта. Пример подобного представления приведен на рисунке. Как видим, данный рисунок представляет собой обзорную карту ледовой обстановки в Карском море. Данная карта-схема разделена на зоны, соответствующие различной величине ледового покрова и сохраняет приблизительный масштаб, однако не учитывает особенностей рельефа и не отражает наличие различных географических объектов, кроме разделения суша/море (рис. 11).

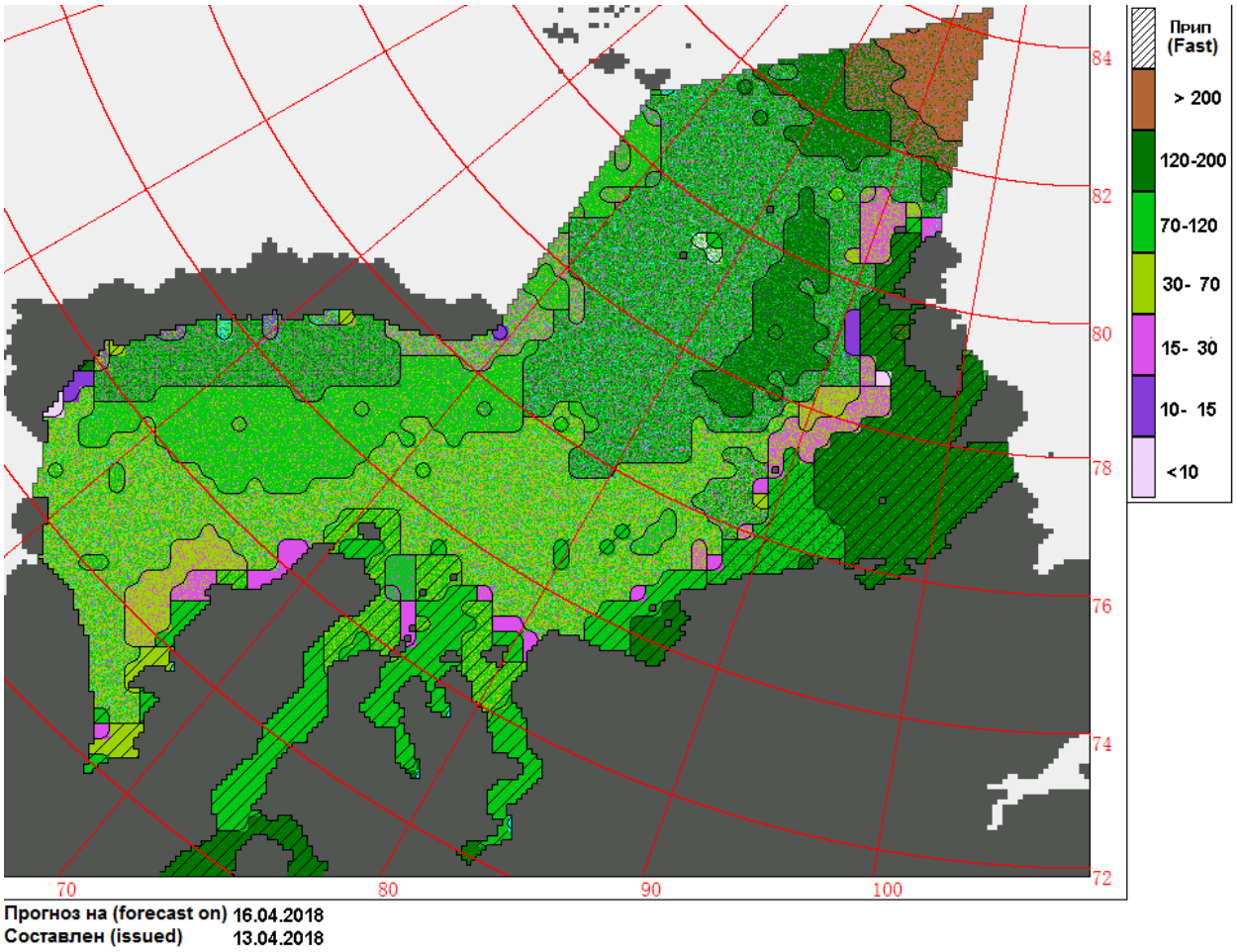

Рис 11. Обзорная карта ледовой обстановки в Карском море на 16.04.18

От обычной карты-схемы модель отличается наличием прикрепленных файлов с цифровыми данными, связанных с представленными на схеме

участками. Данные файлы позволяют нейронным сетям проводить анализ не только графически, распознавая изображения, но и используя математические данные.

2.2.3 Параметры модели

С учетом предложенной модели можно разделить параметры оценки окружающей обстановки на следующие группы (рис. 12-15):

- Навигационно-путевые параметры (плотность и толщина льда, течения, глубины, погодные условия);
- Антропогенные параметры (зоны, опасные для прохождения, например, из-за проводимых учений, радиационные могильники))
- Метеорологические параметры
- Экологические параметры (нахождение на маршруте краснокнижных животных, сезонное поведение животных)
- Специфические параметры (зависят от цели оценки обстановки).

В первую группу включаются совокупности величин, характеризующих место судна в море и его перемещение в заданной системе координат. Значение навигационного параметра, снятое со шкалы измерительного прибора, называют измеренным навигационным параметром. Каждому исправленному навигационному параметру на земной поверхности соответствует определенная навигационная изолиния.

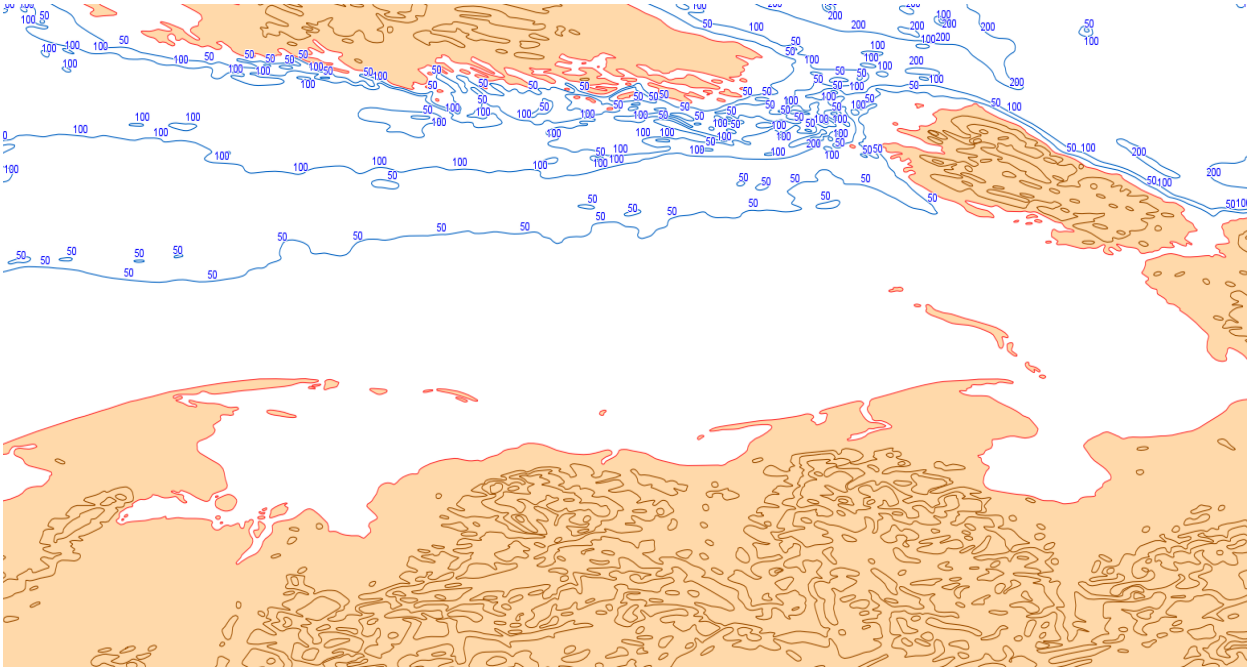

Рис 12. Параметры поля рельефа дна (глубины) как элемент обстановк

Навигационной изолинией называется такая линия на земной поверхности, каждая точка которой соответствует одному и тому же значению исправленного навигационного параметра. В зависимости от характера навигационного параметра (значение пеленга на ориентир, величина расстояния до ориентира, значение горизонтального угла между ориентирами и т.д.) им соответствующие навигационные изолинии имеют различный вид. И, кроме того, каждому значению навигационного параметра соответствует своя навигационная изолиния. Фактическое место судна всегда находится на навигационной изолинии в момент измерения соответствующего ей навигационного параметра и становится вполне очевидным то, что для определения места судна одной навигационной изолинии недостаточно, а значит: для определения места судна необходимо иметь не менее двух пересекающихся навигационных изолиний, причем угол их пересечения должен быть более 30° (оптимальный вариант –90°).

Данная группа параметров имеет наибольшее значение в нашей модели, поэтому весовые коэффициенты, соответствующие параметрам будут повышены.

К антропогенным факторам мы отнесем все, зависящее от людей в данной морской зоне, а именно – зоны прохождения других судов, проведение учений в

некоторых квадратах. Кроме того, к данному параметру мы отнесем захоронения опасных веществ (например, ядерных отходов) на дне.

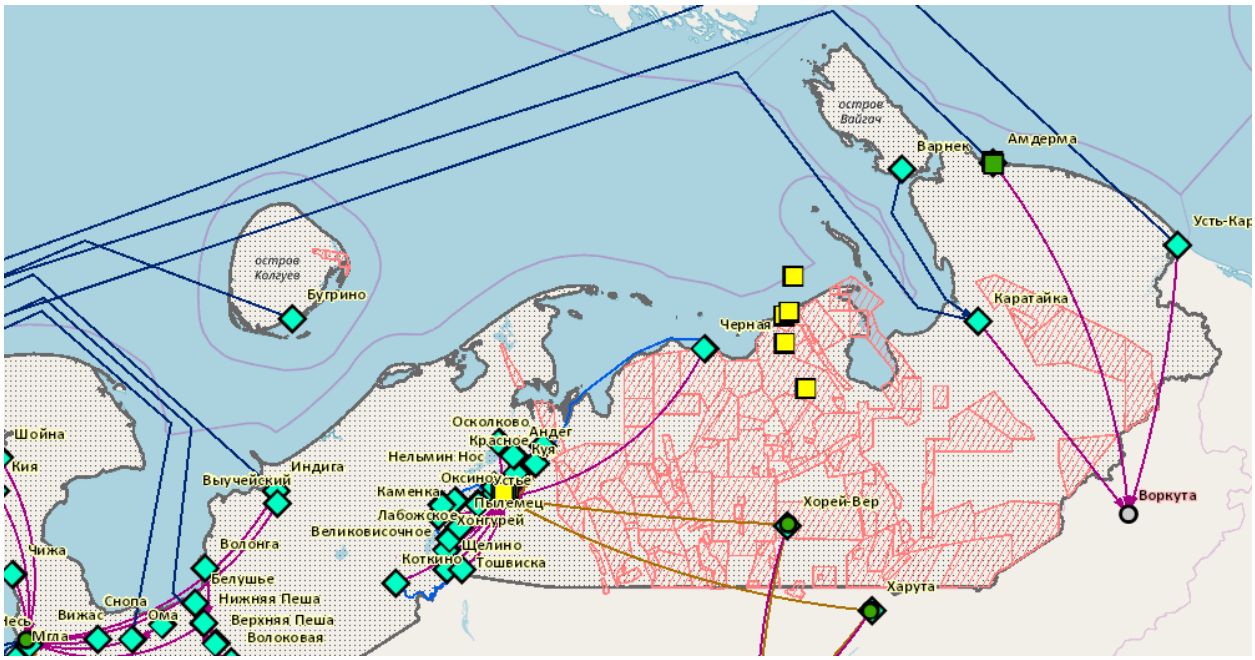

Рис. 13. Экологические параметры - утилизация отходов

Количество данных параметров в оценке меньше, чем навигационных, однако их важность зачастую критична. Поэтому данной группе будут выставлены самые высокие весовые коэффициенты.

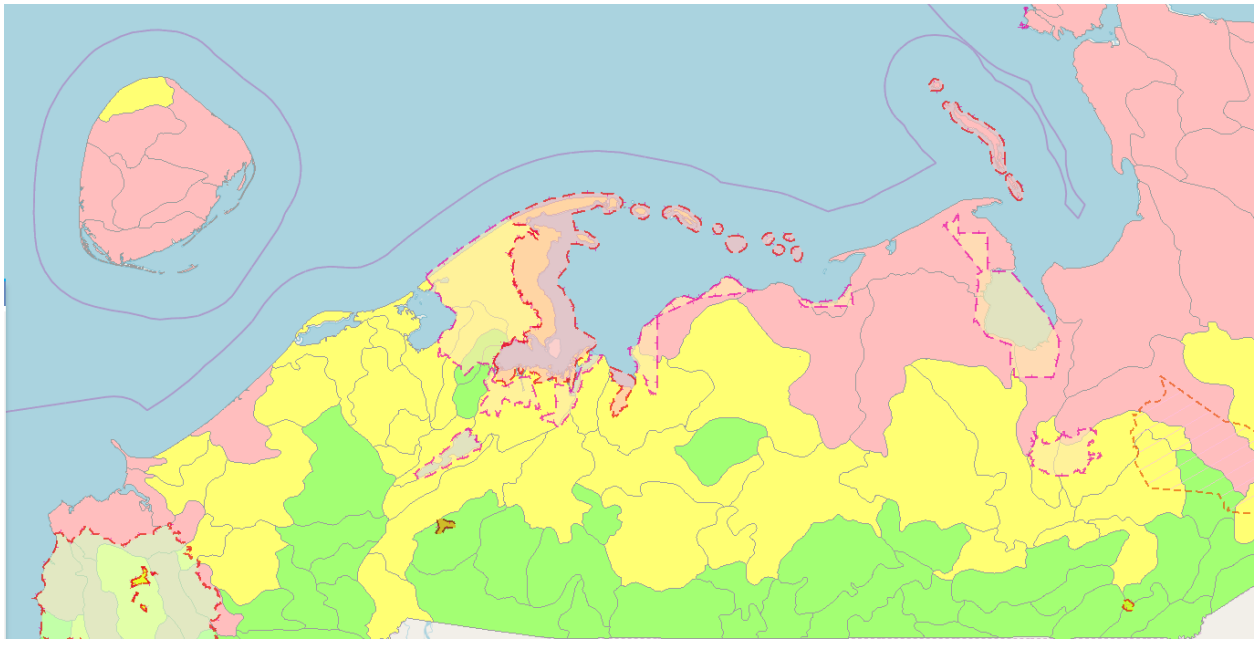

Рис.14. Биологическое разнообразие

Погодные параметры представляют собой цифровые оценки погоды в данном регионе, силу ветра, наличие осадков итд. Данные параметры имеют высокие весовые коэффициенты, ввиду своей важности.

Экологические факторы включают в себя ареалы обитания различных редких животных, зоны нереста рыб и прочие подобные параметры. Данные факторы достаточно важны. Но не представляют решающего значения в оценке, поэтому имеют низкие весовые коэффициенты.

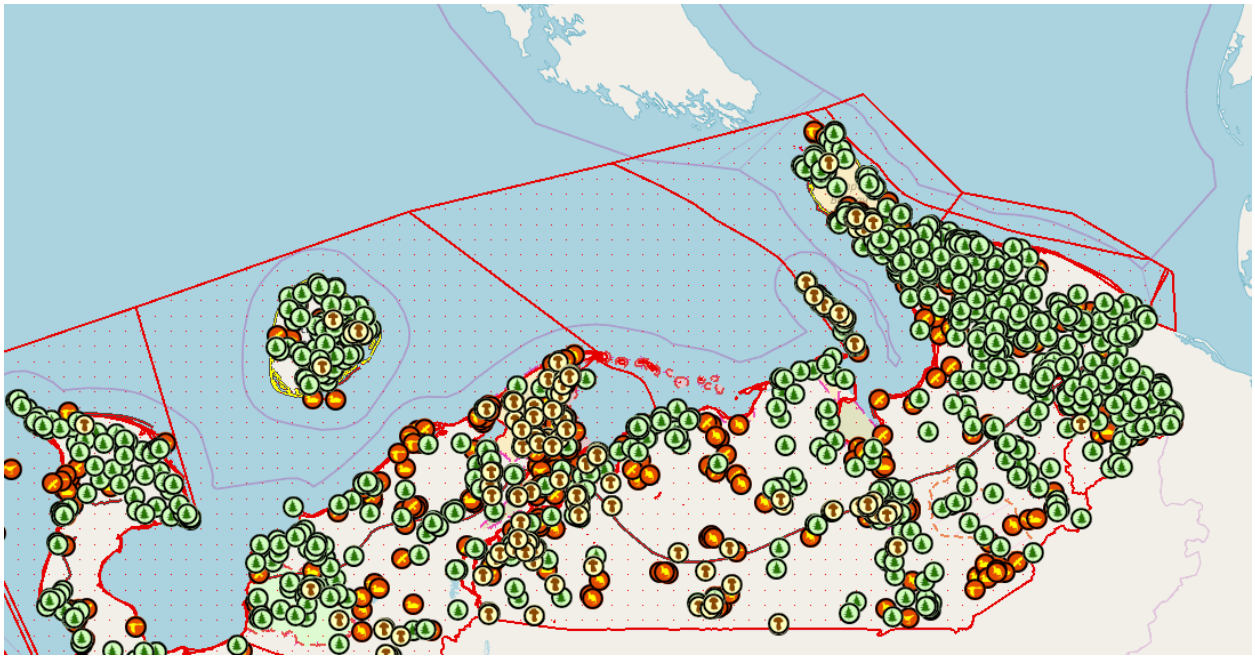

Рис.15. Распределение краснокнижных объектов

К специфическим параметрам мы отнесем все особые параметры, зависящие от цели нашей оценки. К ним можно отнести наличие/отсутствие на маршруте каких-либо объектов.

Данные параметры обладают своими индивидуальными весовыми коэффициентами, зависящими от проводимых исследований.

Стоит также заметить, что при рассмотрении обстановки рассматриваемые факторы могут играть как положительную, так и отрицательную роль.

Кроме того некоторые факторы будут целевыми, то есть их приоритет будет искусственно завышен. Отметим, что такое завышение требует переобучения нейронной сети с учетом расставляемых приоритетов. Целевые факторы не всегда являются положительными, например, если нужно испытать судно в штормовых условиях в качестве целевых факторов могут быть заданы соответствующие погодные условия, не являющимися положительным фактором при иных обстоятельствах.

Таким образом, на вход нейронной сети подаются вектора, состоящие из данных параметров. Стоит отметить, что в модели используются числовые параметры, однако нейронная сеть способна работать с входными данными в диапазоне от 0 до 1, потому предварительно стоит проводить первичное преобразование данных, путем деления их на 10^k где k-размерность каждого параметра.

В результате работы нейронной сети получается число в диапазоне от 0 до 1. В связи с этим оценка будет представлять собой число от 0 до 10, полученное округлением результата работы НС до десятых с последующим умножением на 10. Существует возможность сделать более точную шкалу оценивания, однако это выходит за границы поставленной задачи и существенно усложняет работу программы.

Будем считать, что чем больше оценка, тем хуже ситуация в рассматриваемой части морской зоны. Предлагается следующее разбиение шкалы:

- 10 - критическое состояние – прохождение судна в данной зоне невозможно;
- от 6 до 9 – плохое состояние – прохождение данной зоны затруднено;
- от 3 до 5 – удовлетворительное состояние – при прохождении данной зоны присутствуют как затрудняющие движение факторы, так и некоторое количество положительных/целевых;
- от 1 до 3 – оптимальное состояние - при прохождении данной зоны отсутствуют существенные затрудняющие движение факторы
- от 0 до 1 – отличное состояние – в данной зоне присутствуют только положительные и целевые факторы.

От выставленной оценки будет зависеть дальнейший маршрут прохождения судна.

2.2.4 Формальное представление примитивов и процессов в ближней морской зоне

Модель представляет собой совокупность графического отображения территорий и векторов параметров и оценок, соответствующих каждой территории

$M=ГО+(N_1…N_s,A_1…A_k, P_1…P_q,E_1…E_r, S_1…S_l, O)$.

Каждой территории присваивается вектор данных, полученный при помощи объединения данных со всех картоидов.

Графическая составляющая карты образуется отображением параметра оценок в размерном (возможно, цветовом) виде (рис. 16).

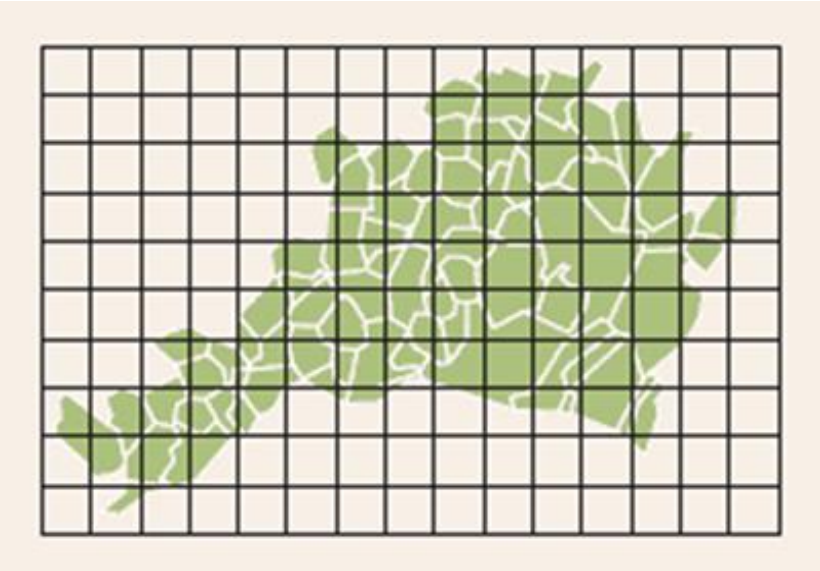

Рис. 16. Одноцветное представление модели с нанесенной координатной сеткой.

2.2.5 Формат и особенности файла модели представления обстановки

Shape-файл — векторный формат для хранения объектов, описываемых геометрией и сопутствующими атрибутами. В формате отсутствует возможность хранения топологической информации. Формат был представлен для ArcView GIS версии 2 в начале 90-х.

Принято использовать термин shape-файла, однако shape-файла это не один файл, а набор файлов с одинаковым именем, но разными расширениями. Основой формата являются три обязательных файла: .shp, .shx и .dbf. Кратко рассмотрим составляющие shape-файла.

•.shp. Главный файл .shp содержит информацию о геометрических объектах. Файл состоит из заголовка фиксированной длины и одной или более

записей переменной длины. Каждая запись переменной длины включает в себя заголовок записи и содержимое. Полное описание формата файла дано в документации по Esri Shapefile.

●.dbf. Файл, в котором записывается атрибутивная информация геометрических объектов, описанных файле .shp. Представляет собой базу данных в формате dBase II.

●.shx. Файл связи между файлами .dbf и .shp. В технической документации его называют индексным файлом (хоть он таковым не является).

●.sbn и .sbx. Файлы пространственных индексов. Ускоряют операции над геометрическими объектами. Формируются автоматически и могут быть удалены без потерь данных (при этом отключается пространственное индексирование).

●.aih и .ain. Индексные файлы атрибутивных таблиц. Формируются автоматически и могут быть удалены без потерь данных (при этом отключается индексирование в атрибутивных таблицах).

В настоящее время shape-файл может хранить типы геометрических объектов, представленные в табл. 2

Таблица 2

Геометрические объекты

Point	Точка
PolyLine	Полилиния — объект, состоящий из нескольких линий (ломаных), которые могут соприкасаться и пересекаться
Polygon	Полигон (может состоять из нескольких частей с пустотами)
MultiPoint	Мультиточка — объект, состоящий из нескольких точек
PointZ	Точка в 3-хмерном пространстве (XYZ)
PolyLineZ	Полилиния в 3-хмерном пространстве
PolygonZ	Полигон в 3-хмерном пространстве
MultiPointZ	Мультиточка в 3-хмерном пространстве
PointM	Точка с каким-либо измеренным значением

PolyLineM	Полилиния с какими-либо измеренными значениями
PolygonM	Полигон с какими-либо измеренными значениями
MultiPointM	Мультиточка с какими-либо измеренными значениями
MultiPatch	Триангуляционные поверхности

Shape-файлы в классическом виде представляют собой нетопологизированные модели. Для работы с нашими моделями необходимо задание координат.

Информация о системе координат пространственных объектов шейп-файла часто отсутствует. В этом случае, параметр Spatial Reference в поле Shape будет иметь значение Unknown или Assumed Geographic (предположительно географическая). Если значения ограничивающих координат пространственных объектов лежат в пределах от -180 до 180 по x и от -90 до 90 по y, то система ArcGIS предполагает, что данные записаны в географической системе координат с датумом NAD27. Если ограничивающие координаты имеют другие значения, программа считает пространственную привязку неизвестной.

Если система координат шейп-файла не задана, с ним все равно можно работать, но не все функциональные возможности будут доступны. Например, данные из такого шейп-файла не будут правильно взаимодействовать с другими данными, а его автоматически создаваемые метаданные будут неполными.

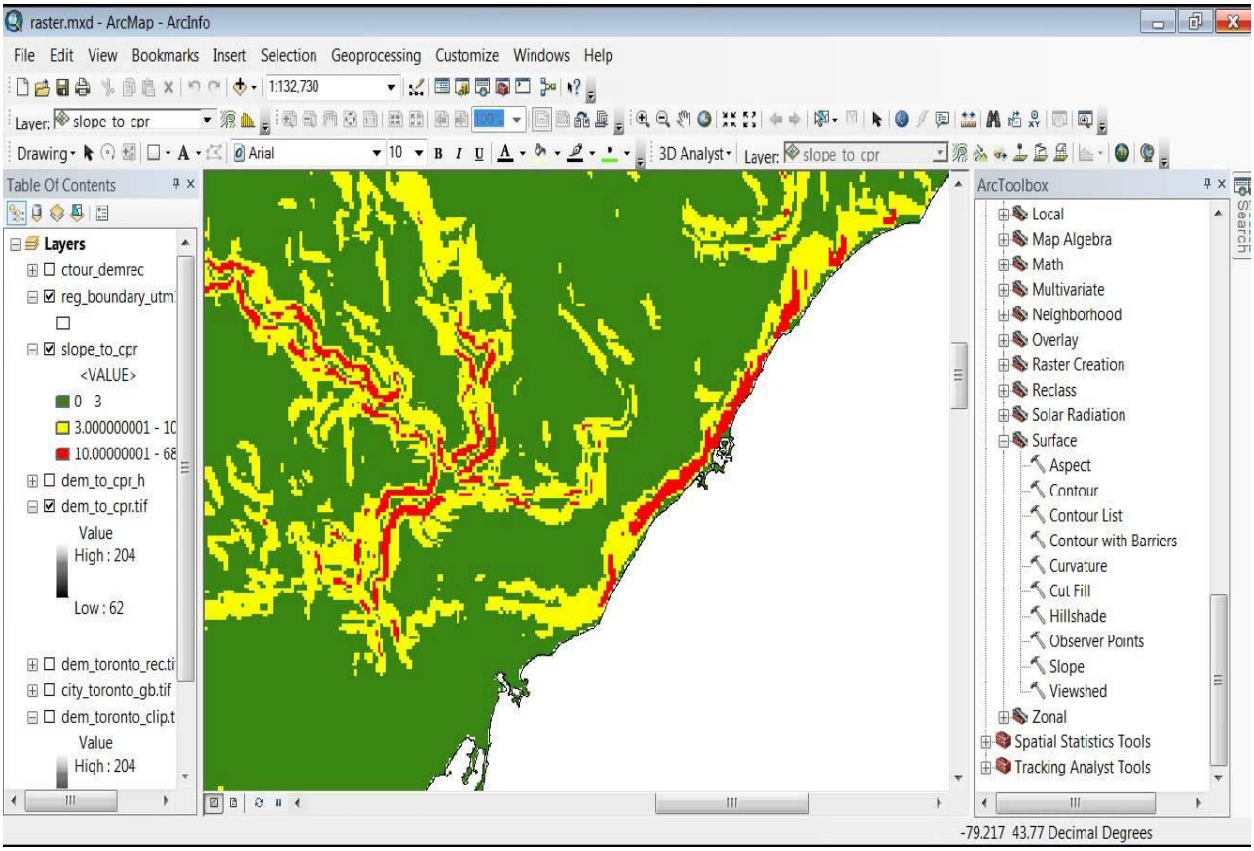

Рис. 17. Интерфейс ArcGIS для задания системы координат

Систему координат шейп-файла можно задать в ArcCatalog следующими способами:

- Выбрать одну из готовых систем координат, имеющихся в ArcCatalog.
- Импортировать параметры системы координат, используемой другим источником данных.
- Задать новую пользовательскую систему координат.

Параметры системы координат должны быть сохранены в файле с расширением .prj в той же папке, что и сам шейп-файл. В имени файла .prj должен содержаться тот же префикс, что и в имени самого шейп-файла. Например, если рабочий шейп-файл называется wells.shp, то параметры его системы координат должны храниться в той же папке в файле wells.prj.

После задания системы координат можно изменить ее отдельные параметры. Например, может потребоваться изменить один из параметров импортированной системы координат или настроить заранее определенную

систему координат. После создания собственной, пользовательской системы координат, можно сохранить ее в отдельном файле – она может понадобиться другим пользователям (рис. 17).

Выводы по главе 2

1. Разработана модель геосреды (обстановки), отличающаяся топологическим переходом от географически конкретного представления территориальной ситуации к пространственно-абстарктному картоиду, что позволяет формировать наборы исходных геоданных для работы (обучения) ИНС.
2. Предложена структура файлов модели, ориентированная на использование искусственных нейронных сетей, работающих в основном со связанными файлами, а не графической составляющей карты и менее требовательная к машинным ресурсам и более компактная, нежели традиционные карты и картоиды [109].
3. Предложенная модель оптимизирована для анализа и обработки динамических данных, что делает ее более удобной в ситуациях, когда необходимо срочное принятие решений.
4. Выявлены ограничения для применения данной модели. Они заключаются в особенности выбранного формата файлов. Шейп-файлы имеют ряд проблем, связанных с хранением атрибутов. Они не могут хранить значения NULL, округлять числа, имеют ограниченную поддержку символов Unicode, не могут хранить поля, имена которых длиннее 10 знаков, и не могут хранить дату и время в одном поле. Кроме того, они не поддерживают возможности, присутствующие в базах геоданных, например, работу с доменами и подтипами.
5. Выявлено, что в рассматриваемых задачах масштаб и детализация карт играют незначительную роль, потому выбран векторный формат представления данных. Таким образом при использовании данной модели нельзя говорить о масштабе и расположение объектов на карте также не отражает реальной действительности.

6. Модель обстановки служит формально-логической основой для разработки методики оценки территориальной ситуации в регионе (БМЗ).

Глава 3. Методика оценки окружающей обстановки с применением ИНС

На вход методике подаются геоданные, агрегированные в пространственной модели обстановки территориальной обстановки.

Общая схема работы методики представлена на рис. 18.

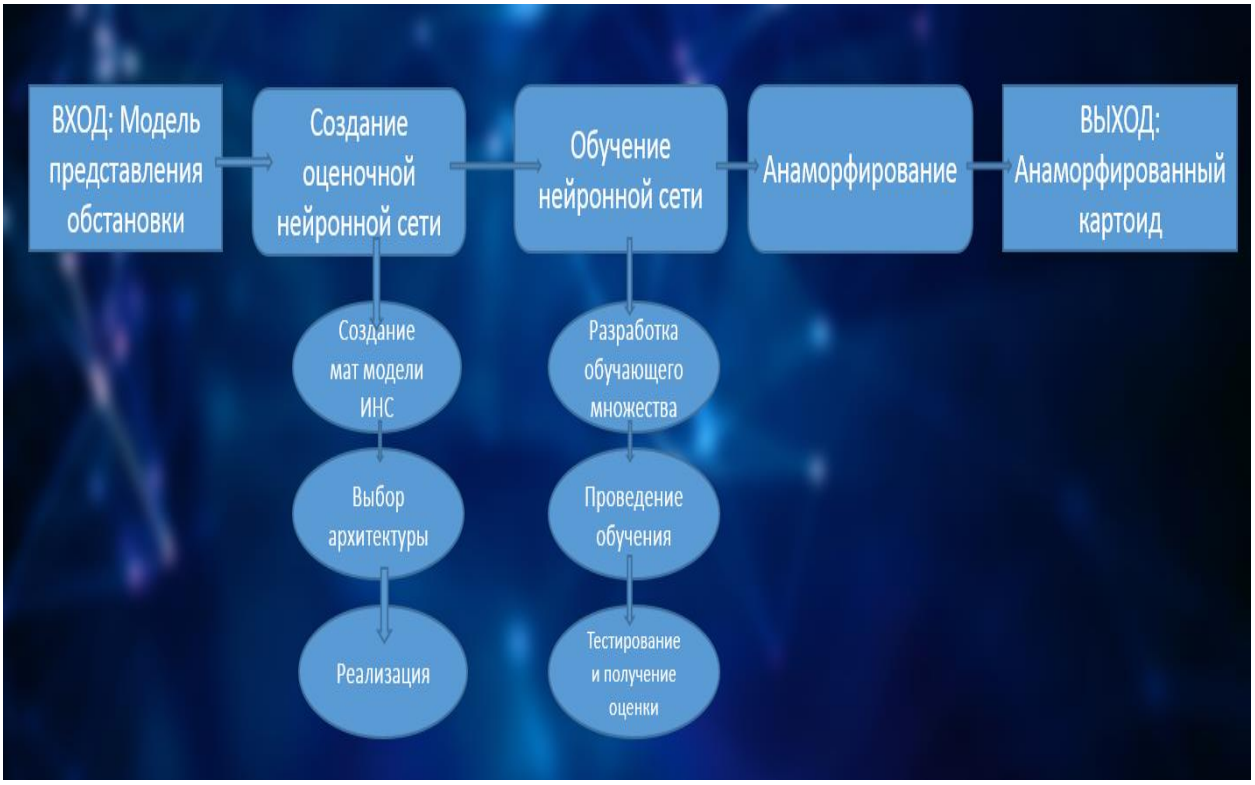

Рис 18. Блок-схема методики оценки обстановки в ближней морской зоне

На первом этапе путем систематизации данных формируется модель представления обстановки в БМЗ, описанная в главе 2. Затем происходит создание нейронной сети – ввиду того, что ИНС обладают свойством неуниверсальности необходимо проводить процедуры выбора архитектуры для каждой новой задач. Для каждой задачи необходимо подобрать несколько типов архитектур, провести тестирование в условиях конкретной задачи. Соответственно возникает необходимость обучения/переобучения вновь созданной нейронной сети. Затем, для удобства дальнейшего использования и вывода графического представления происходит процедура анаморфирования. Данный этап необходим для осуществления контроля человеком работы нейронной сети, в случае когда присутствует дополнительный этап контроля.

3.1 Выбор модели ИНС для процедуры оценки территориальной обстановки

Первым этапом работы рассматриваемой методики является выбор архитектуры нейронной сети. Основными архитектурами, с которыми возможна работа с геоданными являются многослойный перцептрон, рекуррентная нейронная сеть и сверточная нейронная сеть. Необходимо рассмотреть все предложенные архитектуры применительно к поставленной задаче оценки обстановки в ближней морской зоне. На вход ИНС подаются данные, описанные в пп. 2.3 главы 2. Построение, обучение и сравнение архитектур производилось в программном пакете STATISTIKA 8.0, что позволило автоматизировать процедуры сравнения и существенно сократить время обучения тестируемых ИНС.

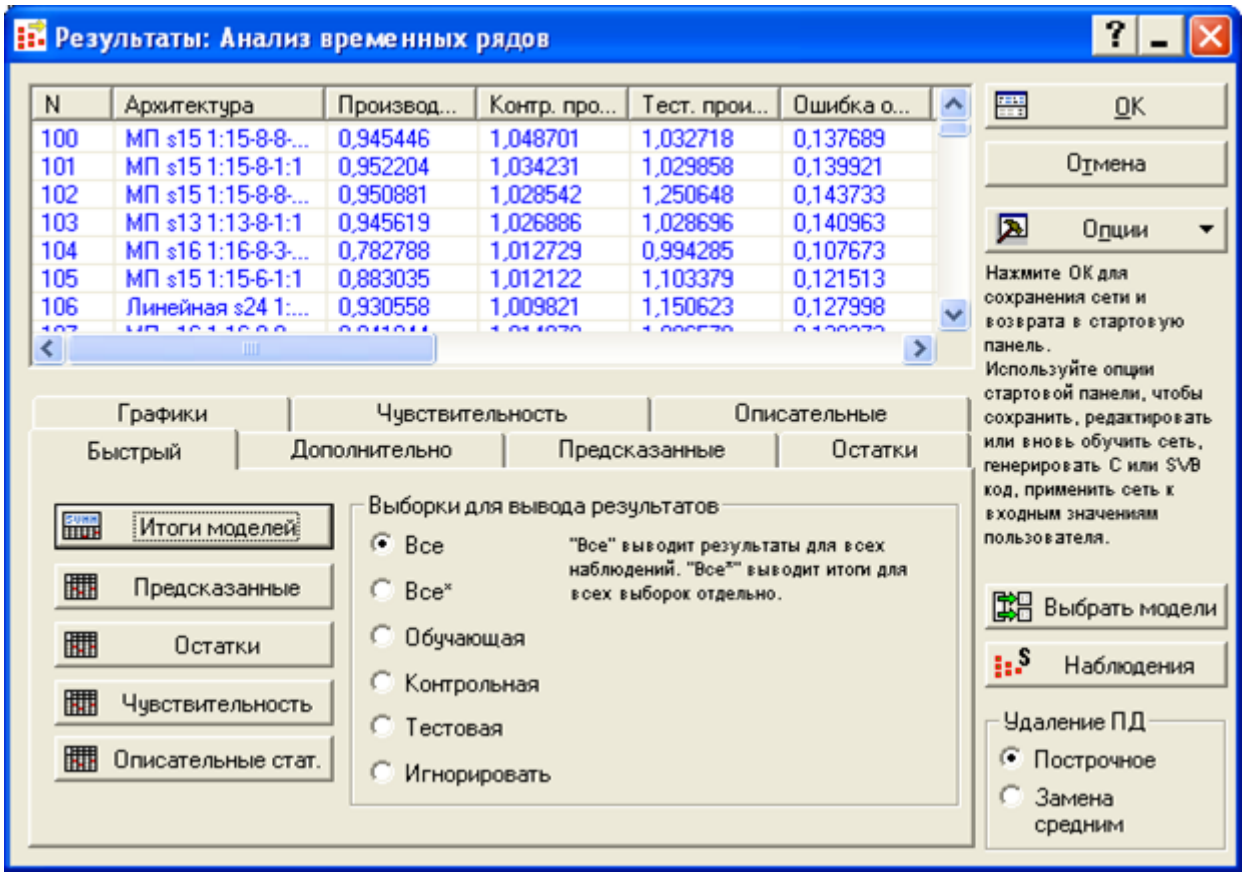

Рис.19 . Результаты массового обучения различных архитектур нейронных сетей

3.1.1 Многослойный перцептрон

В качестве первого варианта архитектуры нейронной сети выберем многослойный перцептрон. С учетом того, что перцептрон является

классическим решением для задач классификации, данная архитектура является одной из самых очевидных для решения поставленной задачи. Также, перцептрон - это система, состоящая из нейронов трех типов: сенсоры, ассоциативные и реагирующие - на вход, то есть на сенсоры подаем вектор, состоящий из параметров, описанных в главе 3.1: навигационных, антропогенных, экологических и специфических параметров. Стоит заметит, что таких параметров может быть различное количество, в зависимости от цели проводимой оценки, поэтому обозначим число выходов N.

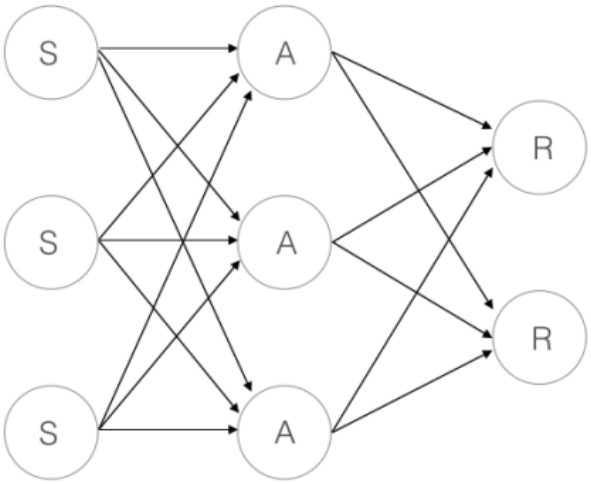

Рис.20. Архитектура многослойного перецептрона прямого распространения

С учетом поставленной задачи нам необходим один выходной нейрон, который, как известно, принимает значения от 0 до 1.

В качестве функции активации нейрона будем использовать сигмовидную функцию $S(x) = \frac{1}{1+\exp(-x)}$.

Данная функция позволяет усиливать слабые сигналы, что делает ее предпочтительной для решения конкретной задачи. Алгоритм обучения нейронной сети следующий:

1. Инициализировать синаптические веса случайными значениями, меньшими некоторого заранее заданного достаточно малого значения.
2. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.

3. Вычислить выход сети.

4. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).

5. Осуществить коррекцию весов сети для минимизации ошибки. Стоит отметить, что в конкретной задаче будем рассматривать корректировку весов нейронов скрытых слоев, так как корректировка выходного слоя не несет практической пользы.

6. Повторить шаги с 2 по 5 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

В качестве базовой архитектуры возьмем два скрытых слоя, состоящих из N нейронов. После обучения нейронной сети достаточным количеством обучающих наборов (см. Приложение), получаем значение ошибки, близкое к заранее заданному. Для получения большей точности к архитектуре нейронной сети были последовательно добавлены еще три скрытых слоя по N нейронов. После сравнения полученных значений ошибки было выявлено, что добавление скрытых слоев начиная с пятого не имеет практического смысла, ввиду незначительного улучшения значений точности при существенном росте затрат машинных мощностей и времени на работу нейронной сети. Следующим рассмотренным вариантом улучшения нейронной сети стало увеличение нейронов в скрытых слоях. Однако, как и в случае со слоями, увеличение не несло существенных преимуществ.

Таким образом, можно сделать вывод, что для решения нашей задачи использовалась нейронная сеть типа многослойный перцептрон, с 4 скрытыми слоями по N нейронов в каждом, N входными нейронами и 1 выходным нейроном.

3.1.2 Рекуррентная нейронная сеть

В качестве следующей архитектуры нейронной сети так же выберем многослойный перцептрон, однако теперь связи между нейронами способны работать в обе стороны. Принципиальным отличием от обычной нейронной сети прямого распространения является наличие так называемых обратных связей,

что дает повышенную точность в реализации, при этом усложняя структуру нейронной сети.

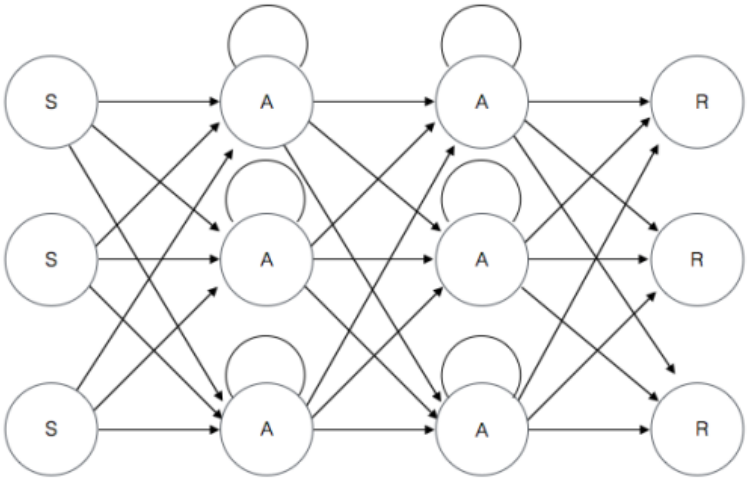

Рис.21. Рекуррентная нейронная сеть

В качестве функции активации нейрона также будем использовать сигмовидную функцию $S(x) = \frac{1}{1+\exp(-x)}$.

Данная функция кроме описанного выше позволяет существенно сократить вычислительную сложность метода обратного распространения ошибки, который будет использован для обучения нейронной сети.

Для обучения нейронной сети будем использовать алгоритм обратного распространения ошибки, данный алгоритм оптимален для задачи классификации с применением нейронной сети.

Стоит также отметить, что данная нейронная сеть нуждается в длительном обучении на большем количестве обучающих наборов, нежели предыдущий рассмотренный вариант нейронной сети. Кратко алгоритм обучения сети в данном случае выглядит следующим образом:

1. Прямой проход сети
2. Вычисление ошибки выходного элемента
3. Расчёт величины корректировки весов связей
5. Определение ошибки элементов первого скрытого слоя
6. Корректировка веса связей

7. Определение величины корректировки оставшихся весов. Завершение обратного прохода сети.

В качестве базовой архитектуры возьмем один скрытый слой, состоящий из N нейронов. После обучения нейронной сети достаточным количеством обучающих наборов (см. Приложение), получаем значение ошибки, близкое к заранее заданному. Для получения большей точности к архитектуре нейронной сети были последовательно добавлены еще два скрытых слоя по N нейронов. После сравнения полученных значений ошибки было выявлено, что добавление скрытых слоев начиная с третьего не имеет практического смысла, ввиду незначительного улучшения значений точности при существенном росте затрат машинных мощностей и времени на работу нейронной сети. Следующим рассмотренным вариантом улучшения нейронной сети стало увеличение нейронов в скрытых слоях. Однако, как и в случае со слоями, увеличение не несло существенных преимуществ.

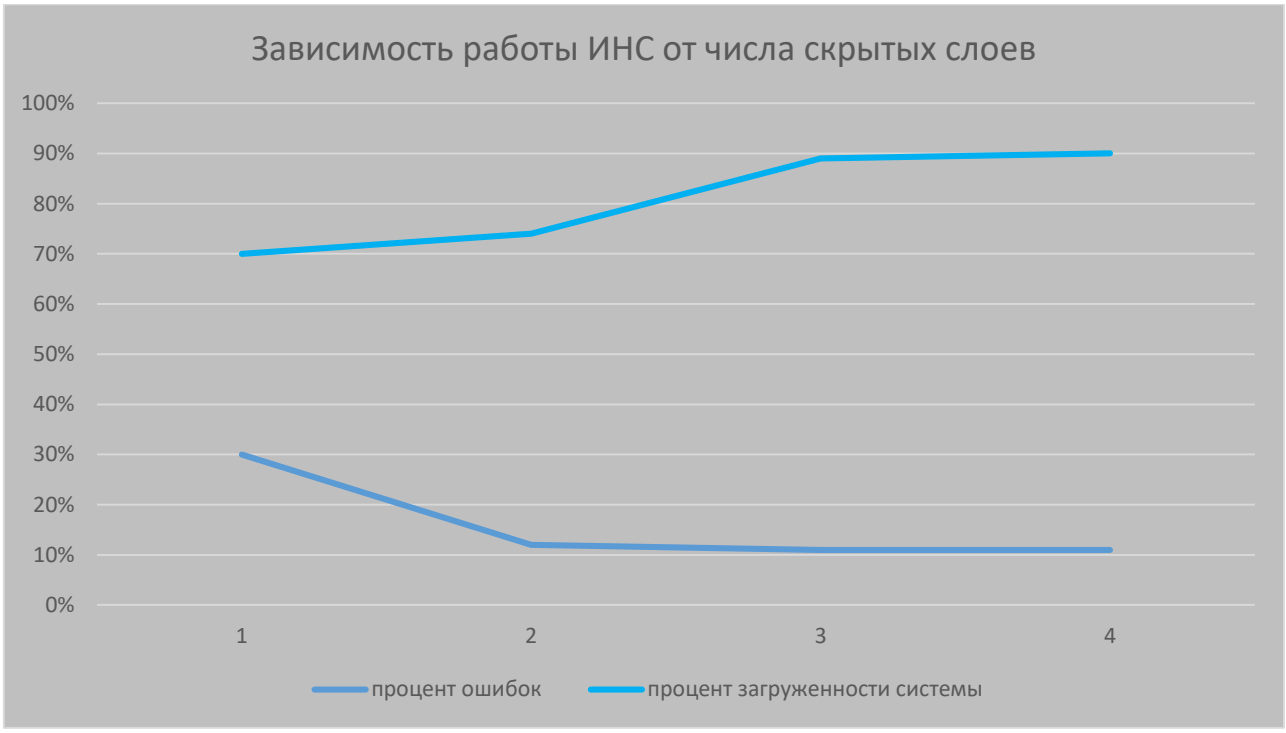

Рис.22. Сравнение точности работы и нагрузки на систему ИНС при увеличении числа скрытых слоев

По результатам рассмотрения данной модели видно, что для решения задачи использовалась нейронная сеть типа многослойный перцептрон, с 2 скрытыми

слоями по N нейронов в каждом, N входными нейронами и 1 выходным нейроном.

3.1.3 Свёрточная нейронная сеть

Рассмотрим архитектуру сверточной нейронной сети. Эта архитектура обрабатывает данные не целиком, а фрагментами, но при этом данные не дробятся на части, а осуществляется своего рода последовательный прогон. Затем данные передаются дальше по слоям. Кроме свёрточных слоёв используются также слои объединения. Слои объединения сжимаются с глубиной (обычно степенью двойки). К конечным слоям добавляются несколько персептронов (сеть прямого распространения), для последующей обработки данных.

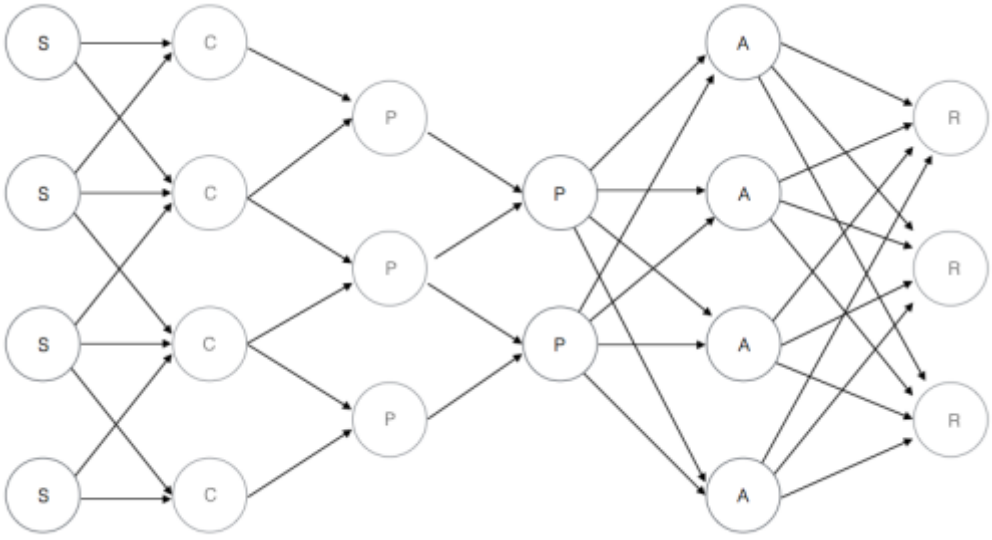

Рис.23. Сверточная нейронная сеть

Однако в процессе реализации данной нейронной сети возникли проблемы технического характера, связанные с форматом входных данных и используемой моделью, что не позволило реализовать данную сеть и адекватно оценить возможности данной архитектуры.

Стоит отметить, что при использовании карты в формате графического изображения данная сеть показывает хорошие результаты, однако рассмотрение подобной модели выходит за пределы данной работы.

3.1.4 Сравнение предложенных архитектур нейронных сетей

Сравнивая предложенные архитектуры можно сделать вывод, что для поставленной задачи оптимальным вариантом будет рекуррентная нейронная сеть, описанная в 3.1.2. Данная нейронная сеть после обучения позволяет получить процент ошибок меньший в сравнении со сверточной нейронной сетью и многослойным перцептроном (графики сравнения приведены на рисунке), кроме того обладает более простой реализацией нежели сверточная сеть (рис. 24).

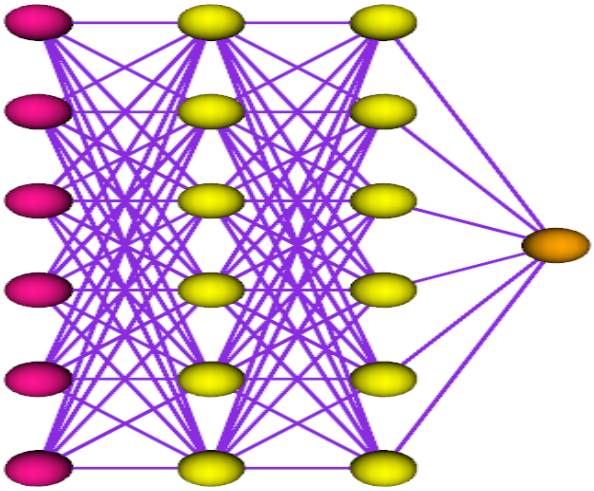

Рис.24. Итоговая архитектура предлагаемой ИНС

Данная архитектура предполагает более оптимальный метод обучения нейронной сети. В качестве недостатка будет выступать большее количество обучающих наборов и большее время обучения, что компенсируется большей точностью вычислений (рис. 25).

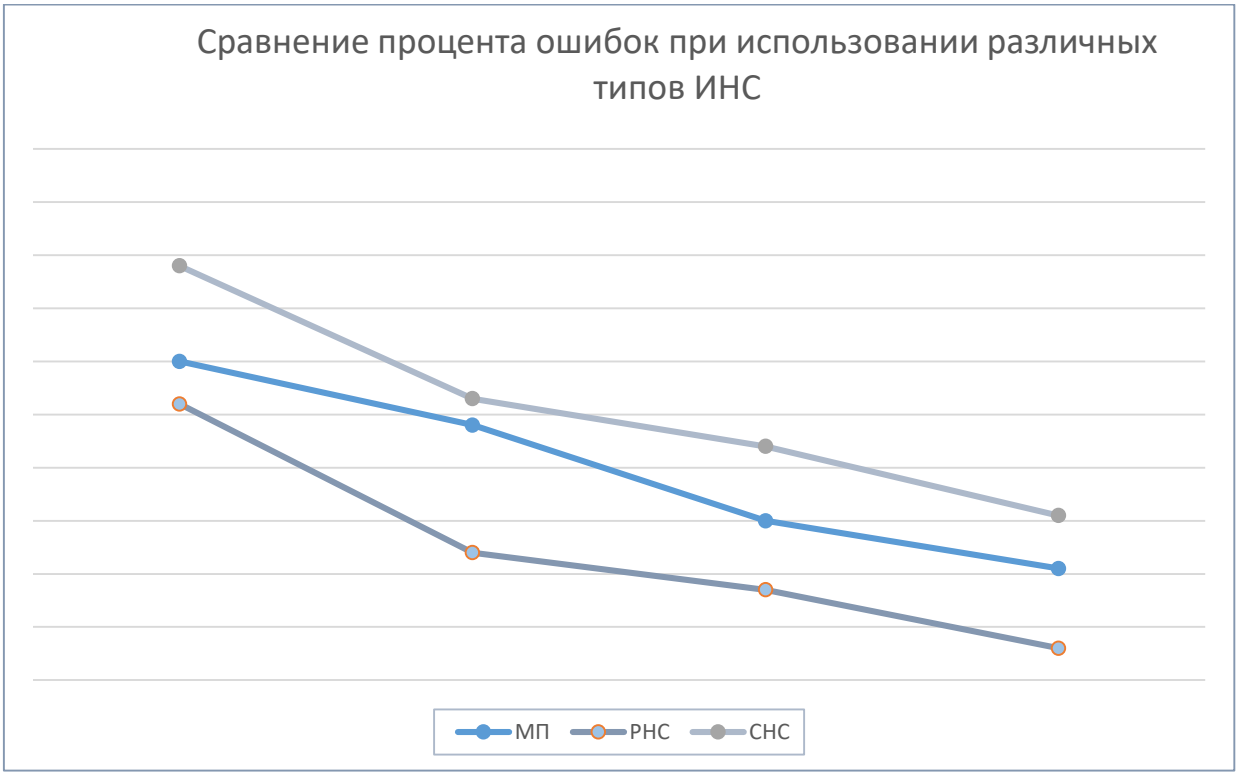

Рис.25. Сравнение процента ошибок классических архитектур ИНС, предназначенных для работы с картами

3.2 Математическое описание методики оценки окружающей обстановки

Теперь необходимо формализовать конкретную задачу оценки обстановки в ближней морской зоне. Решим задачу оптимального управления, моделирующую динамику рассматриваемой искусственной нейронной сети в заданных ниже обозначениях. Динамика сети из n нейронов описывается системой дифференциальных уравнений с запаздыванием.

$$\dot{x}_i(t) = -\gamma_i x_i(t) + \sum_{j=1}^{n} (\omega_{ij}(t) g\left(x_j(t)\right) + u_i(t)); \quad i,j = \overline{1 \dots n}; \qquad (15)$$

$$\dot{x}_i(t) = -\gamma_i x_i(t) + g_i(z_i(t)) + u_i(t); \quad i,j = \overline{1 \dots n};$$

$$z_i(t) = \sum_{j=1}^{n} \omega_{ij}(t) x_j(t-h);$$

$$g_i(z_i(t)) = (1 + \exp(-\lambda \sum_{j=1}^{n} \omega_{ij}(t) x_j(t-h)))^{-1};$$

$$\dot{x}_i(t) = -\gamma_i x_i(t) + \left(1 + \exp\left(-\lambda \sum_{j=1}^{n} \omega_{ij}(t) x_j(t-h)\right)\right)^{-1} + u_i(t);$$

$$i,j = \overline{1 \dots n};$$

С точки зрения биологического прототипа, т.е. нейронной сети, можно сказать, что уравнение описывает накапливаемый потенциал (электрический импульс) нейрона в данный момент времени, а также его изменение с течением времени.

Потенциал нейрона образуется и изменяется под воздействием многих факторов:

$x_i(t)$ – собственный потенциал нейрона в данный момент времени

$\gamma_i -$ собственное затухание i–го нейрона, описывает воздействие на нейрон собственных сил, которые негативно влияют на потенциал, а также затухание сигнала при передаче от одного нейрона к другим. Сумму $\sum_{j=1}^{n} \omega_{ij}(t)x_j(t-h)$ можно назвать суммой потенциалов ансамбля нейронов. Эта сумма воздействия всех соседних нейронов на i-ый нейрон.

Сумма $\sum_{j=1}^{n} \omega_{ij}(t)x_j(t-h)$ является главным элементом в формировании потенциала i-го нейрона, поэтому данную сумму назовем телом i-го нейрона.

$\omega_{ij}(t)x_j(t)$

$x_j(t-h) -$ показывает запаздывание сигнала нейронной сети. Таким образом, на потенциал i-го нейрона достаточно сильным образом воздействует остаточный импульс нейронов в предыдущий момент времени.

Функции управления $\omega_{ij}(t)$ *описывают* аксоны нейронов – электрический или химический импульс, который передается от одного нейрона к другим, тем самым изменяя потенциалы, является важнейшим связующим элементом нейронной сети, т.к. отвечает за взаимодействие и работоспособность всей сети. В данном случае это воздействие на i-ый нейрон, j-го нейрона.

Функция активации $g_i(z_i(t))$ преобразует накопленный потенциал нейрона согласно некоторой функциональной зависимости. Прототипом являются процессы, происходящие в теле нейрона, вызванные, например, сигналами или импульсами со стороны периферической нервной системы организма. (вследствие каких-либо изменений внешней среды.)

Собственный потенциал нейронов не должен выходить за рамки ограничений:

$$x_i(t) \le B_i\,,\ i = \overline{1,n}. \qquad (16)$$

Характеристики нейронов в начальный момент времени известны:

$$x_i(0) = a_i,\ \ i = \overline{1,n}; \qquad (17)$$

$x_i(t) = \varphi_i(t),\ t = \overline{-h,0}$;

Функция управления $u_i(t)$ характеризует внешнее воздействие на –ый нейрон. Это могут быть какие-либо изменения среды, на которые реагирует организм изменением скорости передачи электрических и химических импульсов нервной системы. Известны ограничения на управляющие функции:

$$\left|\omega_{ij}\right| \le b,\quad |u_i|\ \le c \qquad (18)$$

Целью управления динамикой нейронной сети является обучение сети, которое подразумевает следующие задачи(критерии):

1) В конечный момент времени характеристики нейронов должны совпадать с входными данными A_i,

2) Во время выполнения процесса, характеристики нейронов не должны выходить за пределы заданного диапазона значений. (B_i)

3) Управление u_i должно стремиться к минимальному значению для данного процесса.

4) Управление ω_{ij} также должно стремиться к минимальному значению для данного процесса.

Задачи управления можно формализовать в виде следующего целевого функционала

$$\mathrm{I}([x],[\omega],[u],[t]) = S\sum_{i=1}^{n}(x_i(T) - A_i)^2 + \sum_{i=1}^{n}\int_0^T M_i[\max(0; x_i(t) - B_i)]^2\,dt + L\sum_{i=1}^{n}\int_0^T u_i^2(t)\,dt + K\sum_{j=1}^{n}\sum_{i=1}^{n}\int_0^T \omega_{ij}^2(t)\,dt \ \to inf;\quad i = \overline{1,n} \qquad (19)$$

Основной практической целью является получение оптимальных управлений процесса, при помощи которых достигается минимум целевого функционала.

Оптимальные управления процесса мы получим методом градиентного спуска или, в терминологии обучения ИНС, методом обратного распространения ошибки.

Для решения задачи оптимального управления (16)-(19) используем принцип максимума Понтрягина.

Введем функцию Понтрягина:

$$H([x],[\omega],[u],[t]) == -\lambda_0 \sum_{i=1}^{n}(M_i[\max(0; x_i - B_i)]^2 + L\, u_i^2) - \lambda_0 K \sum_{j=1}^{n}\sum_{i=1}^{n} \omega_{ij}^2 + \sum_{i=1}^{n} p_i(t)\big(-\gamma_i\, x_i + (1+\exp(-\lambda \sum_{j=1}^{n} \omega_{ij}(t) x_j(t-h)))^{-1} + u_i\big),\ \ i,j = \overline{1,n} \quad (20)$$

Запишем принцип максимума:

$$-\lambda_0 \sum_{i=1}^{n} M_i[\max(0; \bar{x}_i - B_i)]^2 - \lambda_0 \sum_{i=1}^{n} L\bar{u}_i^2 - \lambda_0 K \sum_{j=1}^{n}\sum_{i=1}^{n} \bar{\omega}_{ij}^2 + \sum_{i=1}^{n} p_i(t)\big(-\gamma_i\, \bar{x}_i + g_i(z_i(t)) + \bar{u}_i\big) = \max(-\lambda_0 \sum_{i=1}^{n} M_i[\max(0; \bar{x}_i - B_i)]^2 - \lambda_0 \sum_{i=1}^{n} L\bar{v}_i^2 - \lambda_0 K \sum_{j=1}^{n}\sum_{i=1}^{n} \bar{w}_{ij}^2 + \sum_{i=1}^{n} p_i(t)\big(-\gamma_i\, \bar{x}_i + g_i(\bar{x}_j\, w_{ij}) + v_i\big)) =$$

$$-\lambda_0 \sum_{i=1}^{n} M_i[\max(0; \bar{x}_i - A_i)]^2 - \sum_{i=1}^{n} p_i(t)\,\gamma_i\,\bar{x}_i + \sum_{i=1}^{n} \underbrace{max}_{w_{ij}} \Big(p_i(t) g_i(\bar{x}_j\, w_{ij})\Big) - \lambda_0 K \sum_{j=1}^{n} w_{ij}^2) + \sum_{i=1}^{n} \underbrace{max}_{v_i} (p_i(t) v_i - \lambda_0 L \bar{v}_i^2)\ ;$$

Сопряженные вектор-функции вычисляются по формулам

$$\dot{p_k}(t) = -\frac{\partial H}{\partial x_k}(t) - \frac{\partial H}{\partial y_k}(t+h), \text{ где } y_k = x_k(t-h)$$

$$\dot{p_k}(t) = 2\lambda_0 M_k \max(0; x_k - A_k) + p_k(t)\gamma_k - \lambda \sum_{i=1}^{n} p_i(t+h)\omega_{ik}(t+h) \exp(-\lambda \sum_{j=1}^{n} \omega_{ij}(t) x_j(t-h))\big(1+\exp\big(-\lambda \sum_{j=1}^{n} \omega_{ij}(t) x_j(t-h)\big)\big)^{-2};$$

и удовлетворяют условиям трансверсальности

$$p_k(T) = -\lambda_0 \frac{\partial \Phi}{\partial x_k} = -2\lambda_0 (x_k(T) - A_k);$$

Принцип максимума позволяет свести задачу оптимального управления процесса к решению краевой задачи:

$$\dot{p_k}(t) = 2\lambda_0 M_k \max(0; x_k - A_k) + p_k(t)\gamma_k - \lambda \sum_{i=1}^{n} p_i(t +$$

$$h)\bar{\omega}_{ik} \exp\left(-\lambda \sum_{j=1}^{n} \omega_{ij}(t) x_j(t-h)\right)\left(1 + \exp\left(-\lambda \sum_{j=1}^{n} \omega_{ij}(t) x_j(t-h)\right)\right)^{-2};$$

$$p_k(T) = -\lambda_0 \frac{\partial \Phi}{\partial x_k} = -2\lambda_0 (x_k(T) - A_k),$$

$$\dot{x_i}(t) = -\gamma_i x_i(t) + \sum_{j=1}^{n} \bar{\omega}_{ij}(t)\, g\left(x_j(t)\right) + \bar{u}_i(t);\ \ i,j = \overline{1 \dots n};$$

$$\dot{x_i}(t) = -\gamma_i x_i(t) + g_i(z_i(t)) + u_i(t);\ \ i,j = \overline{1 \dots n};$$

$x_i(0) = a_i,$ *где* $\bar{\omega}_{ij}$, $\bar{u}_{ij}$ – оптимальные управления

$i,j = \overline{1,n}$

3.2.2 Численное решение дискретной задачи оптимального управления

Получим численное решение дискретной задачи оптимального управления. Для этого приведем исходную задачу к дискретному виду. Разобьем отрезок [0,.T] на q отрезков. Шаг дискретизации $\Delta t = \frac{T}{q}$.

Аппроксимируем систему дифференциальных уравнений по схеме Эйлера:

$$\dot{x_i}(t^l) \approx \frac{x_i^{l+1} - x_i^l}{\Delta t};$$

Обозначим: $x_i(t) = x_i^l$

$h = v\Delta t$

$x_i(t-h) = x_i^{l-v}$

Аппроксимируем интеграл $\int_0^T M_i[\max(0; x_i(t) - A_i)]^2\, dt$ по методу прямоугольников:

$$\int_0^T M_i[\max(0; x_i(t) - B_i)]^2\, dt \approx \sum_{l=0}^{q-1} \sum_{i=1}^{n} M_i\left[\max\left(0; x_i^l - B_i\right)\right]^2 \Delta t;$$

Аппроксимируем интеграл $\int_0^T u_i^2(t)\, dt$ по методу прямоугольников:

$$\int_0^T u_i^2(t)\, dt \approx \sum_{l=0}^{q-1} \sum_{i=1}^{n} (u_i^l)^2 \Delta t;$$

Аппроксимируем интеграл $\int_0^T \omega_{ij}^2(t) dt$ по методу прямоугольников:

интеграл $\int_0^T \omega_{ij}^2(t)dt \approx \sum_{l=0}^{q-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(\omega_{ij}^l)^2\,\Delta t$

Получаем рекуррентное соотношение для вычисления фазовой траектории:

$$x_i^{l+1} = x_i^l + \left(-\gamma_i\, x_i^l + g_i^l(z_i^l) + u_i^l\right)\Delta t;$$

$$z_i^l = \sum_{j=1}^{n} \omega_{ij}^l\, x_j^{l-\nu};$$

$$g_i^l(z_i^l) = \left(1 + \exp(-\lambda \sum_{j=1}^{n} \omega_{ij}^l\, x_j^{l-\nu})\right)^{-1};$$

$$x_i^{l+1} = x_i^l + \left(-\gamma_i\, x_i^l + (1 + \exp(-\lambda \sum_{j=1}^{n} \omega_{ij}^l\, x_j^{l-\nu}))^{-1} + u_i^l\right)\Delta t; \qquad (20)$$

$$(g_\imath\acute{})_{w_{ij}^l} = \lambda \frac{e^{-\lambda \sum_{j=1}^{n} \omega_{ij}^l x_j^{l-\nu}}\, x_j^{l-\nu}}{(1 + e^{-\lambda \sum_{j=1}^{n} \omega_{ij}^l x_j^{l-\nu}})^2};$$

$$(g_i^{l}\acute{})_{w_{ij}^l} = \lambda x_j^{l-\nu} \exp(-\lambda \sum_{j=1}^{n} \omega_{ij}^l\, x_j^{l-\nu})\left(1 + \exp\left(-\lambda \sum_{j=1}^{n} \omega_{ij}^l\, x_j^{l-\nu}\right)\right)^{-2};$$

$$(g_i^{l}\acute{})_{x_k^m} = \lambda \omega_{ik}^{m+\nu} \exp(-\lambda \sum_{j=1}^{n} \omega_{ij}^m\, x_j^{m-\nu})\left(1 + \exp\left(-\lambda \sum_{j=1}^{n} \omega_{ij}^m\, x_j^{m-\nu}\right)\right)^{-2};$$

$$(g_\imath\acute{})_{x_k^m} = \frac{e^{-\lambda \sum_{j=1}^{n} \omega_{ij}^m x_j^{m-\nu}}\, \omega_{ik}^{m+\nu}}{(1 + e^{-\lambda \sum_{j=1}^{n} \omega_{ij}^m x_j^{m-\nu}})^2};$$

Начальные значения:

$$x_i(0) = a_i;$$

$$x_i(t) = \varphi_i(t),\ t = \overline{-h, 0}; \qquad (21)$$

Целевая функция примет вид:

$$\mathrm{I} = S \sum_{i=1}^{n}\left(x_i^q - A_i\right)^2 + \sum_{l=0}^{q-1}\sum_{i=1}^{n} M_i\left[\max\left(0; x_i^l - B_i\right)\right]^2 \Delta t +$$

$$L \sum_{l=0}^{q-1}\sum_{i=1}^{n}(u_i^l)^2 \Delta t + K \sum_{l=0}^{q-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(\omega_{ij}^l)^2\,\Delta t \to \inf; \qquad (22)$$

С помощью метода множителей Лагранжа получим необходимое условие оптимальности.

Составим функцию Лагранжа:

$$L = \lambda_0 \left(S \sum_{i=1}^{n}\left(x_i^q - A_i\right)^2 + \sum_{l=0}^{q-1}\sum_{i=1}^{n} M_i\left[\max\left(0; x_i^l - B_i\right)\right]^2 \Delta t +\right.$$

$$L \sum_{l=0}^{q-1}\sum_{i=1}^{n}(u_i^l)^2 \Delta t + K \sum_{l=0}^{q-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(\omega_{ij}^l)^2\,\Delta t\Big) + \sum_{l=0}^{q-1}\sum_{i=1}^{n}(p_i^{l+1}\ (x_i^{l+1} -$$

$$x_i^l - \Delta t\big(-\gamma_i x_i^l + (1 + \exp(-\lambda \sum_{j=1}^{n} \omega_{ij}^l\, x_j^{l-\nu}))^{-1} + u_i^l\big);$$

Запишем условия стационарности:

$$\frac{\partial L}{\partial x_k^m} = p_k^m - p_k^{m+1} + p_k^{m+1}\gamma_k \Delta t - \Delta t \sum_{i=1}^{n} p_i^{m+1} \frac{\partial g_k^m(z_k^m)}{\partial x_k^m} +$$

$$+M_k 2\lambda_0 \Delta t[\max(0; x_k^m - B_k)] = 0;\ \ k = \overline{1 \dots n}; m = \overline{0 \dots q-1}$$

$$\frac{\partial L}{\partial x_k^m} = p_k^m - p_k^{m+1} + p_k^{m+1}\gamma_k \Delta t -$$

$$\lambda \Delta t \sum_{i=1}^{n} p_i^{m+\nu+1} \omega_{ik}^{m+\nu} \exp(-\lambda \sum_{k=1}^{n} \omega_{ik}^m x_k^{m-\nu})\ (1 +$$

$$\exp(-\lambda \sum_{k=1}^{n} \omega_{ik}^m x_k^{m-\nu}))^{-2} + M_k 2\lambda_0 \Delta t[\max(0; x_k^m - B_k)] = 0;\ \ k = \overline{1 \dots n}; m = \overline{0 \dots q-1}; \quad (23)$$

$$\frac{\partial L}{\partial x_k^q} = 2\lambda_0 S\left(x_k^q - A_i\right) + p_k^q = 0;\ \ k = \overline{1 \dots n}; \quad (24)$$

Запишем значения градиента:

$$\frac{\partial L}{\partial \omega_{km}^s} = -p_k^{s+1} \frac{\partial g_k^m(z_k^m)}{\partial \omega_{km}^s} \Delta t + 2K\Delta t \omega_{km}^s;\ k = \overline{1 \dots n};\ s = \overline{1 \dots n};$$

$$m = \overline{0 \dots q-1};$$

$$\frac{\partial L}{\partial \omega_{km}^s} = -\lambda \Delta t p_k^{s+1} x_m^{s-\nu} \exp(-\lambda \sum_{m=1}^{n} \omega_{km}^s x_m^{s-\nu})\ (1 +$$

$$\exp(-\lambda \sum_{m=1}^{n} \omega_{km}^s x_m^{s-\nu}))^{-2} + 2K\Delta t \omega_{km}^s;$$

(25)

$$\frac{\partial L}{\partial u_k^m} = 2Lu_k^m \Delta t - p_k^{m+1} \Delta t;\ k = \overline{1 \dots n}; m = \overline{0 \dots q-1}; \quad (26)$$

Выразим через данные уравнения импульсы p_k^m и p_k^q:

$$p_k^m = p_k^{m+1} - p_k^{m+1}\gamma_k \Delta t +$$

$$\lambda \Delta t \sum_{i=1}^{n} p_i^{m+\nu+1} \omega_{ik}^{m+\nu} \exp(-\lambda \sum_{k=1}^{n} \omega_{ik}^m x_k^{m-\nu})\ (1 +$$

$$\exp(-\lambda \sum_{k=1}^{n} \omega_{ik}^m x_k^{m-\nu}))^{-2} - 2M_k \lambda_0 \Delta t\ [\max(0; x_k^m - B_k)]; \quad (27)$$

$$p_k^q = -2S\lambda_0\left(x_k^q - A_k\right); \quad (28)$$

Выполним предельный переход и проверим соответствие с краевой задачей, тем самым проверяя правильность решения:

$\frac{p_k^{m+1}-p_k^m}{\Delta t} = \; p_k^{m+1}\gamma_k - \lambda\sum_{i=1}^n \omega_{ik}^{m+\nu}\, p_i^{m+\nu+1}\, \exp\left(-\lambda\sum_{j=1}^n \omega_{ij}^m\, x_j^{m-\nu}\right)\left(1 + \exp\left(-\lambda\sum_{j=1}^n \omega_{ij}^m\, x_j^{m-\nu}\right)\right)^{-2} + 2M_k\lambda_0[\max(0; x_k^m - B_k)];$

перейдем к пределу: $\lim\limits_{\Delta t\to 0}\frac{p_k^{m+1}-p_k^m}{\Delta t}$ получим

$$\dot{p}_k(t) = 2\lambda_0\, M_k \max(0; x_k - B_k) + p_k(t)\gamma_k - \lambda\sum_{i=1}^n p_i(t+h)\omega_{ik}(t+h)\exp\left(-\lambda\sum_{j=1}^n \omega_{ij}(t)x_j(t-h)\right)\left(1+\exp\left(-\lambda\sum_{j=1}^n \omega_{ij}(t)x_j(t-h)\right)\right)^{-2};$$

$$p_k^q = -2\lambda_0 S\left(x_k^q - A_k\right);$$

$$p_k(T) = -2\lambda_0\, S(x_k(T) - A_k);$$

3.2.3 Алгоритм поиска численного решения и блок схема алгоритма

Формализуем полученный алгоритм:

1) Задаем параметры модели.

2) Задаем параметры метода: ε – точность вычисления, α – шаг градиентного спуска, q – количество точек разбиения отрезка и начальный набор управлений $[u]^{(0)}$ и $[\omega]^{(0)}$.

3) По формуле (7) и (8) найдем набор $[x]^{(0)}$. Вычисляем $I^{(0)}$ по формуле (9), задавая $[M]^{(1)}$.

4) По формуле (14) и (15), начиная с q-го слоя, вычисляем $[p]^{(k)}$, $k = 0,1,\ldots$. По формулам (12) и (13) вычисляем $\left[\frac{\partial\tilde{L}}{\partial\omega_{ks}^m}; \frac{\partial\tilde{L}}{\partial u_k^m}\right]^{(k)}$.

5) Улучшаем управление на $k+1$ шаге: $[u]^{(k+1)} = [u]^{(k)} - \alpha\frac{\partial\tilde{L}}{\partial u}$

$$[w]^{(k+1)} = [w]^{(k)} - \alpha\frac{\partial\tilde{L}}{\partial w}$$

6) Вычисляем $[x]^{(k+1)}$, затем $[I]^{(k+1)}$.

7) Сравниваем значения $[I]^{(k)}$ и $I^{(k+1)}$. Если $I^{(k+1)} > I^{(k)}$, то уменьшаем шаг градиентного спуска $\alpha = \frac{\alpha}{N}$, где $N \in \mathbb{N}$ и переходим к пункту (5) той же итерации, иначе переходим к пункту (8)

8) Проверяем условие $\left|I^{(k+1)} - I^{(k)}\right| < \varepsilon$, если оно выполнено, то считаем, что оптимальное решение при заданном параметре $[M]^{(1)}$ найдено и переходим к пункту (9). В противном случае переходим на следующую итерацию $k := k + 1$ и переходим к пункту (4).

9) Проверяем условия:

$$\sum_{l=1}^{q-1}\sum_{i=1}^{n} M_i^l\left[\max\left(0; x_i^l - A_i\right)\right]^2 \le \varepsilon$$

$$\sqrt{\sum_{l=0}^{q}\sum_{i=1}^{n}\left(x_i^{l(k)} - x_i^{l(k+1)}\right)^2} \le \varepsilon$$

$$\sqrt{\sum_{l=0}^{q}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\omega_{ij}^{l(k)} + \omega_{ij}^{l(k+1)}\right)^2} \le \varepsilon$$

При совместном выполнении всех условий считаем, что решение найдено, в противном случае определяем новый штрафной коэффициент $[M]^{(1)} = \beta[M]^{(0)}$ и переходим к пункту (3) .

3.3 Обучение формализованной искусственной нейронной сети

Следующим этапом методики оценки обстановки в ближней морской зоне является обучение искусственной нейронной сети. Необходимо выбрать алгоритм обучения, подходящий к конкретной задаче. По результатам исследования наилучшие результаты были получены с помощью алгоритма обратного распространения ошибки. Это связано в первую очередь с тем, что алгоритм обратного распространения создан и оптимизирован под выбранную архитектуру ИНС и функцию активации, что позволяет максимально использовать его потенциал.

Его основная идея заключается в том, что изменение весов синапсов происходит с учетом локального градиента функции ошибки. Разница между реальными и правильными ответами нейронной сети, определяемыми на выходном слое, распространяется в обратном направлении (см. рисунок 26) —

навстречу потоку сигналов. В итоге каждый нейрон способен определить вклад каждого своего веса в суммарную ошибку сети. Простейшее правило обучения соответствует методу наискорейшего спуска, то есть изменения синаптических весов пропорционально их вкладу в общую ошибку [17].

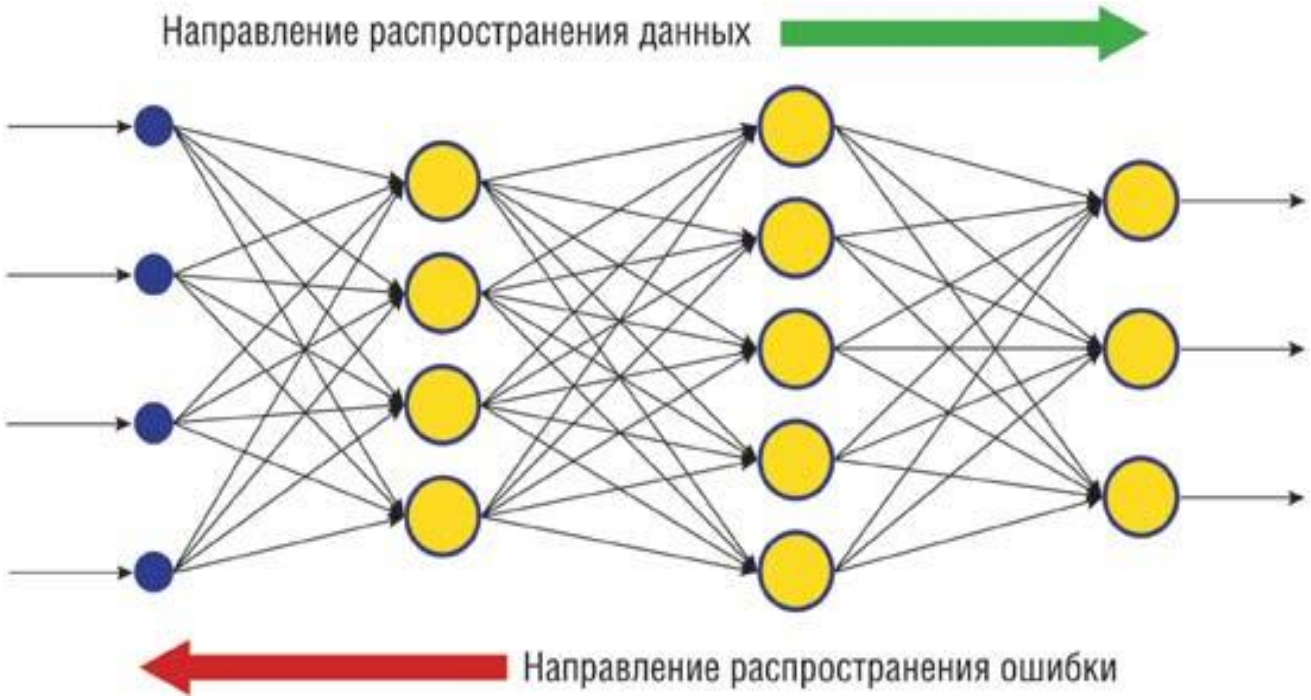

Рис.26. Метод обратного распространения ошибки для многослойной полносвязанной нейронной сети

Конечно, при таком обучении нейронной сети нет уверенности, что она обучилась наилучшим образом, поскольку всегда существует возможность попадания алгоритма в локальный минимум. Для этого используются специальные приемы, позволяющие «выбить» найденное решение из локального экстремума. Если после нескольких таких действий нейронная сеть сходится к тому же решению, то можно сделать вывод о том, что найденное решение, скорее всего, оптимально [10].

Алгоритм обучения НС с помощью процедуры обратного распространения строится так [15]:

1. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних.

2. Вычислить разницу между идеальным и полученным значениями выхода для выходного слоя.

Рассчитать изменения весов слоя.

3. Рассчитать разницу между идеальным и полученным значениями выхода и изменения весов для всех остальных слоев.

4. Скорректировать все веса в НС.

5. Если ошибка сети существенна, перейти на шаг 1. В противном случае – конец.

Сети на шаге 1 попеременно в случайном порядке предъявляются все тренировочные образы, чтобы сеть, образно говоря, не забывала одни по мере запоминания других.

Когда выходное значение стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться, поэтому область возможных значений выходов нейронов [0,1] желательно сдвинуть в пределы [-0.5,+0.5], что достигается простыми модификациями логистических функций. Например, сигмоид с экспонентой ($\sigma(x) = \frac{1}{1+e^{-x}}$) преобразуется к виду

$$f(x) = -0.5 + \frac{1}{1+e^{-x}}$$

Теперь коснемся вопроса емкости НС, то есть числа образов, предъявляемых на ее входы, которые она способна научиться распознавать. Для сетей с числом слоев больше двух, он остается открытым. Для НС с двумя слоями, то есть выходным и одним скрытым слоем, детерминистская емкость сети оценивается так:

$$\frac{N_w}{N_y} < C_d < \frac{N_w}{N_y} \times \log \frac{N_w}{N_y}$$

где:

- C_d – детерминистская емкость сети;
- N_w – число подстраиваемых весов;
- N_y – число нейронов в выходном слое.

Следует отметить, что данное выражение получено с учетом некоторых ограничений. Во-первых, число входов N_x и нейронов в скрытом слое N_h должно удовлетворять неравенству $N_x+N_h>N_y$. Во-вторых, $\frac{N_w}{N_y} > 1000$.

Фигурирующее в названии емкости прилагательное "детерминистский" означает, что полученная оценка емкости подходит абсолютно для всех возможных входных образов, которые могут быть представлены N_x входами. В действительности распределение входных образов, как правило, обладает некоторой регулярностью, что позволяет НС проводить обобщение и, таким образом, увеличивать реальную емкость. Так как распределение образов, в общем случае, заранее не известно, мы можем говорить о такой емкости только предположительно, но обычно она раза в два превышает емкость детерминистскую. Схематично данный алгоритм представлен на рисунке.

Одновременно с выбором обучающего алгоритма необходимо разработать соответствующие априорно верные наборы входных данных, так называемые обучающие наборы. Обучающий набор состоит из достаточного количества векторов (для поставленной задачи достаточно 10 000 векторов), представляющих собой наборы входных данных с правильным результатом. В нашем случае обучающие наборы состоят из данных, полученных из открытых порталов геоданных и проанализированных специалистами-экспертами ТвГУ.

В наборы были включены такие параметры как глубина, расстояние до берега, наличие на участке военных действий, наличие транспортного маршрута и судов в районе прохождения зоны, осадки, давление и наличие опасных погодных условий, наличие на маршруте рыбных ресурсов и животных, занесенных в Красную книгу региона. Каждому из этих параметров был присвоен соответствующий приоритет, необходимый для функционирования ИНС. В соответствии со спецификой поставленной задачи основной упор оценки

был сделан на толщину ледяного покрова в ближней морской зоне, для чего этот параметр был вынесен в разряд специальных, тем самым повышая приоритет. Обучающие вектора составлялись в реалиях побережья Баренцева моря.

Таблица 3.

Фрагмент обучающего набора

Связанная территория	Параметр	1	2	3	4
Навигационные параметры	Глубина, м (N1)	17	36	54	12
	Расстояние до берега, км (N2)	26	36	65	84
Антропогенные параметры	Наличие военных конфликтов/учений (A1)	0	0	1	1
	Наличие транспортного маршрута (A2)	0	1	0	1
	Наличие судов в данной зоне (A3)	0	1	1	1
Погодные параметры	Количество осадков, мм (P1)	7	3	6	4
	Давление, мм рт.ст. (P2)	756	751	745	745
	Наличие опасных погодных условий (P3)	0	1	1	0
Экологические параметры	Лицензионные участки УВС (E1)	0	1	0	1
	Наличие животных из Красной книги региона (E2)	1	1	1	0
Специфические параметры	Толщина ледяного покрова, м (S1)	1,2	0	0,3	0,6

Обучение ИНС происходило по следующему правилу. Сначала на вход сети были последовательно поданы все 10 000 векторов обучающего набора, с помощью чего было произведено обучение по алгоритму обратного распространения ошибки. Каждый раз, после обучения на 2000 наборов

производился тест обучения – проверялась ошибка на 100 произвольных наборах из уже пройденных (тестовый контроль ошибки). Затем после завершения всего обучения была проверена ошибка на 1000 (10% от обучающего множества) – этап окончательного контроля. На данном этапе ошибочная оценка была принята на 8 наборах, что составило 0,8% при заданном пороге в 1%, что является хорошим результатом.

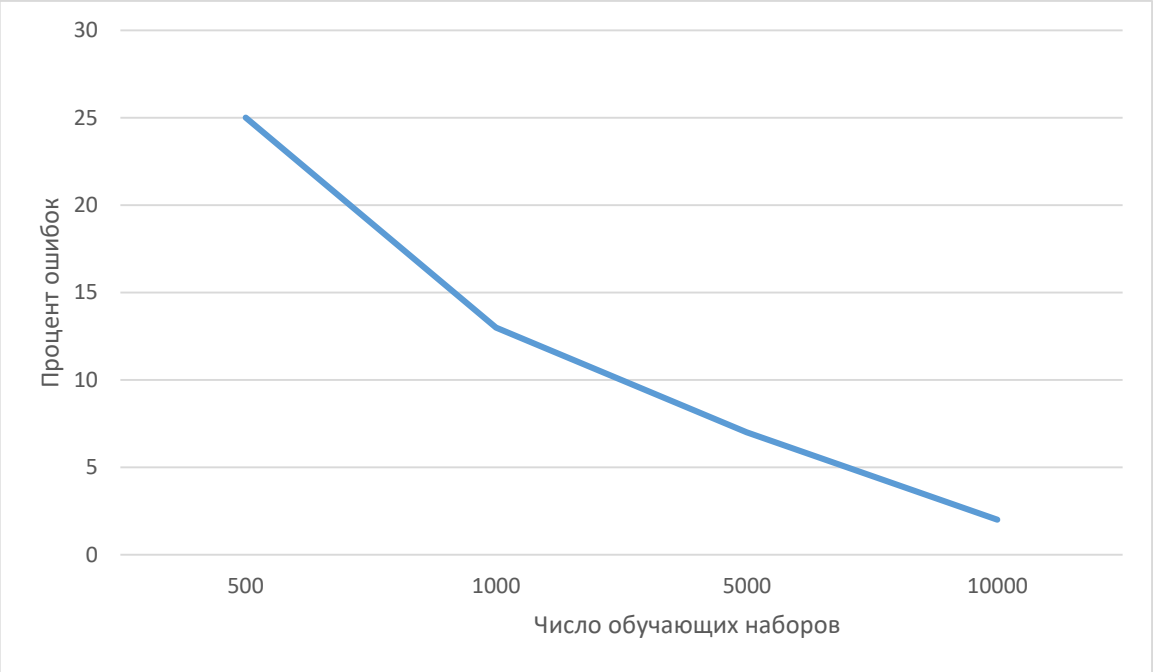

Рис. 27. Зависимость процента ошибочных срабатываний нейронной сети от числа обучающих наборов

3.4 Анаморфирование геоизображения оценки обстановки

Следующим этапом является донесение информации оператору для быстрого анализа и принятия решения для широкого спектра практических задач через графическое представление информации. Эта информация должна всесторонне учитывать различные факторы и характеристики исследуемого процесса или явления, а для того, чтобы ее восприятие и обработка привели к резкому сокращению времени ее анализа и, соответственно, времени принятия обоснованного варианта решения, носить в основном, визуальный характер - в виде визуальных информационных образов. Метод анаморфирования,

относящийся к когнитивной компьютерной графике, используется в качестве одного из способов анализа исходных данных и принятия обоснованного варианта решения на основе визуализации актуальной информации. Существует достаточно большое количество численных методов построения анаморфоз, каждый из которых обладает рядом достоинств и недостатков. Но эти методы не уделяют должного внимания проблеме отображения результатов анаморфирования изображения.

Понятие анаморфозы, которая определяется как переход от одного визуального образа, построенного на основе евклидовой метрики, к другому визуальному образу, в основе которого лежит метрика рассматриваемого процесса или явления на основе выбранного показателя, является основой метода анаморфирования. При этом возможность визуализировать сложные и неочевидные показатели, учитываемые при принятии решений, является важным преимуществом метода анаморфирования.

Ключевым аспектом анаморфирования является выбор так называемых ключевых точек.

Соответственно задача отображения результатов анаморфирования состоит в следующем. В качестве исходных данных поступают начальные и конечные координаты ключевых точек изображения. Координаты любых двух этих точек не совпадают. Также имеется некоторое растровое изображение. Необходимо получить изображение, преобразованное в соответствии с перемещением ключевых точек.

Существует достаточно большое количество методов анаморфирования. Большинство из них опирается на разбитие анаморфируемой области на какие-либо участки с последующим изменением их площади, либо методом выравнивания плотности показателя, либо афинными преобразованиями с соответствующими коэффициентами. В методе, предложенном У. Тоблером, методе треугольников, разработанном в Московском государственном университете им. Ломоносова, алгоритме лаборатории Лоуренс Беркли изображение разбивается на геометрические фигуры - ячейки. В ходе работы

алгоритмов положение вершин ячеек меняется так, чтобы каждая ячейка приобрела определенную площадь. Вышеперечисленные методы подробно описывают способы определения нового положения вершин, но они не содержат пояснений, как отразить эти изменения на визуальном образе.

Можно разделить методы анаморфирования на две категории: рассматривающие все точки области и только точки границы. Первая группа методов дает преимущество в точности, однако вторая превосходит в скорости расчетов.

Предлагается следующий алгоритм преобразования:

1. Проводим разбитие региона на территории по однородности показателя.
2. Задаем линии абсцисс и ординат
3. На границах территорий определяем ключевые точки, после чего соединяем их зубчатой линией.
4. Согласно коэффициенту показателя, осуществляем перемещение точек к центру каждой зоны.
5. Выполняем проверку на заданные ограничения.
6. Получаем анаморфированный картоид.

В качестве предмета анаморфирования предполагается использовать картоид, разбитый на территории, согласно приблизительной однородности плотности рассматриваемого показателя. Такая форма представления наиболее удобна для дальнейшей обработки и визуализации. (Рис.26).

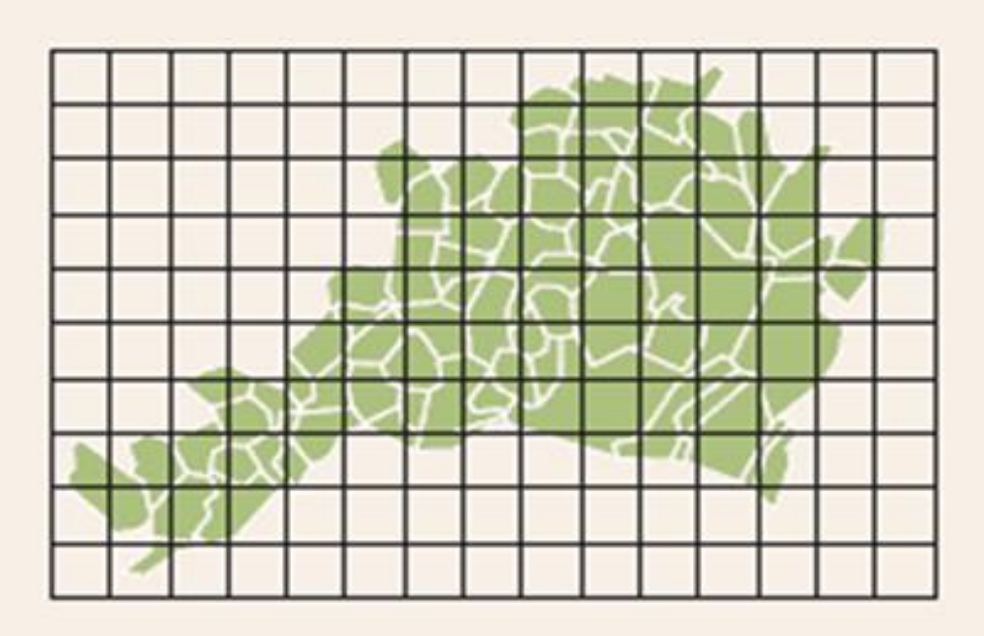

Рис.28. Картоид, отображающий ледовую обстановку на участке Баренцева моря с заданной координатной сеткой

Важным аспектом для дальнейших действий является выбор центра территории. Для удобства центром территории будем считать центр тяжести фигуры. Воспользуемся следующими формулами

$$x_c = \frac{\iint_D x\,\rho(x,y)dxdy}{m}$$

$$y_c = \frac{\iint_D y\,\rho(x,y)dxdy}{m},$$

где $m = \iint_D \rho(x,y)dxdy$ – масса фигуры, $\rho(x,y)$ – плотность параметра фигуры, D – площадь фигуры. Стоит отметить, что функция плотности может быть как рассмотрена не только как функция положения $\rho(x,y)$, но и как функция времени $\rho(x,y,t)$.

Затем необходимо осуществить привязку области к координатам. Удобнее всего это делать в геоинформационной системе ArcGIS. Она позволяет осуществлять привязку и последующее редактирование координатной сетки встроенными средствами системы. После осуществления привязки необходимо выбрать ключевые точки на границе каждой области. От частоты разбиения во многом зависти точность и показательность анаморфирования (Рис.29)

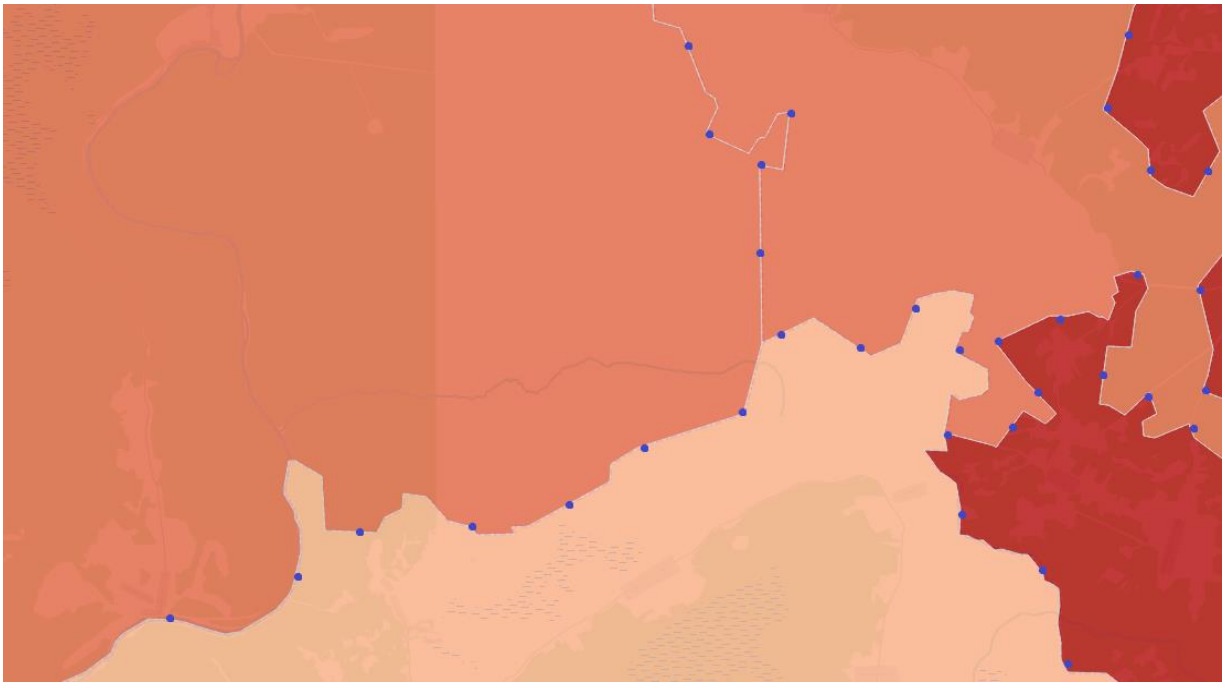

Рис. 29. Фрагмент картоида с нанесенными на него ключевыми точками.

Построим уравнения прямых, на каждой из которых лежит ключевая точка границы A(x_a, y_a) и центр C(x_c, y_c) (Рис.30).

Уравнения таких прямых будут иметь вид:

$$\frac{x-x_c}{x_a-x_c} = \frac{y-y_c}{y_a-y_c}.$$

Рис.30. Фрагмент картоида с заданными прямыми.

Ключевым шагом анаморфирования будет изменение площади территорий. Для этого каждая ключевая точка границы будет смещаться к центру территории на расстояние k. Расстояние считается в метрике пространства. Данное действие выполняется для каждой территории в отдельности, так как ключевые точки принадлежат как минимум двум территориям. Обход территорий начинается из левого нижнего угла картоида.

$k = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ – формула расстояния между ключевой точкой и центром, где $k = \frac{\max(p_i)-p_i}{q}$. При этом q – расстояние между двумя максимально удаленными точками границы, иначе говоря диаметр области, p_i – параметр

анаморфирования конкретной области. Из вышеприведенных формул следует формула итогового сдвига.

$$\begin{cases} \dfrac{x - x_c}{x_a - x_c} = \dfrac{y - y_c}{y_a - y_c}; \\ k = \sqrt{(x - x_{c)}{}^2 + (y - y_{c)}{}^2} \end{cases}$$

После выполнения аффинного преобразования всех координат необходимо отобразить

область внутрь прямоугольника, с ограничениями $x_{min} = 0$, $x_{max} = L_x$, $y_{min} = 0$, $y_{max} = L_y$.Для упрощения, параметры L берутся целочисленными. Выполним преобразования Фурье.

$$u_x(x, y, t) = -\frac{L_y}{\pi\rho(x, y, t)} \sum_{m=1}^{\infty} \sum_{n=0}^{\infty} \left[\frac{m}{m^2 L_y^2 + n^2 L_x^2} \rho(m, n) * \sin(\frac{m\pi x}{L_x}) \cos(\frac{n\pi y}{L_y}) \right]$$

$$u_y(x, y, t) = -\frac{L_x}{\pi\rho(x, y, t)} \sum_{m=0}^{\infty} \sum_{n=1}^{\infty} \left[\frac{m}{m^2 L_y^2 + n^2 L_x^2} \rho(m, n) * \cos(\frac{m\pi x}{L_x}) \sin(\frac{n\pi y}{L_y}) \right]$$

Рассмотрим две меры локальной ошибки искажения.

$$e(x, y) = \ln\left(\frac{a(x, y)}{b(x, y)}\right);$$

$$\bar{e}(x, y) = 2\arcsin(\frac{a(x, y) - b(x, y)}{a(x, y) + b(x, y)})$$

В случае конформного отображения, эти ошибки должны быть равны между собой и равны 0. Однако это идеальный, не всегда достижимый случай. Потому, в качестве глобальной меры для расчета отклонения картограммы от конформности, мы можем использовать максимальную локальную ошибку искажения.

Предлагается два варианта графического представления: одноцветное с разбиением на территории и многоцветное с разбиением на меньшие территории и раскраской в различные цвета. Первое представление оптимально в задачах, где важна скорость реакции и число входных параметров невелико (до десяти), второе позволяет проводить более подробный анализ оценки и оптимально для

задач долгосрочного планирования. На рис. 31 представлен первый вариант графического представления.

Рис. 31. Анаморфированный картоид геоизображения ледовой обстановки на участке Баренцева моря

Проанализировав полученный картоид, можно выделить районы, оптимальные для прохождения судна. Эти районы выделены на рис. 32 синим цветом.

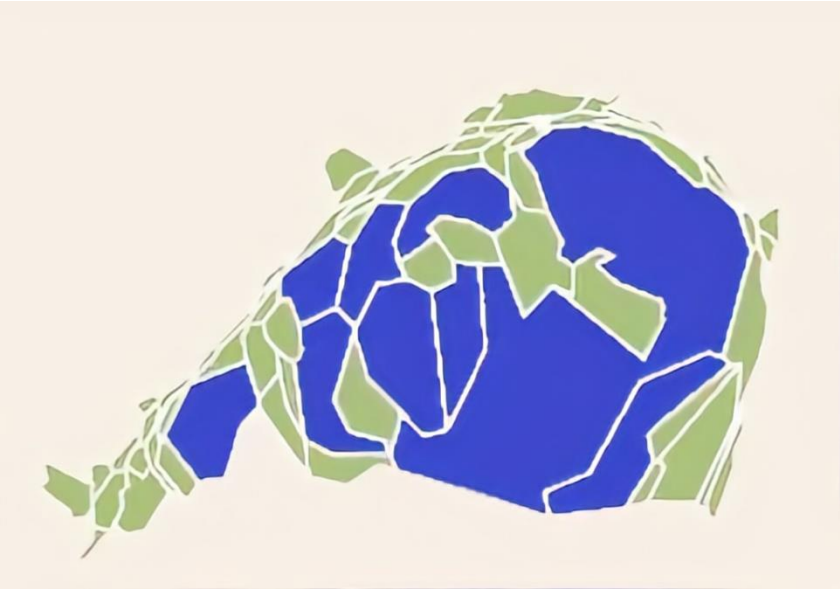

Рис. 32. Картоид-анаморфоза с выделенными (фиолетовым цветом) оптимальными районами плавания.

Сравнение описанной выше методики с классическим алгоритмом Гастнера-Ньюмана, приведено в табл. 4.

Таблица 4

Показатели точности работы предложенного алгоритма и методики Гастнера-Ньюмана

Рассматриваемая область	Алгоритм	e_0	e_∞	$\overline{e_0}$	$\overline{e_\infty}$	t (сек)
Участок Баренцева моря в условиях ледовой обстановки	Гастнера-Ньюмана	0,256	6,53	0.384	7,88	25
	Растяжением границы	0.243	6,55	0,341	7,65	5,7

Согласно данным таблицы, алгоритм не дает существенного изменения ошибки, однако виден существенный прирост ускорения выполнения алгоритма.

В качестве финального этапа исследований можно осуществить детопологизацию картоида и перенос результатов оценки на географическую карту. Перенос представлен на рис. 33. Красным цветом на географической карте выделены оптимальные для прохождения судов районы. Размытость изображения обусловлена приблизительным характером перехода.

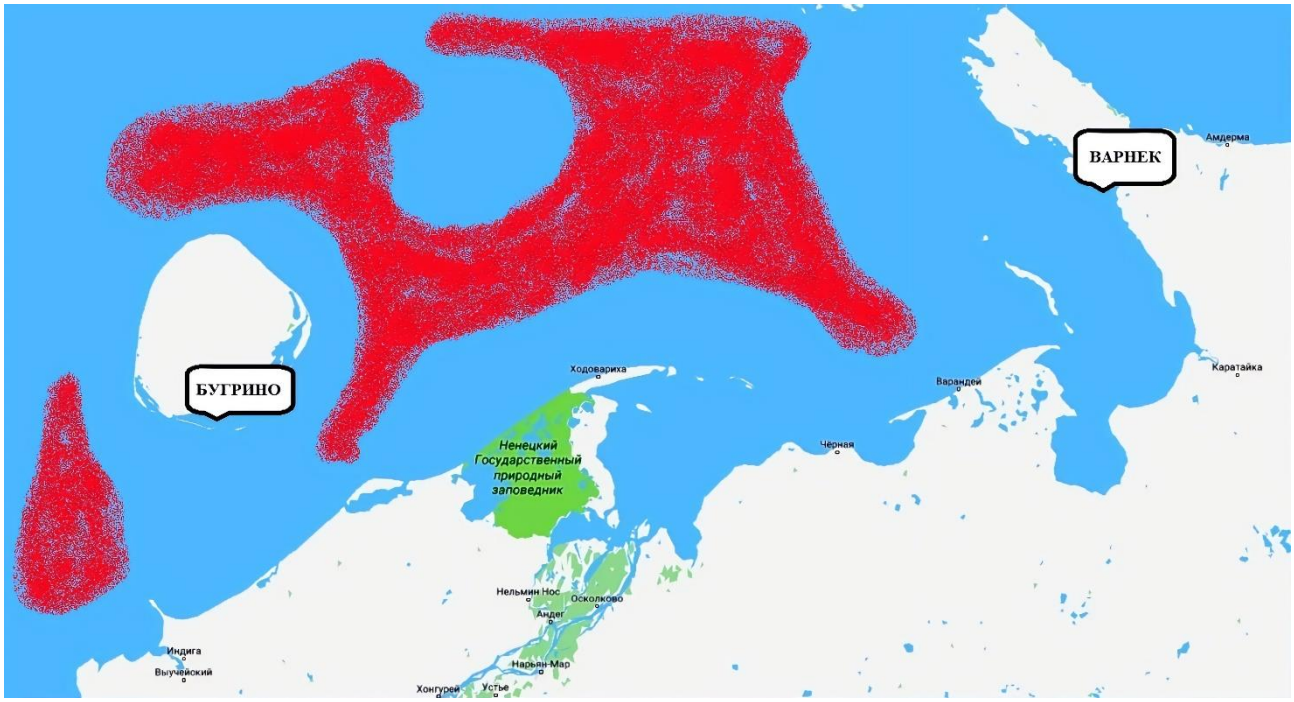

Рис. 33. Детопологизация полученных оценок — перенос оптимальных районов плавания на географическую карту.

Итоговое представление оценки, полученное сочетанием операции анаморфирования с аналитическим аппаратом искусственных НС позволяет проводить анализ обстановки в ближней морской зоне, минимизируя субъективный фактор, однако финальным этапом методики предусмотрен контроль уполномоченного лица над процедурой принятия решения, либо непосредственное принятие решения человеком на основании полученного графического представления.

3.5 Сравнение предложенной методики с существующими

Существуют и иные методики оценки обстановки в ближней морской зоне. Рассмотрим одну из методик, наиболее близкую к предложенной. Кратко рассмотрим ее суть.

Известно исходное состояние обстановки, определяемое в интеллектуальной ГИС совокупностью пространственно-соотнесенных данных $|d_s|$. Известны правила (отношения):

$$F_{zv}(d_{zv_e}; e=\overline{1,E_z}) \to d_{zv_a}, \quad (28)$$
$$z=\overline{1,Z};\ v=\overline{1,V_z},$$

связывающие эти данные с другими возможными пространственно-соотнесенными данными. Путем анализа статистики установлены временные характеристики, свойственные реальным процессам, развивающимся по этим правилам.

Требуется найти программу *PRG* развития обстановки на интервале времени *T*, а по ней результат $|d_w(T)|$ прогноза. Этот результат – те пространственно соотнесенные данные, которые ожидается наблюдать по истечении времени *T*.

С формальной точки зрения программу PRG можно записать в виде:

$$PRG = PRG\left\{ |d_s|; F_{zv}(d_{zv_e}; e = \overline{1, E_z}) \to d_{zv_a}; z = \overline{1, Z}; v = \overline{1, V_z}; |d_w(T)| \right\}. \qquad (29)$$

Согласно зависимости (2.18) ищется программа, позволяющая перейти от $|d_s|$ к $|d_w(T)|$. При этом изначально $|d_w(T)|$ определяется в неявном виде, как некоторая функция от времени *T*.

Отличие этой задачи от известных моделей в том, что при логическом выводе необходимо проверять условие $t_{PRG_j} \le T$. В соответствии с ним очередной *j*+1 шаг логического вывода может осуществляться только в случае, когда время t_{PRG_j} развития обстановки в соответствии программой PRG_j не превышает допустимого, *T*. С учетом этого $|d_w(T)|$ определяется как

$$|d_w(T)| = \begin{cases} |d_w(t_j)|, при \quad t_j = T; \quad j = \overline{1, J}; \\ неопределено, \ в \ противном \ случае. \end{cases} \qquad (30)$$

Здесь $|d_w(t_j)|$ - текущий *j*-й результат логического вывода, соответствующего программе PRG_j; *J* – заданное предельно допустимое число шагов логического вывода. Под шагом логического вывода в нашем случае понимается увеличение длины параллельной программы на одну функцию.

В перспективных системах база знаний может быть полностью не детерминирована (не определены все правила продукций, присутствуют элементы случайного вывода продукций, и т. д). Тогда логический вывод предлагается производить на основе модифицированного байесовского метода. Классическая формула Байеса, представлена ранее.

$$P(H:E) = \frac{P(E:H)P(H)}{P(E)}; \qquad (31)$$

$$P(E) = P(E:H)P(H) + P(E:\overline{H})P(\overline{H}), \quad (32)$$

где: *P(H)* – априорная вероятность гипотезы *H* при отсутствии каких-либо свидетельств (в нашем случае это вероятность того факта, что правило достоверно); *P(H:E)* – апостериорная вероятность *H* при наличии свидетельства *E* (в нашем случае это вероятность достоверности правила при проявлении на очередном шаге соответствующего результата логического вывода прогноза развития событий).

Если в базе знаний хранятся все возможные случаи проявления данного свидетельства, то формулу (2.21) можно переписать в более общем виде:

$$P(E) = \sum_{i=1}^{n} P(E:H_i)P(H_i). \quad (33)$$

O(H) – шансы, являющиеся функциями *P(E:H)* и $P(E:\overline{H})$, и если перейти к логарифмам величин, а в базе знаний хранить логарифм отношения p^+/p^- подтверждающих и опровергающих свидетельств, то все вычисления сведутся к суммированию.

$$O(H) = \frac{P(H)}{(1 - P(H))}; \quad (34)$$

$$O(H:E) = \left(\frac{P(E:H)}{P(E:\overline{H})}\right) O(H). \quad (35)$$

Считается, что найден наиболее вероятный результат, если существует *H*, *P(min)* которой больше, чем *P(max)* для любой другой гипотезы.

Вероятным заключение является, если *P(min)*, больше верхнего порога принятия решения.

Неопределенными будут заключения, когда *P(min)* меньше верхнего порога, но *P(max)* больше нижнего. В этом случае будут дальше задаваться вопросы до снятия неопределенности.

Когда *P(max)* меньше нижнего порога принятия решений, никакой вывод невозможен.

Однако для применения определенного выше метода необходимые для расчета вероятности *P(E:H)* и $P\left(E:\overline{H}\right)$ можно получить лишь проведя статистический эксперимент.

При этом в теоретических моделях обычно пользуются интерполяцией. Если мы знаем, что *E* истинно с определенностью *E'*, например *P(E:E')*, то отсюда следует, что

$$P\left(H:E'\right)=P\left(E:E'\right)P\left(H:E\right)+\left(1-P\left(E:E'\right)\right)P\left(H:\overline{E}\right). \qquad (36)$$

Взаимосвязь неопределенных показателей с действительными, можно представить в виде, показанном на рис. .

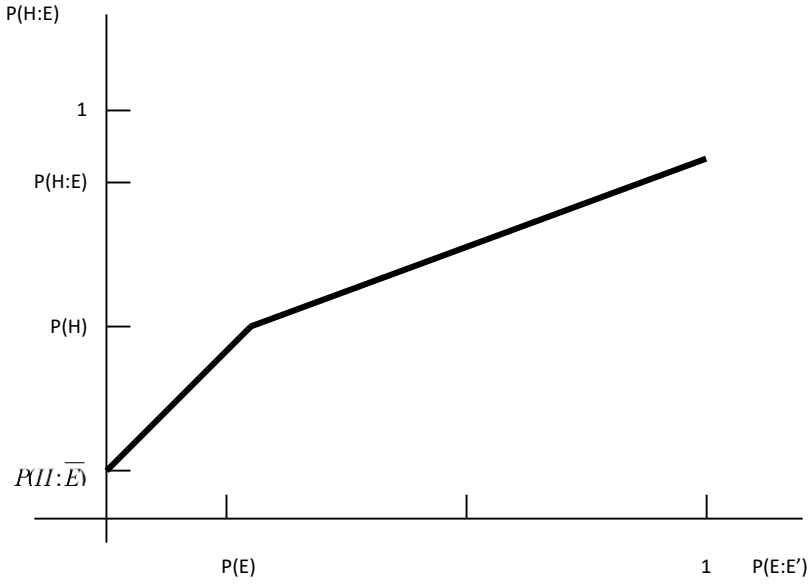

Рис. 34 – Оперирование с неопределенным показателем

При таком подходе возникает проблема задания предварительных значений шансов для всех правил вывода. Эксперты обычно задают предварительные

значения шансов для узлов субъективным (нематематическим) образом. Следовательно, такая сеть логических выводов обычно не бывает математически непротиворечивой. Допустим, что *E′* - это показание, на основании которого алгоритм предполагает присутствие *E*. Это изменит вероятность для *H* в соответствии с *P(H:E')*, и ее значение будет лежать между $P\left(H:\overline{E}\right)$ и *P(H:E)*. Если *P(E:E')* равно нулю, то *P(H:E)* должно соответствовать $P\left(H:\overline{E}\right)$. Если же *P(E:E')* равно 1, то *P(H:E')* должно иметь значение *P(H:E)*. Однако если мы ничего не знаем о *E* (т.е. *P(E:E') =P (E)*), то предшествующее значение шансов *H* не должно изменяться. Отсюда *P(H:E') = P(H)*. Эти три точки определяют взаимосвязь между *P(H:E')* и *P(E:E')*, как это показано на рис. 34. Такой прием позволяет справиться с проблемой противоречивости значений предшествующих значений шансов, присваиваемых экспертом каждому узлу сети. В результате получаем зависимость

$$P(H:E')=P\left(H:\overline{E}\right)+\frac{\left(P(H)-P\left(H:\overline{E}\right)\right)}{P(E)}P(E:E') \qquad (37)$$

при условии $0\le P(E:E')\le P(E)$ и

$$P(H:E')=P(H)+\frac{\left(P(H:E)-P(H)\right)}{1-P(E)}\left(\ P(E:E')-P(E)\ \right) \qquad (38)$$

при условии $P(E)\le P(E:E')\le 1$, а формула 35 примет вид:

$$O(H:E')=\left(\frac{P(E:E')}{P\left(E:\overline{E'}\right)}\right)O(H). \qquad (39)$$

Предложенный выше подход позволяет исключить распространение по сети ошибочных значений вероятностей, обусловленных противоречивостью заданных экспертом значений параметров. Такой же прием служит для

принудительного согласования вводимых в алгоритм значений вероятности и предшествующих значений шансов, приписываемых экспертом соответствующему узлу сети логических выводов. Требуется получить значение фактора определенности C(E:E'), лежащее между - 5 (отрицание) и + 5 (подтверждение). При этом принято присваивать C(E:E') значение, равное нулю, если нет воздействия на предшествующую вероятность, C(E:E') = 5 - для принудительной установки значения вероятности, равного 1, и C(E:E') = 5 для принудительной установки ее значения, равного нулю. Этому условию соответствует граф (в виде кусочно-линейной функции, представленной на рис.), связывающий C(E:E') с вероятностью P(E), из чего следует:

$$P(E:E') = P(E) + \frac{C(E:E')}{5}(1 - P(E)) \qquad (40)$$

для *C(E:E')>0* и

$$P(E:E') = P(E) + \frac{C(E:E')}{5}P(E) \qquad (41)$$

для $C(E:E) \le 0$

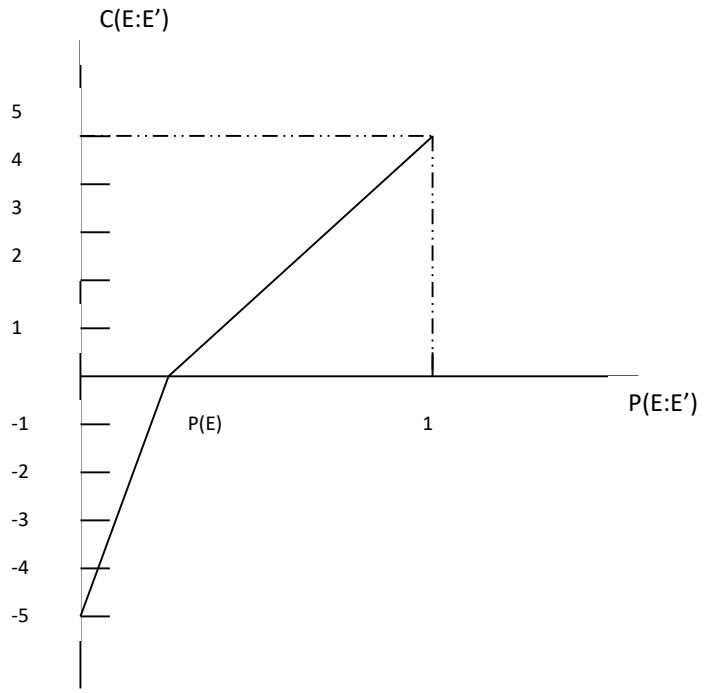

Рис.35– Взаимосвязь между показателями определенности и вероятности

При проявлении вытекающих из правил событий делался вывод о достоверности того или иного правила. Эксперимент продолжался до тех пор, пока количество реализаций по самому редко встречающемуся правилу не достигнет требуемых значений (например, 50). В результате получаем значения вероятности вывода правил при обнаружении соответствующих событий. Для принятия решения о выводе правила устанавливается порог $P(H:E) = 0{,}9$ (см. рис. 36).

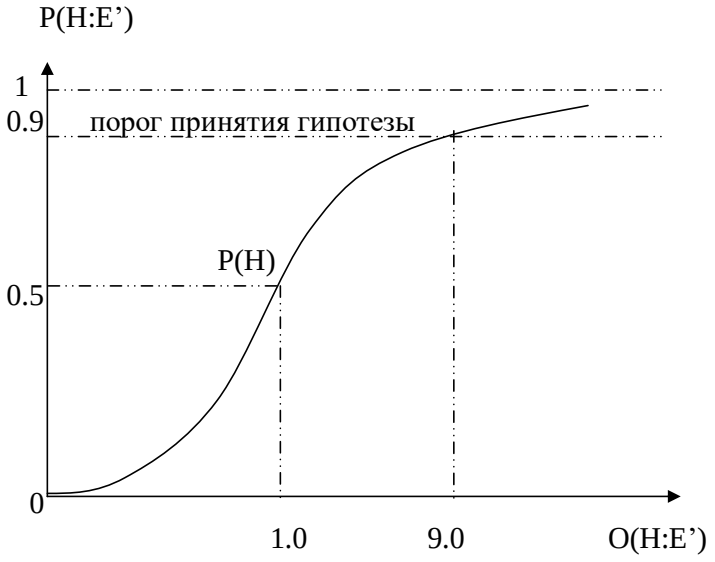

Рис. 36 – Зависимость вероятности гипотез от отношения подтверждающих и опровергающих свидетельств

Таким образом, в процессе осуществления логического вывода описанный выше механизм позволяет учитывать вероятность вывода того или иного конкурирующего правила.

Рассмотрим работу методики на основе логического вывода в сравнении с методикой на основе искусственных нейронных сетей.

Проведем сравнение на трех этапах обучения нейронной сети – в начале обучения, на половине (5000) обучающих наборов и полностью обученной (10 000) сети. Из представленных графиков видно, что искусственная нейронная сеть, за счет априорности действий дает существенный прирост быстродействия, снимая нагрузку с системы на всех трех этапах обучения. Однако необученная сеть показывает значительный процент ошибок (50%). При этом на половинном обучении ошибка существенно снижается (9%). И на полностью обученной сети видно незначительное превосходство точности методики на основе нейронной сети (4%). Кроме того, можно сделать вывод, что за счет свойства обучаемости происходит улучшение свойств нейронной сети.

Для эксперимента нейронная сеть, обученная в реалиях Баренцева моря, была перенесена в условия Берингова моря с сохранением структуры обучающих векторов. По результатам работы на первых 100 наборах ошибка сети составляла около 14%, однако с ростом количества наборов ошибка снижалась и уже на 500 наборах она составила 8% с дальнейшим уменьшением. Данный результат еще раз подтверждает адаптивность и самообучаемость предложенной методики за счет использования аппарата искусственных нейронных сетей.

Выводы по главе 3

1. Для разработки ИНС-методики оценки территориальной ситуации выполнено сравнение различных архитектур ИНС.

2. Доказано, что для обработки ГИ о быстро меняющейся гидрометеорологической и навигационной обстановке в ближней морской зоне

наиболее оптимальной может быть использована такая модель ИНС, как многослойный перцептрон с двумя скрытыми слоями по N нейронов в каждом, N входными нейронами и одним выходным нейроном. Предложенная модель достаточно адекватно и оперативно отображает оперативную территориальную ситуацию в БМЗ.

3. Предложенная нейронная сеть прошла обучение и была протестирована на практической задаче оценки ледовой обстановки в конкретном районе Баренцева моря. Получены частные оценки территориальной ситуации, а также результирующая общая оценка обстановки в виде пространственной зоны (покрытия, области) рекомендованных маршрутов перехода в районе. Вычислена ошибка тестирования, которая составляет не более 1 % отклонения от априорно верного результата на векторах тестового множества наборов. При допустимом значении 2 % это позволяет говорить о работоспособности метода.

3. Определены ограничения предложенного метода: неуниверсальность архитектур нейронной сети (необходимость создавать обучающие наборы для каждого отдельного случая в зависимости от целей оценки и оцениваемых областей), пренебрежение отдельными факторами обстановки, не оказывающими существенного влияния на условия текущей задачи, длительность процедуры обучения, а также зависимость точности первичной работы нейронной сети от качества обучающих наборов [112].

4. Предложенные в первом приближении процедуры топологизации (анаморфирования) геоизображения исследуемого района требуют дальнейшей детализации и доработки как для оптимизации структуры входных наборов исходной ГИ, так и для построения визуальных графических оценок территориальной ситуации.

5. Для проверки работоспособности методики требуется решение примера на конкретной территориальной ситуации. Такой типовой задачей является поиск оптимального маршрута перехода в сложных навигационных условиях.

Глава 4. Методика построения оптимального маршрута с применением ИНС

4.1 Постановка задачи построения маршрута

Для описания методики построения оптимального маршрута рассмотрим конкретную задачу: построение оптимального и безопасного маршрута перехода судна из пункта А (Бугрино) в пункт В (Варнек) в условиях меняющейся ледовой обстановки с учётом дополнительных параметров территориальной ситуации (течения, ветер, осадки, видимость, экологическая безопасность, навигационные опасности, интенсивность судоходства и др.)

Рис. 37. Географическая карта расположения исходного и конечного пунктов

Методика основана на работе искусственных нейронных сетей и оценке обстановки в ближней морской зоне. В связи с этим, необходимо осуществить

постановку задачи в условиях модели представления обстановки в ближней морской зоне.

Параметры модели представляют собой совокупность данных, полученных из открытых источников, размещенных на геопорталах map.openseamap.org и gis.adm-nao.ru (рисунок 37).

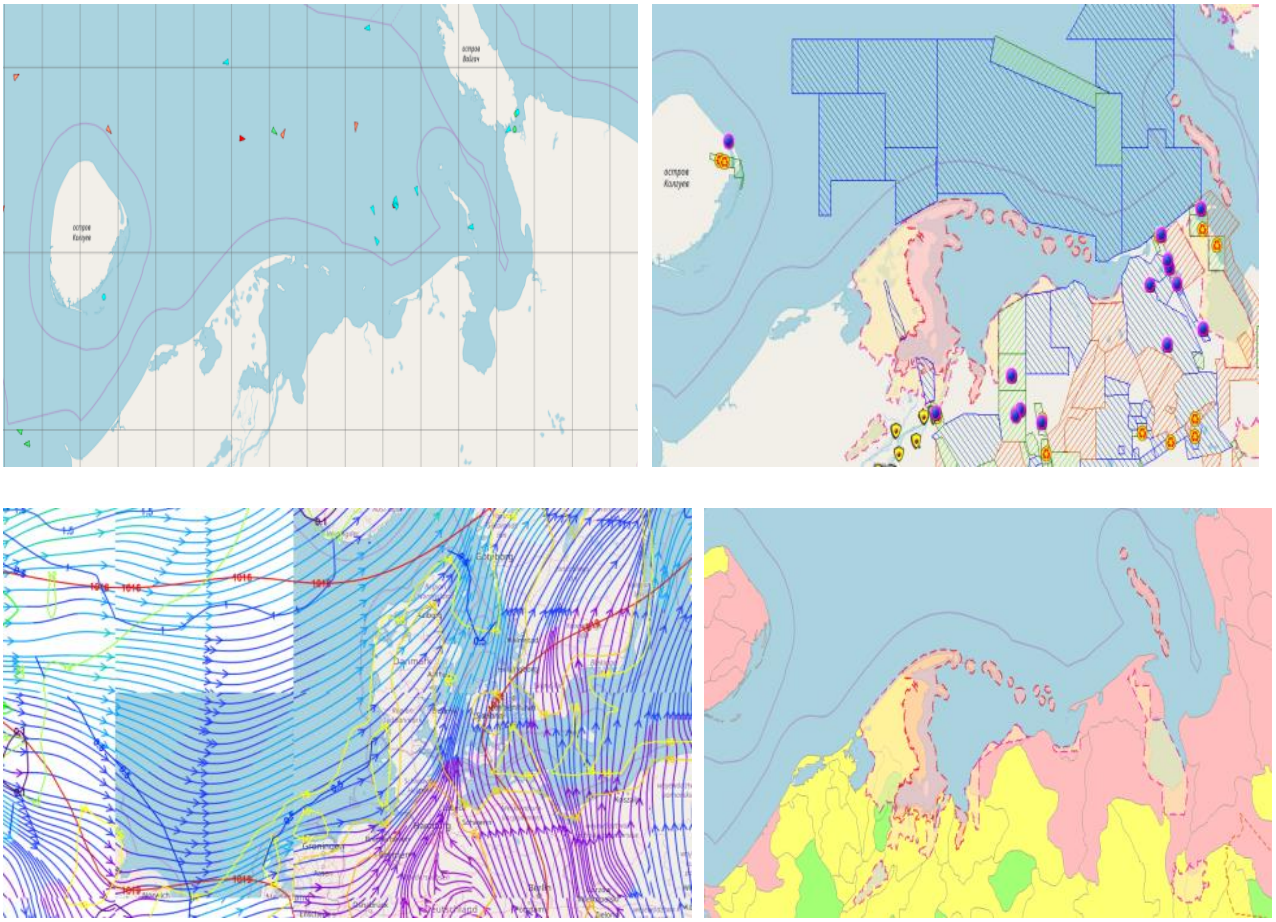

Рис.38. территориальное распределение (карта) параметров обстановки в ближней морской зоне

Полученные векторы параметров обобщаются в виде таблицы данных, которая затем будет подана для обучения искусственной нейронной сети (рис. 38). Запишем необходимый функционал для оценочной нейронной сети в текущей задаче.

4.1.1 Математическое описание и обработка поставленной задачи в условиях аппарата искусственной нейронной сети

Рассмотрим дискретную систему, состоящую из векторов, физический смысл которых описан в главе 3. На каждом шаге k обозначим их как набор $x^{k-\nu}$, $x^{k-\nu+1}$, x^k и введем вектора управления u^{k-p}, u^k. Тогда мимнимизируемый функционал примет вид:

$$I([x][u]) = \sum_{(k=0)}^{(q-1)} f_k^0(x^{(k-n)},...,x^k,u^{(k-m)},...,u^k) + \Phi(x^q); \quad (42)$$

Переход из k-ого состояния в k+1 описывается рекуррентными соотношениями

$$x^{k+1}=f_k(x^{k-n},\ldots,x^k,u^{k-m},\ldots,u^k), k=\overline{0,q-1}; \quad (43)$$

$$x^k=b^k\in \mathbb{R}^l, k=-\nu,\ldots,0; \quad (44)$$

$$u^k=u^k\in \mathbb{R}^r, k=-p,\ldots,0; \quad (45)$$

с заданными начальными условиями $z^k=(x^{k-n},\ldots,x^k,u^{k-m},u^k)$; $[u]=[u^0,\ldots,u^{q-1}]$;

$[\tilde{u}]=[u^0,\ldots u^{i-1},u^i+\Delta,u^{i+1},\ldots,u^{q-1}]$; $[x]=[x^0,\ldots,x^q]$; $[\tilde{x}]=[x^0,\ldots x^{i-1},x^i+\Delta,x^{i+1},\ldots,x^q]$.

Дадим приращение вектору $x^i+\Delta$, тогда последующие векторы $\tilde{x}^{i+1},\ldots,\tilde{x}^q$ вычисляются согласно формуле (2),

$$x^{i+1} = f_i(x^{i-n},\ldots,x^{i-1},x^i+\Delta,\ldots,x^{q-1})$$

Определим сопряженный вектор

$$p^i = \frac{dI([x],[u])}{dx^i} = \lim_{\Delta\to 0}\frac{I([\tilde{x}][\tilde{u}])-I([x][u])}{\Delta} \quad (46)$$

Из (5) следует, что если i=q, то имеет место равенство

$$p^q = \frac{dI([x][u])}{dx^q} = \frac{\partial\Phi(x^q)}{\partial x^q}; \quad (47)$$

если i=q-1, то

$$p^{q-1} = \frac{\partial I}{dx^{q-1}} + (\frac{\partial x^q}{\partial x^{q-1}})^T\frac{dI}{dx^{q-1}} = \frac{\partial}{\partial x^{q-1}}\left[f_{q-1}^0(z^{q-1}) + \left(p^q, f_{q-1}(z^{q-1})\right)\right]. \quad (48)$$

В общем случае имеем

$$p^i = \frac{\partial I}{\partial x^i} + \left(\frac{\partial x^{i+1}}{\partial x^i}\right)^T \frac{\partial I}{\partial x^i} + \ldots + \left(\frac{\partial x^{i+\nu+1}}{\partial x^i}\right)^T \frac{dI}{dx^{i+\nu+1}} = \frac{\partial f_i^0\left(z^i\right)}{\partial x^i} + \ldots$$

$$+ \frac{\partial f_{i+\nu}^0\left(z^{i+n}\right)}{\partial x^i} + \left[\frac{\partial f_i(z^i)}{\partial x^i}\right]^T p^{i+1} + \ldots\ldots + \left[\frac{\partial f_{i+\nu}^0\left(z^{i+\nu}\right)}{\partial x^i}\right] p^{i+n+1} = \tag{49}$$

$$\sum_{l=0}^{n} \frac{\partial}{\partial x^i}\left[\partial f_{i+l}^0\left(z^{i+l}\right) + \left(p^{i+l+1}, f_{i+l}\left(z^{i+l}\right)\right)\right], \; i = \overline{1, q-1}, \; p^j = 0, \; j > q.$$

Введем функцию $H_k(z^k, p^{k+1}) = \lambda_0 f_k^0(z^k) + (p^{k+1}, f_k(z^k))$тогда (8) с помощью этой функции преобразуется следующим образом

$$p^i = \sum_{l=0}^{\nu} \frac{\partial H_{i+l}\left(z^{i+l}, p^{i+l+1}\right)}{\partial x^i}, \; i = \overline{q-1,1}; \tag{48}$$

Множество векторов [x]=[x^1,…,x^q]зависит от начального набора состояния x^0 и допустимых векторов [u]=[u^0,..,u^{q-1}], или [x]=[x(u)] и минимизировать сложную функцию I=I([x(u)],[u]) аргумента u. Полная производная функции которой по вектору u^i определяется выражением

$$\frac{dI\left(\left[x(u)\right],[u]\right)}{du^i} = \frac{dI\left(\left[x(u),[u]\right)\right.}{du^i} + \sum_{l=0}^{p} \frac{\partial x^{i+l+1}[u]}{\partial u^i} \frac{dI\left(\left[x(u)\right],[u]\right)}{dx^{i+l+1}} =$$

$$\sum_{l=0}^{p} \frac{\partial f_{i+l}^0\left(z^{i+l}\right)}{\partial u^{i+l}} + \left[\frac{\partial f_{i+l}^0\left(z^{i+l}\right)}{\partial u^{i+l}}\right]^T p^{i+l+1} \tag{51}$$

$$= \sum_{l=0}^{p} \frac{\partial}{\partial u^i} H_{i+l}\left(z^{i+l}, p^{i+l+1}\right)$$

Функция Лагранжа для дискретной задачи оптимального управления (1)-(4) имеет вид

$$\mathrm{L}([x],[u],[p],\lambda_0) = \sum_{i=0}^{q-1}\left[\lambda_0 f_i^0\left(z^i\right) + \left(p^{i+1}, f_i\left(z^i\right) - x^{i+1}\right)\right] + \lambda_0 \Phi\left(x^q\right).$$

Чтобы получить необходимые условия оптимальности, вычислим производные функции Лагранжа по переменным x^i, $i=\overline{1,q}$, u^i, $i=\overline{1,q-1}$:

$$\frac{\partial L}{\partial x^i}=\sum_{l=0}^{\nu}\frac{\partial H_{i+l}\left(z^{i+l},p^{i+l+1}\right)}{\partial x^i}-p^i=0,\ i=\overline{1,q-1},\frac{\partial L}{\partial x^q}=_0\frac{\partial\Phi\left(x^q\right)}{\partial x^q}-p^q=0,$$
$$\frac{\partial L}{\partial u^i}=\sum_{l=0}^{p}\frac{\partial H_{i+l}\left(z^{i+l},p^{i+l+1}\right)}{\partial u^i}=0,\ i=\overline{0,q-1}. \quad (51.1)$$

Сравнивая формулы (51) и (51.1), получим следующее равенство:

$$\frac{\partial L}{\partial u^i}=\frac{dI\left(\left[x(u)\right],[u]\right)}{du^i},i=\overline{0,q-1} \quad (52)$$

Формула (51.1) может использоваться для получения приближенного оптимального решения, например, градиентным методом.

Если в задаче (42)-(45) присутствуют ограничения на вектор управления $u^i \in U_i$, $\overline{0,q-1}$, например

$$U_i=\left\{v\in\mathbb{R}^r: g_l^i(v)\le 0, l=\overline{1,k}, h_l^i(v)=0, l=\overline{k+1,s}\right\}, \quad (53)$$

То можно использовать метод Лагранжа с новой функцией Лагранжа. В этом случае новая функция Лагранжа $\tilde{L}$ для задачи (1)-(4) определяется выражением

$$\tilde{L}=L+\sum_{i=0}^{q-1}\left[\sum_{l=0}^{k}\mu_l^i\ g_l^i(v)+\sum_{l-k+1}^{s}v_l^i h_l^i(v)\right]$$

Для решения задачи (1)-(4) в этом случае можно использовать метод штрафных функций [3].

Рассмотрим модель нейронной сети, в которой присутствует запаздывание по вектору состояния, а управляющими функциями являются весовые коэффициенты нейронной сети[4].

Пусть переход из k-ого состояния в k+1 осуществляется по следующему правилу:

$$x^{k+1}=h(x^k,x^{k-\nu})+(W_0+W_k)g(x^{k-\nu}) \quad (54)$$

или в покомпонентном виде –

$$x_i^{k+1} = h_i(x^k, x^{k-\nu}) + \sum_{j=1}^{n}\left[\omega_{ij}^0 + \omega_{ij}^k\right] g_j(x^{k-\nu}), \qquad i = \overline{1,n} \qquad (55)$$

с начальными условиями $x^i=a^i$, i=-ν,…,0; здесь $a^i \in \mathbb{R}^n$, i=-ν,…,0– заданные векторы в $\mathbb{R}^n$, h, g- заданные n–мерные векторные непрерывно дифференцируемые функции, W_0- заданная постоянная матрица, $W_k=\{\omega_{ij}^k\}, i,j = \overline{1,n}$ - n×n- матрица весовых коэффициентов или управлений на k – ом шаге, принимающих значения в заданном множестве, например,

$$|\omega_{ij}^k| \le A_{ij}^k, k = 0,\dots q-1; i,j = \overline{1,n}. \qquad (56)$$

Весовые коэффициенты ω_{ij}^k выбираются из условия минимума функции

$$I(\omega) = \Phi\left(x^q\right) + \sum_{k=0}^{q-1} E_k\left(x^k, x^{k-\nu}\right) + \frac{1}{2}\sum_{k=0}^{q-1}\sum_{i,j=1}^{n} r_{ij}^k\left(\omega_{ij}^k\right)^2. \qquad (57)$$

В выражении (16) коэффициенты $r_{ij}^k > 0$ при каждом $k = \overline{0,q-1}$ образуют положительно определенную матрицу, которую мы обозначим через R_0, $\omega_{ij}^0, \omega_{ij}^q$ – набор n×n матриц, $E_k\left(x^k, x^{k-\nu}\right)$ - заданные скалярные функции, $x^{k-\nu} = (x^{k-\nu_1}{}_1, \dots, x^{k-\nu_n}{}_n)$ то есть каждая компонента вектора $x^{k-\nu}$ имеет свое запаздывание.

В модели (14) – (17) оптимальные весовые коэффициенты $\bar{\omega}_{ij}^k$ определяются равенством

$$\bar{\omega}_{ij}^k = p_i^{k+1}\left[r_{ij}^k\right]^{-1} g_j\left(x^{k-\nu}\right) \qquad (58)$$

сопряженные векторы удовлетворяют следующими рекуррентными соотношениями:

$$p_i^k =_0 \left[\frac{\partial E_k\left(\bar{x}^k, \bar{x}^{k-\nu}\right)}{\partial x_i^k} - \frac{\partial E_{k+\nu}\left(\bar{x}^{k+\nu}, \bar{x}^k\right)}{\partial x_i^k}\right] + p_i^{k+1}\frac{\partial h_i\left(x^k, x^{k-\nu}\right)}{\partial x_i^k}$$
$$+ p_i^{k+1+\nu}\left[\frac{\partial h_{i+\nu}\left(x^{k+\nu}, x^k\right)}{\partial x_i^k} + \sum_{j=1}^{n}\omega_{ij}^{k+1}\frac{\partial g_j}{\partial x_i^k}\left(x^k\right)\right]. \qquad (59)$$

с граничными условиями:

$$p_i^q = \lambda_0 \frac{\partial \Phi\left(\overline{x^q}\right)}{\partial x_i^q},\ i = \overline{1,n};\ p_i^s = 0,\ s > q; i = \overline{1,n}.$$

4.2 Описание методики в условиях поставленной задачи

В качестве входных данных будет выступать модель обстановки в ближней морской зоне, оптимизированная для работы с ИНС, с нанесенными на нее исходной и конечной точками маршрута. Данная модель получается топологическим переходом от географической карты к картоиду, аналогичным подходу, описанному в методике оценки обстановки в ближней морской зоне.

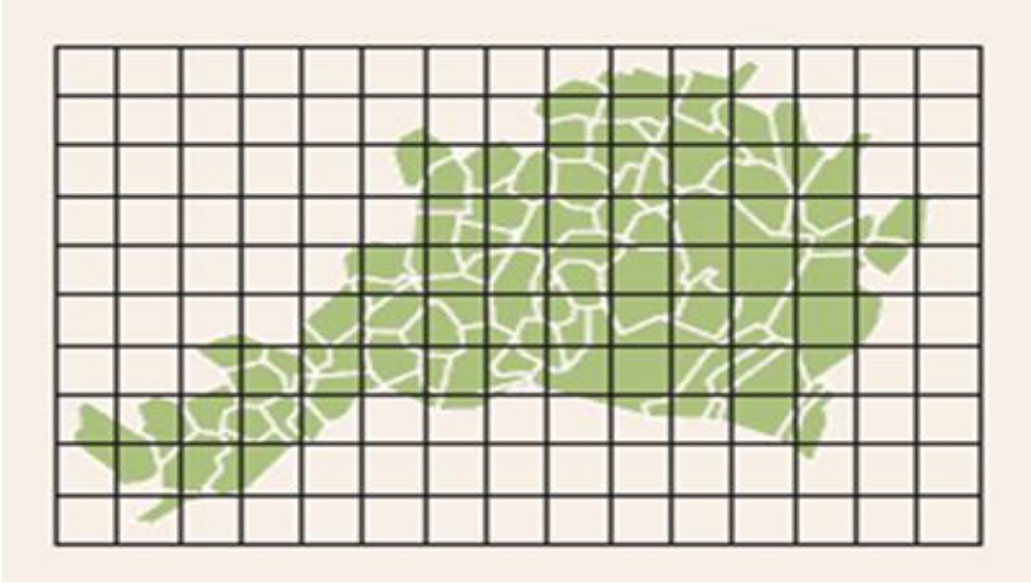

Рис.39. Преобразованный картоид

Важным моментом данного этапа является нанесение координатной сетки (рисунок 39). От этого зависит точность последующего анаморфирования.

В основе дальнейших преобразований описываемой методики лежит каскад искусственных нейронных сетей одинаковой архитектуры: рекуррентная нейронная сеть. Модель подается на вход только первой нейронной сети. Вторая сеть получает на вход результаты работы первой сети, то есть преобразованный картоид оценки обстановки.

Вторым этапом методики является проведение оценки обстановки, согласно перечисленным входным векторам и условиям задачи. Математические преобразования, производимые нейронной сетью, приведены в главе 4.1.1. Однако, в отличие от методики оценки обстановки в ближней морской зоне, в целях ускорения работы системы данный этап проводится только в режиме изменения размерности и не имеет этапа финального контроля человеком. Дополнительно следует понизить разрешение графической составляющей, так как на данном этапе она не важна, но ее обработка может занять существенную часть ресурсов и повлечь за собой увеличение времени обработки данных.

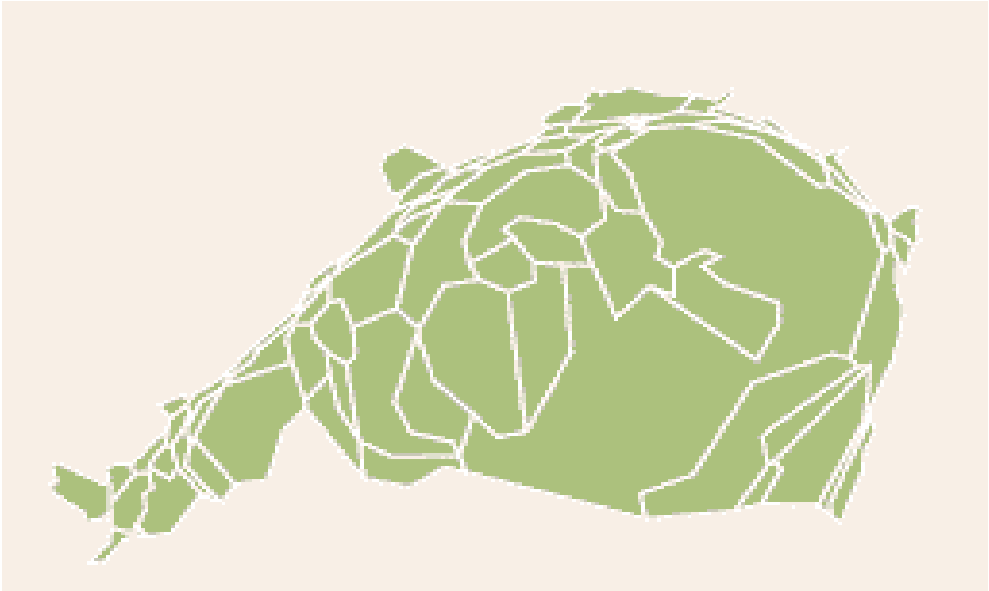

Рис.40. Анаморфированный картоид

Результирующий картоид подается на вход второй нейронной сети. Согласно проведенным экспериментам нейронная сеть имеет архитектуру многослойного перцептрона с обратной связью, т.е. рекуррентной нейронной сети. Данная нейронная сеть является второй частью каскада нейронных сетей, лежащего в основе данной методики. На вход сети подаются результаты работы первой нейронной сети, то есть анаморфированный картоид оценки обстановки в ближней морской зоне (рис. 40).

Обучение второй нейронной сети также осуществляется с помощью алгоритма обратного распространения ошибки, однако существенно изменена структура

обучающего набора. В данной ситуации оцениваются не каждая область по отдельности, а все области целиком. То есть обучающее множество состоит из матрицы, строками которой являются соответствующие области, на которые разбит картоид, а столбцами входные значения: удаленность от кратчайшего маршрута и оценка обстановки в данном регионе, а также выходное значение, принимающее значения 0 и 1 – соответственно маршрут не проходит либо проходит через рассматриваемую область.

Таблица 5

Фрагмент матрицы из обучающего набора ИНС, выбирающей оптимальный путь

Область	Входные параметры			Маршрут проходит
	Оценка	Близость к оптимальному маршруту	Близость к текущему маршруту	
1	8	2	1	0
2	6	5	0	0
3	1	3	1	1
4	4	3	1	1

В силу того, что нужно оценивать карту целиком, для обучения нейронной сети понадобится большее количество обучающих наборов (20 000 для рассматриваемой задачи). По результатам работы нейронная сеть выстраивает ломаную линию, проходящую из исходного пункта маршрута в конечный пункт. Чем мельче разбиение на области, тем точнее будет маршрут. Однако, делая более мелкое разбиение, не следует забывать, что сложнее становится и описываемая задача с точки зрения обработки данных, и тем больше машинных ресурсов требуется. В качестве вариантов оптимизации и повышения точности выстраиваемого маршрута возможны варианты использования распараллеливания вычислительных процессов путем переноса их на

графические процессоры (видеокарты), либо использование дополнительного алгоритма для построения маршрута в каждой из рассматриваемых областей (рисунок 42). Второй вариант улучшения методики приведен в приложении 2.

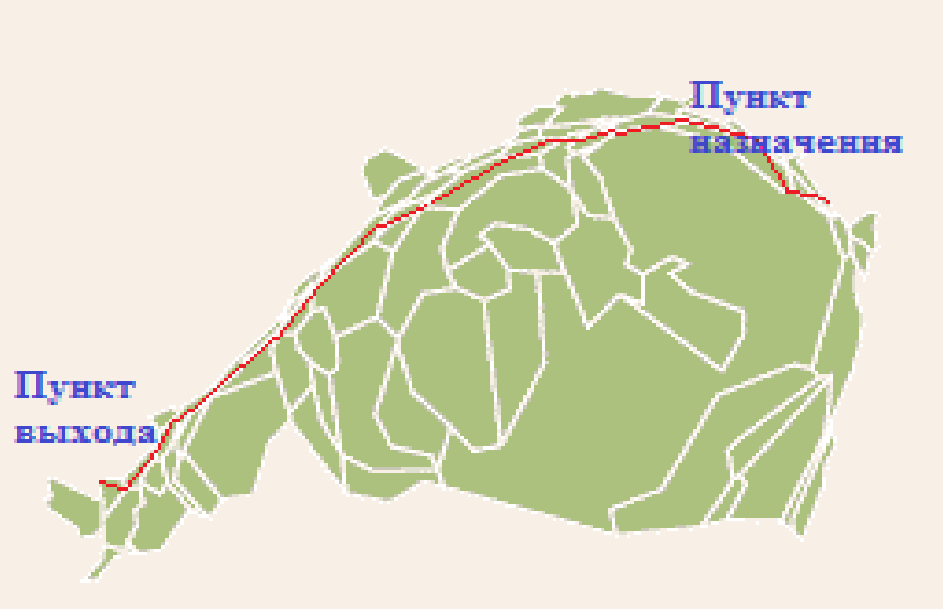

Рис.41. Картоид с проложенным маршрутом

Затем с учетом заданной ранее координатной сетки осуществляется процесс детопологизации, то есть перехода от графического картоида к географической карте. Стоит отметить, что при использовании классических моделей на основе shape-файлов подобный переход не является возможным. При этом он составляет важный шаг методики, позволяя выстроить маршрут в условиях реальной карты.

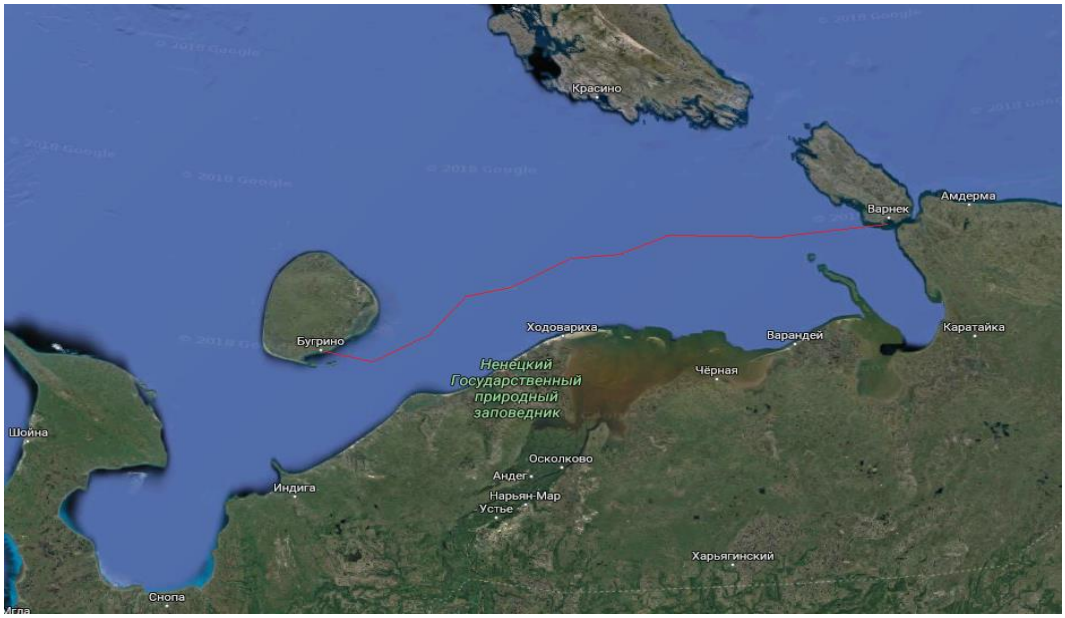

Рис.42. Результат работы методики построения маршрута

Результатом работы методики является карта с выделенными исходным и конечным пунктами и построенным между ними маршрутом, приведенная на рис. 42.

4.3 Анализ зависимости точности построения маршрута от степени обученности нейронной сети

Оптимальность построенного маршрута во многом зависит от количества обучающих наборов, и соответственно этапа обучения нейронной сети. На рисунке ниже приведены маршруты, построенные нейронной сетью на различных этапах ее обучения в условиях поставленной задачи.

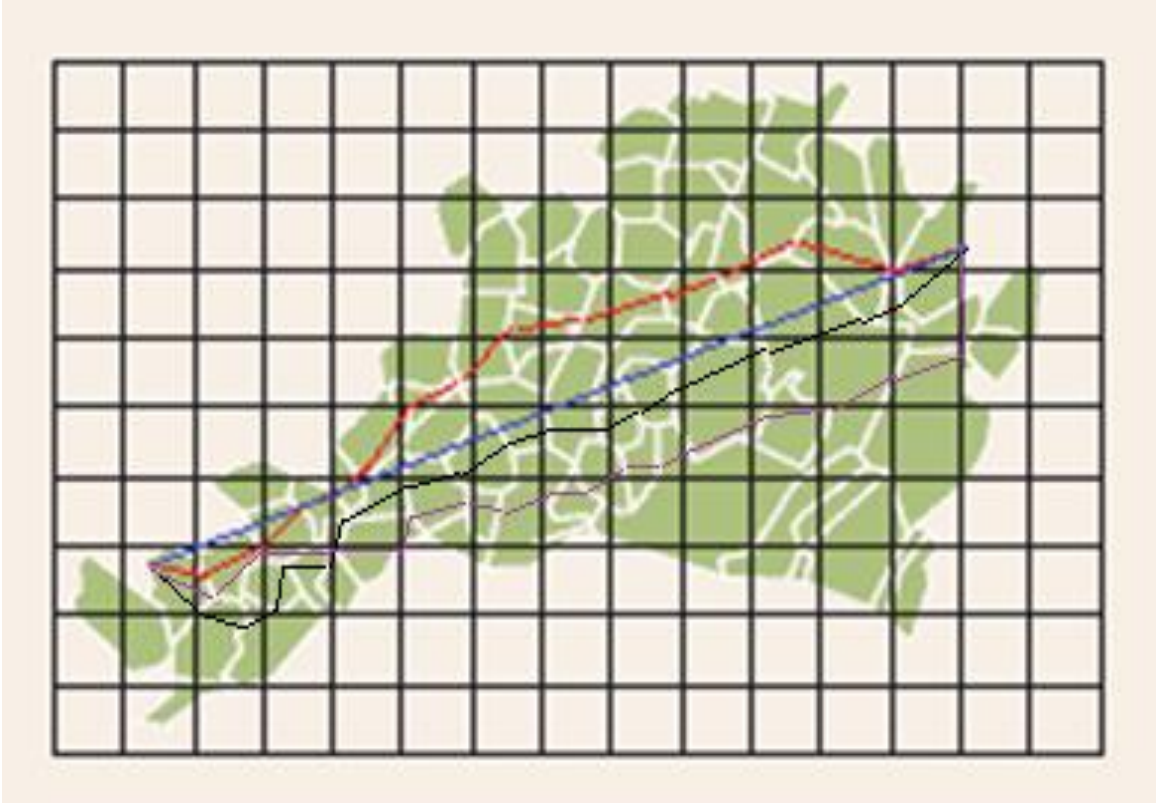

Рис.43. Промежуточные результаты работы методики построения маршрута

Синей линией обозначен кратчайший путь между пунктами А и Б**.**

Сиреневой линией обозначен маршрут, построенный необученной нейронной сетью. На рисунке видно, что маршрут не является безопасным (то есть пролегает по областям с большой плотностью льда). Кроме того, данный маршрут далек от оптимального.

Чёрной линией выделен маршрут, построенный сетью, обученной на половине наборов (10 000 в условиях поставленной задачи). Видно, что он становится ближе к кратчайшему, но при этом всё еще проходит через опасные области с оценкой обстановки выше 6. Однако заметнызначительные улучшения.

Красной линией представлен маршрут, выстроенный полностью обученной (20 000 в условиях поставленной задачи). В среднем он более удален от кратчайшей линии, нежели чёрная линия, однако проходит через относительно безопасные области, что отвечает условиям поставленной задачи.

Таким образом можно сделать вывод, что маршрут, построенный обученной нейронной сетью, отвечает условиям поставленной задачи. При это не всегда

кратчайший маршрут будет соответствовать данным условиям, как видно на приведенном примере. Нейронная сеть учитывает все параметры и проводит всесторонний анализ, поэтому построенный ею маршрут удовлетворяет конкретным условиям и ограничениям.

Дополнительно был проведен эксперимент с переносом полностью обученной нейронной сети в иные условия. Был рассмотрен вариант построения оптимального маршрута с переносом на реалии Карского моря, с сохранением начальных условий – рассматривается ледовая обстановка в ближней морской зоне. При первом запуске нейронной сети точность маршрута соответствовала примерно обучению приблизительно на половине наборов. Однако процент ошибки падал значительно быстрее, чем при первичном обучении.

Предложенная методика построения маршрута на основании оценки обстановки в ближней морской зоне отличается наличием дополнительных процедур топологизации для поиска решений в географически абстрактной среде и детопологизации первичного решения для адаптации его в географически конкретной обстановке с применением аппарата ИНС.

Выводы по главе 4

1. Для проверки работоспособности ИНС-оценок территориальной ситуации разработан каскад нейронных сетей, который позволяет реализовывать методику поиска оптимального на основании оценки осбтановки в БМЗ.
2. Реализованы и проанализированы различные варианты архитектур нейронных сетей,сделан выбор в пользу рекуррентных нейронных сетей с обратной связью.
3. Предложенные нейронные сеть прошла обучение и была протестирована на практической задаче построения маршрута в условиях динамически изменяющейся ледовой обстановки в конкретном районе Баренцева моря [44,45,112].
4. ИНС-методика обеспечила прирост скорости построения маршрута на 8% в сравнении с традиционными методами (например логического

построения), кроме того снизила до 12% нагрузку на аппаратные ресурсы систем. Точность построения в сравнении с традиционными маршрутами выозросла на 1,2%.

Заключение

По результатам проделанной работы можно сделать следующие выводы:

- Выполнен анализ текущего состояния проблемы – выявлены проблемные моменты, такие как быстродействие, многофакторность и субъективность оценок
- Создана и описана модель обстановки в ближней морской зоне, оптимизированная для последующего анализа с применением аппарата ИНС
- Разработана математическую модель для последующей реализации методики оценки обстановки в ближней морской зоне
- Произведен отбор и последующее сравнение нескольких реализаций архитектур ИНС по результатам которого выявлена оптимальная архитектура для поставленной задачи - нейронная сеть типа многослойный перцептрон, с 2 скрытыми слоями по N нейронов в каждом, N входными нейронами и 1 выходным нейроном.
- Проведен анализ эффективности предложенной методики, по результатам которого выявлено, что искусственная нейронная сеть, за счет априорности действий дает существенный прирост быстродействия, снимая нагрузку с системы на всех трех этапах обучения. Однако необученная сеть показывает значительный процент ошибок (50%). При этом на половинном обучении ошибка существенно снижается (9%). И на полностью обученной сети видно незначительное превосходство точности методики на основе нейронной сети (4%).
- Предложены и протестированы варианты практического применения методики в алгоритмах построения оптимального по указанному параметру маршрута судна в ближней морской зоне

В процессе работы были получены следующие научные результаты:

1. Разработана топологическая модель представления обстановки в ближней морской зоне, основанная на анаморфировании, и оптимизированная для работы с искусственными нейронными сетями. Научная новизна результата состоит в том, что предложенная модель геосреды (обстановки) отличается топологическим переходом от географически конкретного представления территориальной ситуации к пространственно-абстарктной анаморфозе (картоиду), что позволяет формировать наборы исходных геоданных для работы (обучения) ИНС.

2. На основании разработанной модели создана методика оценки обстановки в ближней морской зоне, основанная на работе искусственных нейронных сетей и анаморфированном представлении обстановки. Научная новизна результата заключается в применении специально спроектированных и обученных на оригинально сформированных априорных наборах геоданных ИНС, что позволяет повысить быстродействие систем и снизить нагрузку на аппаратные ресурсы, а так же топологизации результатов территориальных оценок, что позволяет более наглядно отображать проблемные зоны геосреды и упрощать процессы оптимизации решений на конкретной геоситуации (за счёт снижения размерности пространства обстановки).

3. Представлена и апробирована методика построения маршрута на основании оценки обстановки в ближней морской зоне, реализованная с применением каскада настраиваемых искусственных нейронных сетей. Представленная методика построения маршрута отличается наличием дополнительных процедур топологизации для поиска решений в географически абстрактной среде и детопологизации первичного решения для адаптации его в географически конкретной обстановке с применением аппарата ИНС.

Список литературы

1. Hopfield J. J. Neural networks and physical systems with emergent collective computational abilities// Proc. Natl. Acad. Sci. 1984. V. 9. P. 147-169.
2. Hopfield J. J., Feinstein D. I., Palmer F. G. Unlearning has a stabilizing effect in collective memories./ Nature. 1983. V. 304. P. 141-152.
3. Hopfield J. J., Tank D. W. Neural computation of decision in optimization problems/J Biol. Cybernet. 1985. V. 52. P. 141-152.
4. Joey Rogers, Object-Oriented Neural Network in C++, Academic Press, San Diego, CA, 1997
5. M.T. Hagan, H.B. Demuth and M.H. Beale, Neural Network Design, PWS Publishing, Boston, MA, 1995
6. MATLAB. Getting Started with MATLAB. Version 5. The MathWorks, Inc., 1998. — 70 p.
7. MATLAB. Release 11 New Features. The MathWorks, Inc., 1999.
8. MATLAB. Using MATLAB.The MathWorks, Inc., 1999.
9. Mattbew Rapaport. // Object-Oriented Data Bases: The Next Step in DBMS Evolution // Сотр. Lang.- 5, N 10,- 1988. p. 91-98
10. Minsky M. L. Steps toward artificial intelligence // Proceedings of the Institute of Radio Engineers. — 1961. — N 49. — P. 8 30.
11. MinskyM. L. Theory of neural-analog reinforcement systems and its application to the brain-model problem: Ph. D. Thesis. — Princeton: Princeton University, N. J., 1954. — 143 p.
12. Raster Imagery in GIS // Практическое руководство по работе с растровыми данными в ГИС.
13. Rochester N., Holland J. H., Haibt L. H., Duda W. L. Tests on a cell assembly theory of the action of the brain, using a large digital computer // IRE Transactions on Information Theory. — 1956. — NIT-2. — P. 80 93.
14. Rosenblatt F. The Perceptron: A probabilistic model for information storage and organization in the brain // Psychological Review. — 1958. — N 65. — P. 386 - 408.
15. Rosenblatt F. The peseptron: a probabilistic model for information storage and organization in the brain)/ Psychol. Rev. 1958. V. 65.
16. Taylor D. R. F. // A conceptual basis for cartography / new directions for the information era. — Cartographica, 1991. — Vol. 28. № 4., p. 1 — 8.
17. Uttley A. M. A theory of the mechanism of learning based on conditional probabilities // Proc. of the 1st International Conference on Cybernetics. — Paris: Namur, Gauthier-Villars, 1956. — P. 83 92.
18. Uttley A. M. Information Transmission in the Nervous System. — London: Academic Press, 1979. — 215 p.
19. Winograd S., Cowan J. D. Reliable Computation in the Presence of Noise. — Cambridge, MA: MIT Press, 1963. — 247 p.
20. Won Kim // Object-Oriented Databases: Definition and Research Directions // IEEE Trans. Data and Knowledge Eng.- 2, N 3.- 1990.- p.327-341.
21. Андреева Е. А. Оптимальное управление динамическими системами. Тверь: ТвГУ, 1999.

22. Андреева Е. А. Оптимизация искусственной нейронной сети// Применение функционального анализа в теории приближений: Сб. научн. тр. Тверь: ТвГУ, 1996. С. 7-12.
23. Андреева Е. А., Бенке Х. Оптимизация управляемых систем. Тверь: ТвГУ, 1996.
24. Андреева Е. А., Пустарнакова Ю. А. Оптимизация нейронной сети с запаздыванием. Ч. 1.J. Применение функционального анализа в теории приближений: Сб. научн. тр. Тверь: ТвГУ, 2000. С. 14-30.
25. Андреева Е. А., Пустарнакова Ю. А., Семыкина Н. А. и др. Модели управляемых систем. Тверь: ТвГУ, 1999. Ч. I, II.
26. Андреева Е.А., Храмов И.С. Биологические нейронные сети. Актуальные направления научных исследований XXI века: теория и практика. 2017. Т. 5. № 10 (36). С. 32-39.
27. Андреева Е.А., Храмов И.С. Алгоритм построения приближенного оптимального решения задач на основе искусственных нейронных сетей с учетом запаздывания. Информационные технологии. 2017. – Т. 23. – № 12. – С. 904-909.
28. Андреева Е.А., Храмов И.С. Применение нейронной сети для предсказания результатов футбольных матчей. В сборнике: Математические методы управления сборник научных трудов. Тверь, 2015. С. 19-25.
29. Андрейчиков А.В., Андрейчикова О.Н. Анализ и синтез. Планирование решений. // Финансы и статистика № 131 М.:, 2002 г.
30. Андрианов В. 10. Англо-русский толковый словарь по геоинформатике. М.: Дата+, 2001 г. -122 с.
31. Антонов А.В. Системный анализ М.: Высшая школа. 2004 г. 453 с.
32. Арманд А.Д. Самоорганизация и саморегулирование географических систем М.: Наука. 1988 г. 242 с.
33. Арманд Д.Л. Информационные модели природных комплексов. М.:Наука. 1975 г.
34. Асланикашвили А.Ф. Метакартография. Тбилиси: Просвещение, 1974 г.
35. Берлянт, А.М. Геоиконика М.: Акад. естественных наук РФ, изд-во Астрея, 1996 г. 224 с.
36. Бехтерева Н. П. Здоровый и больной мозг человека. Л.: Наука, 1980.
37. Биденко С.И., Бородин Е.Л., Травин С.В., Кратович П.В., Храмов И.С. Экспертная ГИС-поддержка управления морской территориальной активностью. Информационные технологии и системы: управление, экономика, транспорт, право. 2017. № 2 (20). С. 77-83.
38. Биденко С.И., Бородин Е.Л., Травин С.В., Кратович П.В., Храмов И.С. Структура ГИС-поддержки управления морской территориальной активностью. Информационные технологии и системы: управление, экономика, транспорт, право. 2017. № 2 (20). С. 64-68.
39. Биденко С.И., Бородин Е.Л., Травин С.В., Хекерт Е.В., Храмов И.С. Геоинформационная поддержка управления морской транспортной активностью: методический аспект. Эксплуатация морского транспорта. – Новороссийск: ГМУ им. адм. Ф.Ф.Ушакова. – 2018. – № 2. – С. 80 – 95.

40. Биденко С.И., Бородин Е.Л., Храмов И.С. Анаморфирование карты обстановки как элемент управления морской территориальной активностью. // Эксплуатация морского транспорта. – Новороссийск: ГМУ им. адм. Ф.Ф.Ушакова, 2019. – № 1. – С. 89 – 102.
41. Биденко С.И., Бородин Е.Л., Храмов И.С. Оценка обстановки в ближней морской зоне с использованием искусственных нейронных сетей. // Эксплуатация морского транспорта. – Новороссийск: ГМУ им. адм. Ф.Ф.Ушакова, 2018. – № 4. – С. 82 – 90.
42. Биденко С.И., Храмов И.С. Оценка тактической обстановки в районах интенсивной морской активности флота с использованием аппарата искусственных нейронных сетей. Сборник научных трудов «Проблемы обороноспособности и безопасности». – М.: ФГБНУ «Экспертно-аналитический центр», 2018. – Вып. 19. – С. 135 – 143.
43. Биденко С.И., Храмов И.С. Применение аппарата нейронных сетей в задачах поддержки безопасного маневрирования в районах интенсивной морской активности флота. Сборник научных трудов «Проблемы обороноспособности и безопасности». – М.: ФГБНУ «Экспертно-аналитический центр», 2018. – Вып. 19. – С. 127 – 134.
44. Биденко С.И., Храмов И.С. Топологизация геоизображения района интенсивной морской активности флота при оценке тактической обстановки. Сборник научных трудов «Проблемы обороноспособности и безопасности». – М.: ФГБНУ «Экспертно-аналитический центр», 2018. – Вып. 19. – С. 144 – 152.
45. Биденко С.И., Храмов И.С. Топологические преобразования аналитических карт местности в аспекте ИНС-оценок района морской территориальной активности. Сборник научных трудов «Проблемы обороноспособности и безопасности». – М.: ФГБНУ «Экспертно-аналитический центр», 2018. – Вып. 20. – С. 242 – 255.
46. Биденко С.И., Храмов И.С., Шилин М.Б. Оценка территориальной ситуации с использованием искусственных нейронных сетей. // Ученые записки Российского государственного гидрометеорологического университета. – СПб: РГГМУ, 2019. – Вып. 54. – С. 38 – 49.
47. Биденко С.И., Панамарев Г.Е. Геоинформационная поддержка управления сложными территориальными объектами и системами. Новороссийск: Изд-во МГА, 2011. 202 с.
48. Блум Ф., Лейзерсон А., Хофстефтер JL Мозг, разум и поведение. М.: Мир, 1988.
49. Болтянский В. Г. Математические методы оптимального управления. М.: Наука, 1966.
50. Брюхомицкий Ю.А. Нейросетевые модели для систем информационной безопасности. Учебное пособие. – Таганрог: Изд-во ТРТУ, 2005. – 160 с
51. Вагизов М.Р. Инвентаризация лесов на основе обработки технологий интеллектуального анализа геоданных. // Материалы научно-технической конференции - Леса России: политика, промышленность, наука, образование. / Том 1/Под.ред. В.М.Гедьо, Спб.:СПБГЛТУ, 2018 г.-224с. -С. 17-20.

52. Вагизов М.Р. Навалихин С.В. Баенгуев Б.А. Разработка геоинформационной системы благоустройства зелёных насаждений общего пользования г. Санкт-Петербурга. // «Фундаментальные исследования» - 2017. № [11] стр.35-40.
53. Вагизов М.Р. Проектирование интеллектуальной геоинформационной системы с анализом геоданных при возникновении чрезвычайных ситуаций. // Информационные системы и технологии: теория и практика: сб. Научн. тр. Вып. 10. 4.2. /отв. Ред. А.М. Заяц. -Спб.: СПбГЛТУ, 2018. С. 113-118.
54. Вагизов М.Р. Разработка геоинформационной системы мониторинга лесных экосистем. // Метеорологический вестник. 2017. Т. 9. № 2. С. 39-42.
55. Вагизов М.Р. Разработка интерактивных геоинформационных систем: принципы построения и конструирования системы. // Информационные системы и технологии: теория и практика Сборник научных трудов научно-технической конференции института леса и природопользования. 2017. С. 21-27.
56. Васильев Ф. П. Численные методы решения экстремальных задач. М.: Наука, 1980.
57. Габасов Р. Ф., Кириллова Ф. М. Принцип максимума в теории оптимального управления. Минск: Наука и техника, 1974.
58. Галушкин А.И. Нейронные сети. Основы теории. 2010. - 480 с. - ISBN: 978-5-9912-0082-0
59. Гольцёв А. Д. Яркостная сегментация изображения при помощи нейроподобной сети// Автоматика. 1965. N 5. С. 40-50.
60. Горбань А. Н. Обучение нейронных сетей. М.: Изд. СССР-США СП "ПараГраф", 1990.
61. Горбань А. Н., Россиев Д. А. Нейронные сети на персональном компьютере. Новосибирск: Наука, 1996.
62. Искусственный интеллект. В 3-х кн. Справочник. Под ред. Д.А.Поспелова. М.: Радио и связь, 1990.
63. Капралов Е.Г., Кошкарев А.В., Тикунов В.С. Геоинформационпые системы // Учебное пособие в 2 томах. М.: Академия, 2004 г. 352 с.
64. Коваленко Е. Г. Региональная экономика и управление: Учеб. пособие. СПб: Питер, 2005. 225 с.
65. Коновалова Н.В., Капралов Е.Г. Введение в ГИС М.: Библион, 1997 г. 160 с.
66. Королев Ю.К. Общая геоинформатика, выпуск 1: Теоретическая геоинформатика. -СПб.: Дата+, 1998 г.
67. Кофман А. Введение в теорию нечётких множеств. М.: Радио и связь, 1982.-432 с.
68. Кошкарев А.В., Каракин В.П. Справочник по картографии М.: МАКС Пресс, 1988, 303 с.
69. Кошкарев А.В., Тикунов В.С. Геоинформатика М.: Картгеоцеитр-Геодезиздат, 1993г. 213 с.
70. Круг П.Г. Нейронные сети и нейрокомпьютеры. Учебное пособие по курсу «Микропроцессоры». – М.: Издательство МЭИ, 2002. – 176 с. ISBN 5-7046-0832-9

71. Лисицкий Д.В. Общность и различие понятий "цифровая модель местности", "цифровая карта" и "электронная карта". //51-я научно-техническая конференция 16-19 апр. 2001 г. Тез. докл. Новосибирск: СГГА, 2001. с. 143 - 144.
72. Лисицкий, Д.В. Основные принципы цифрового картографирования местности. М.: Недра, 1988. 261 с.
73. Лурье И.К. Основы геоинформатики и создание ГИС. / Дистанционное зондирование и географические информационные системы. М.: Инэкс, 2002.
74. Майкл Зейлер. Моделирование нашего мира. Руководство ESRI по проектированию базы геоданных. М.: Промт, 2002 г.
75. Майкл Н. ДеМерс. Географические информационные системы. Основы.
76. Мамагулашвили Д.И., Биденко С.И., Бородин Е.Л., Хренов М.М. Концепция геоинформационного моделирования экономической ситуации региона // Вестник ТвГУ. Серия «Экономика и управление», 2016. №3. С. 123–132.
77. Мартыненко А.А. Три периода развития военной картографии: разработка новых концепций и технологий // Геодезия и картография. 1996. -№7. - с. 44-47.
78. Медведев В. С., Потемкин В. Г. Нейронные сети: Matlab 6. — М.: Диалог-МИФИ, 2002. —489 с.
79. Мелихов А.Н., Берштейн JI.C., Коровин С.Я. Ситуационные советующие системы с нечеткой логикой. М.: Наука, 1990. - 272с.
80. Митчелл Э. Руководство по ГИС анализу — Ч. 1: Пространственные модели и взаимосвязи: Пер. с англ. — Киев, ЗАО ЕСОММ Со; Стллос, 2000 г. 198 с.
81. Мосалёв В. Системы дистанционного наблюдения за полем боя на базе разведывательно-сигнализационных приборов. // Общие Военные проблемы «Зарубежное военное обозрение». -М.: 2000, №2.
82. Мустафин Н.Г., Пирог В.П., Смирнов А.В. Методы и модели систем поддержки принятия решений. Учебное пособие. СПб.: Электротехнический Университет, 2004 г.
83. Нейронные сети. STATISTICA Neural Networks: Методология и технологии современного анализа данных / под ред. В.П. Боровикова. - 2-е изд., доп. Москва : Горячая Линия - Телеком, 2008. 392 с.
84. Нейронные сети: основные модели. И.В. Заенцев. Издатель: Воронеж Год издания: 1999 Страниц: 76
85. Ньюкомер Э. Веб-сервисы для профессионалов СПб.: Питер, 2003 г. 314 с.
86. Панкрушин, В.К. Математическое моделирование и идентификация геодинамических систем. Новосибирск: СГГА, 2002. с.424
87. Потемкин В. Г. Система MATLAB. Справочное пособие. — М: Диалог-МИФИ, 1997. —350 с.
88. Пустарнакова Ю. А. Математическое моделирование искусственных нейронных сетей// Учёные записки Тверского гос. ун-та: Сб. научн. тр. Тверь: ТвГУ, 1999. Т. 5. С. 49-53.
89. Пустарнакова Ю. А. Моделирование искусственной нейронной сети/ / Научная конференция, посвященная 70-летию со дня рождения акад. В. А. Мельникова: Сб. докладов. М., 1999. С. 205-206.

90. Пустарнакова Ю. А. Оптимизация искусственной нейронной сети/ / "Понтрягинские чтения — X". Тезисы докладов. Воронеж: ВГУ, 1999. С. 202.
91. Розенблатт Ф. Аналитические методы изучения нейронных сетей// Зарубежная радиоэлектроника. 1965. N 5. С. 40-50.
92. Розенблатт Ф. Принципы нейро динамики. М.: Мир, 1965. (Rosenblatt F. Principles of neurodynamics. Spartan, Washington, D.C., 1962.)
93. Сигеру Омату. Нейроуправление и его приложения. [Текст]: монография: 2-е изд. / Сигеру Омату, Марзуки Халид, Рубия Юсоф. М.: ИПРЖР, 2000. 272 с.
94. Соколов Е. Н., Вайткявичус Г. Г. Нейроинтеллект: от нейрона к нейрокомпьютеру. М.: Наука, 1989.
95. Суворов С. В., Матихина Н. Ю. Программное моделирование нейро-подобных структур j I Распределенная обработка информации. Улан-Уде, 1989.
96. Суровцев И.С., Клюкин В.И., Пивоварова Р.П. Нейронные сети. — Воронеж: ВГУ, 1994. — 224с.
97. Тасаки И. Нервное возбуждение. М.: Мир, 1971.
98. Тикунов В.С. Моделирование в картографии М.: МГУ, 1985г. 280 с.
99. Тикунов В.С. Моделирование в картографии М.: МГУ, 1997г. 405 с.
100. Тэнк Д. У., Хопфилд Д. Д. Коллективные вычисления в нейронопо-добных электронных схемах/. В мире науки. 1988. N 2. С. 44-53.
101. Уоссермен Ф. Нейрокомпъютерная техника: Теория и практика. М.: Мир, 1992.
102. Ф. Уоссермен. Нейрокомпьютерная техника: теория и практика. М. Мир - 1992.
103. Хайкин С. Нейронные сети: полный курс, 2-е издание. [Текст]: пер. с англ. М.: Издательский дом «Вильямс», 2006. 1104 с.
104. Хинтон Д. Е. Как обучаются нейронные сети./ В мире науки. 1992. N 11. С. 103-107.
105. Ходжкин А. JI. Нервный импульс. М.: Мир, 1965.
106. Храмов И.С. Задача оптимального управления, описывающая динамику осцилляторной нейронной сети, состоящей из n нейронов. Norwegian Journal of Development of the International Science. 2018. № 4-1 (17). С. 3-5.
107. Храмов И.С. Интеграция искусственных нейронных сетей с геоинформационными системами // Вестник ТвГУ. Серия «Математические методы управления». 2017. С. 118–120.
108. Храмов И.С. Исследование работы искусственной нейронной сети в условиях помех. В сборнике: ОБРАЗОВАНИЕ В XXI ВЕКЕ материалы Всероссийской научной заочной конференции. Министерство образования и науки РФ, Тверской государственный технический университет. 2015. С. 184-186.
109. Храмов И.С. Методика построения оптимального маршрута перехода судна в районе морской территориальной активности с применением аппарата искусственных нейронных сетей. Сборник научных трудов «Проблемы обороноспособности и безопасности». – М.: ФГБНУ «Экспертно-аналитический центр», 2018. – Вып. 20. – С. 230 – 241.

110. Храмов И.С. Модель геоданных для представления тактической обстановки в районе интенсивной морской территориальной активности. Сборник научных трудов «Проблемы обороноспособности и безопасности». – М.: ФГБНУ «Экспертно-аналитический центр», 2018. – Вып. 20. – С. 219 - 229.
111. Храмов И.С. Перспективы развития искусственного интеллекта // Актуальные направления научных исследований XXI века: теория и практика. 2015. Т. 3. № 8-1 (19-1). С. 375–377.
112. Храмов И.С. Представление и оценка экономической ситуации в регионе с использованием искусственных нейронных сетей Представление и оценка экономической ситуации в регионе с использованием искусственных нейронных сетей. // Вестник ТвГУ. – Серия "Экономика и управление". – Тверь: ТвГУ, 2018. – № 4. – С. 146 – 155.
113. Храмов И.С. Сравнение быстродействия реализации искусственной нейронной сети в различных средах программирования. В сборнике: МАТЕМАТИЧЕСКИЕ МЕТОДЫ УПРАВЛЕНИЯ Сборник научных трудов. Тверь, 2016. С. 108-112.
114. Шилин М.Б., Биденко С.И., Кравченко П.Н. и др. Концепция моделирования геоэкологической ситуации // Ученые записки РГГМУ. 2015. № 39. С. 157–164.
115. Якушев Д.И. Геоинформационные управляющие системы и технологии. СПб.: Изд-во СПб ун-та МВД России, 2014. - 248 с.
116. Якушев Д.И. Метод выделения нестационарных периодов геофизических процессов. // Записки по гидрографии. 2016. – № 277а. С. 28 – 34.
117. Якушев Д.И., Антонов А.Е. Полигармоническая модель уровня Каспийского моря и долгосрочный прогноз его изменения. // Навигация и гидрография. 2011. № 31. С. 60-64.

Приложения

Приложение А. Код программы Анаморф на языке Java

```
package ch.epfl.anamorf;
import java.util.Vector;
import com.vividsolutions.jump.workbench.JUMPWorkbench;
import com.vividsolutions.jump.workbench.model.LayerManager;
import com.vividsolutions.jump.workbench.ui.LayerViewPanel;
import com.vividsolutions.jump.workbench.ui.TreeLayerNamePanel;
public class AppContext
{

        /**
         * Setting the debug flag enables the output of console messages
         * for debugging purposes.
         */
        public static boolean DEBUG = false;
                /**
         * The short program name.
         */
        public static String shortProgramName =
                "anamorf";
        /**
         * The program name including the version number.
         */
        public static String longProgramName =
                "anamorf version 1.2.1";
        /**
         * The copyright notice.
         */
        public static String copyrightNotice =
```

```
            "Copyright 2018. Храмов.";
/**
 * The application's main window.
 */
public static MainWindow mainWindow;
/**
 * The layer manager from the JUMP project.
 */
public static LayerManager layerManager = null;

/**
 * The JUMP workbench.
 */
public static JUMPWorkbench workBench = null;
/**
 * The cartogram wizard.
 */
public static CartogramWizard cartogramWizard = null;
/**
 * The LayerViewPanel (pane displaying the maps) from the JUMP project.
 */
public static LayerViewPanel layerViewPanel = null;
/**
 * This panel is needed by the LayerViewPanel for displaying the layers
 * as a tree with the category names.
 */
public static TreeLayerNamePanel layerListPanel = null;
/**
 * The map panels displays the maps.
 */
public static MapPanel mapPanel = null;
public static SizeErrorLegend sizeErrorLegend = new SizeErrorLegend();
```

```
}
package ch.epfl.anamorf;
import java.awt.Color;
import java.awt.Font;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.util.Iterator;
import java.util.List;
import java.util.TreeMap;
import java.util.Vector;
import com.sun.swing.SwingWorker;
import com.vividsolutions.jts.geom.Coordinate;
import com.vividsolutions.jts.geom.Envelope;
import com.vividsolutions.jts.geom.GeometryFactory;
import com.vividsolutions.jts.geom.LinearRing;
import com.vividsolutions.jts.geom.LineString;
import com.vividsolutions.jts.geom.Polygon;
import com.vividsolutions.jump.feature.AttributeType;
import com.vividsolutions.jump.feature.BasicFeature;
import com.vividsolutions.jump.feature.Feature;
import com.vividsolutions.jump.feature.FeatureCollectionWrapper;
import com.vividsolutions.jump.feature.FeatureDataset;
import com.vividsolutions.jump.feature.FeatureSchema;
import com.vividsolutions.jump.workbench.model.Layer;
import com.vividsolutions.jump.workbench.model.LayerManager;
import com.vividsolutions.jump.workbench.ui.renderer.style.BasicStyle;
import com.vividsolutions.jump.workbench.ui.renderer.style.LabelStyle;
```

```
public class Cartogram extends com.sun.swing.SwingWorker
{

    /**
     * The cartogram wizard. We need the wizard reference for updating
     * the progress status informations.
     */
    CartogramWizard mCartogramWizard = null;

    /**
     * The layer manager used for cartogram computation.
     */
    LayerManager mLayerManager = null;


    /**
     * The category name for our cartogram layers.
     */
    String mCategoryName = null;


    /**
     * The name of the master layer.
     */
    String mMasterLayer = null;

    /**
     * The name of the master attribute.
     */
    String mMasterAttribute = null;

    /**
```

```
 * Is the master attribute already a density value, or must
 * the value be weighted by the polygon area (only available
 * for polygons).
 */
boolean mMasterAttributeIsDensityValue = true;


String mMissingValue = "";



/**
 * The projected master layer. We store this in order to make the
 * computation report after the projection.
 */
Layer mProjectedMasterLayer = null;



/**
 * The layers to deform simultaneously.
 */
Vector mSlaveLayers = null;

/**
 * The layers used for the constrained deformation.
 */
Vector mConstrainedDeforamtionLayers = null;

/**
 * The initial envelope for all layers.
 */
```

```
Envelope mEnvelope = new Envelope(0.0, 1.0, 0.0, 1.0);

/**
 * The size of the cartogram grid.
 */
int mGridSizeX = 100;
int mGridSizeY = 100;

/**
 * All the deformation is done on this cartogram grid.
 */
CartogramGrid mGrid = null;

/**
 * The amount of deformation is a simple stopping criterion.
 * It is an integer value between 0 (low deformation, early stopping)
 * and 100 (high deformation, late stopping).
 */
int mAmountOfDeformation = 50;


/**
 * Are the advanced options enabled or should the parameters be estimated
 * automatically by the program?
 */
boolean mAdvancedOptionsEnabled = false;

/**
 * If the advanced options are enabled, this is the grid size for the
 * diffusion algorithm.
 */
int mDiffusionGridSize = 128;
```

```
/**
 * If the advanced options are enabled, this is the number of iterations
 * the diffusion algorithm is run on the cartogram grid.
 */
int mDiffusionIterations = 3;


/**
 * The maximum running time in seconds. After this amount of time,
 * the cartogram computation is finalized. This is to avoid that the
 * computation lasts for a very very long time.
 * The default value is 3 hours.
 */
int mMaximumRunningTime = 10800;

/**
 * The maximum length of one line segment. In the projection process,
 * a straight line might be deformed to a curve. If a line segment is
 * too long, it might result in a self intersection, especially for
 * polygons. This parameter can be controlled manually or estimated
 * using the maximumSegmentLength heuristic.
 */
double mMaximumSegmentLength = 500;


/**
 * Should we create a grid layer ?
 */
boolean mCreateGridLayer = true;
```

```
/**
 * The size of the grid which can be added as a deformation grid.
 */
int mGridLayerSize = 100;

/**
 * The layer containing the deformation grid.
 */
Layer mDeformationGrid = null;


/**
 * Should we create a legend layer ?
 */
boolean mCreateLegendLayer = true;

/**
 * An array containing the legend values which should be represented
 * in the legend layer.
 */
double[] mLegendValues = null;

/**
 * The layer containing the cartogram legend.
 */
Layer mLegendLayer = null;



/**
 * The computation report.
```

```
 */
String mComputationReport = "";



/**
 * Used for storing the start time of computation. The computation
 * duration is computed based on this value which is set before starting
 * the compuation.
 */
long mComputationStartTime = 0;









/**
 * The constructor for the cartogram class.
 */
Cartogram (CartogramWizard cartogramWizard)
{

        // Storing the cartogram wizard reference.
        mCartogramWizard = cartogramWizard;

}       // Cartogram.<init>

/**
 * The construct method is an overriden method from
 * SwingWorker which does initiate the computation process.
 */
public Object construct()
```

```
{

        try
        {

                mComputationStartTime = System.nanoTime();
                // Estimating the grid size and number of loops based on
                // the amount of deformation value.
                // The amount of deformation is a value between 0 and 100.
                //  mGridSizeX and mGridSizeY are the cartogram grid size values,
                //  gastnerGridSize is the size of the diffusion grid (power of 2),
                //  gastnerLoops is the number of loops for the diffusion algorihtm.
                // The cartogram grid size varies between 100 and 1100.
                // The gastner grid size varies between 128 (2^7) and 512 (2^9).
                // The number of gastner loops varies between 1 and 4.
                                        int gastnerGridSize = 128;
                int gastnerLoops = 1;

                if (mAdvancedOptionsEnabled)
                {

                        gastnerGridSize = mDiffusionGridSize;
                        gastnerLoops = mDiffusionIterations;
                                        }
                else
                {
                        // Automatic estimation of the parameters using the
                        // amount of deformation slider.
                        mGridSizeX = (mAmountOfDeformation * 10) + 100;
                        mGridSizeY = mGridSizeX;

                        double gastnerGridPower = 8;
```

```
        if (mAmountOfDeformation < 34) gastnerGridPower = 7;
        if (mAmountOfDeformation > 66) gastnerGridPower = 9;
        double gastnerGridSizeDbl = Math.pow(2, gastnerGridPower);
        gastnerGridSize = (int)gastnerGridSizeDbl;

        double gastnerLoopsDbl = (double)mAmountOfDeformation / 25.0;
        gastnerLoopsDbl = Math.floor(gastnerLoopsDbl);
        gastnerLoops = (int)gastnerLoopsDbl + 1;
        if (gastnerLoops < 1) gastnerLoops = 1;
        if (gastnerLoops > 4) gastnerLoops = 4;

        mDiffusionGridSize = gastnerGridSize;
        mDiffusionIterations = gastnerLoops;

}
                        // User information.
mCartogramWizard.updateRunningStatus(0,
        "Подготовка...",
        "Вычисление");
                        // Compute the envelope given the initial layers.
// The envelope will be somewhat larger than just the layers.
this.updateEnvelope();

// Adjust the cartogram grid size in order to be proportional
// to the envelope.
this.adjustGridSizeToEnvelope();
if (AppContext.DEBUG)
        System.out.println("Слой: " + mGridSizeX + "x" +
                mGridSizeY);
                        mCartogramWizard.updateRunningStatus(20,
        "Подготовка...",
        "Создание картограммы");
```

```
// Create the cartogram grid.
mGrid = new CartogramGrid(mGridSizeX, mGridSizeY, mEnvelope);



if (Thread.interrupted())
{
        // Raise an InterruptedException.
        throw new InterruptedException(
                "Прервано пользователем.");
}




// Check the master attribute for invalid values.

mCartogramWizard.updateRunningStatus(50,
        "Проверка значений",
        "");

Layer masterLayer = AppContext.layerManager.getLayer(mMasterLayer);
CartogramLayer.cleanAttributeValues(masterLayer, mMasterAttribute);



// Replace the missing values with the layer mean value.
if (mMissingValue != "" && mMissingValue != null)
{
        double mean = CartogramLayer.meanValueForAttribute(
                masterLayer, mMasterAttribute);
```

```
        Double missVal = new Double(mMissingValue);

        CartogramLayer.replaceAttributeValue(masterLayer,
                mMasterAttribute, missVal.doubleValue(), mean);
}




// Compute the density values for the cartogram grid using
// the master layer and the master attribute.

mCartogramWizard.updateRunningStatus(100,
        "Вычисление плотности...",
        "");

mGrid.computeOriginalDensityValuesWithLayer
        (masterLayer,
         mMasterAttribute,
         mMasterAttributeIsDensityValue);

if (Thread.interrupted())
{
        // Raise an InterruptedException.
        throw new InterruptedException(
                "Прервано пользователем");
}
```

```
// *** PREPARE THE GRID FOR THE CONSTRAINED DEFORMATION
***

if (mConstrainedDeforamtionLayers != null)
{
        mCartogramWizard.updateRunningStatus(300,
                "Подготовка картоида...",
                "");

        mGrid.prepareGridForConstrainedDeformation(
                mConstrainedDeforamtionLayers);
}

if (Thread.interrupted())
{
        // Raise an InterruptedException.
        throw new InterruptedException(
                "Прервано пользователем.");
}



// *** RUN THE DIFFUSION ALGORITHM ***

for (int i = 0; i < gastnerLoops; i++)
{
        int adv0 = (int)(300 + ((double)i / (double)gastnerLoops) * 400);
        int adv1 = (int)(300 + ((double)(i+1) / (double)gastnerLoops) * 400);
        String text1 = "Вычислено " + (i+1) +
                " из " + gastnerLoops;
        mCartogramWizard.updateRunningStatus(adv0, text1, "");
```

```
            CartogramGastner cgast = new CartogramGastner(mGrid);
            cgast.mProgressStart = adv0;
            cgast.mProgressEnd = adv1;
            cgast.mProgressText = text1;
            cgast.mCartogramWizard = mCartogramWizard;

            cgast.compute(gastnerGridSize);

            if (i < (gastnerLoops - 1))
                  mGrid.updateDensityValues();



            if (Thread.interrupted())
            {
                  // Raise an InterruptedException.
                  throw new InterruptedException(
                        "Прервано пользователем.");
            }

}

// *** CONSTRAINED DEFORMATION ***
if (mConstrainedDeforamtionLayers != null)
{
            mCartogramWizard.updateRunningStatus(700,
                  "Постройка деформации слоев",
                  "");

            mGrid.conformToConstrainedDeformation();
}
```

```
if (Thread.interrupted())
{
        // Raise an InterruptedException.
        throw new InterruptedException(
                "прервано пользователем");
}
                        // *** PROJECTION OF ALL LAYERS ***

mCartogramWizard.updateRunningStatus(750,
        "Подготовка...",
        "");

Layer[] projLayers = this.projectLayers();




if (Thread.interrupted())
{
        // Raise an InterruptedException.
        throw new InterruptedException(
                "Прервано пользователем");
}

// *** CREATE THE DEFORMATION GRID LAYER ***
if (mCreateGridLayer)
        this.createGridLayer();
                        // *** CREATE THE LEGEND LAYER ***
if (mCreateLegendLayer)
        this.createLegendLayer();
```

```
            mCartogramWizard.updateRunningStatus(950,
                    "Подготовка отчёта...",
                    "");




            return projLayers;



}
catch (Exception e)
{

            String exceptionType = e.getClass().getName();

            if (exceptionType == "java.lang.InterruptedException")
            {
                    mCartogramWizard.setComputationError(
                            "Прервано.",
                            "",
                            "");
            }
            else
            {
                    // Retrieve the complete stack trace.
                    StringWriter sw = new StringWriter();
                    PrintWriter pw = new PrintWriter(sw, true);
                    e.printStackTrace(pw);
                    pw.flush();
```

```
                    sw.flush();

                    mCartogramWizard.setComputationError(
                        "Ошибка!",
                        e.getLocalizedMessage(),
                        sw.toString());
                }

                mCartogramWizard.goToFinishedPanel();


                if (AppContext.DEBUG)
                {
                    System.out.println("Ошибка!");
                    e.printStackTrace();
                }
                return null;
            }


}       // Cartogram.construct




/**
 * This method is called once the construct method has finished.
 * It terminates the computation, adds all layers and produces the
 * computation report.
 */
```

```
public void finished ()
{


        // *** GET THE PROJECTED LAYERS ***

        Layer[] lyr = (Layer[])this.get();



        // *** HIDE ALL LAYERS ALREADY PRESENT ***
        List layerList = mLayerManager.getLayers();
        Iterator layerIter = layerList.iterator();
        while (layerIter.hasNext())
        {
                Layer l = (Layer)layerIter.next();
                l.setVisible(false);
        }



        // *** ADD ALL THE LAYERS ***

        String catName = this.getCategoryName();

        if (mLayerManager.getCategory(catName) == null)
                mLayerManager.addCategory(catName);


        int nlyrs = lyr.length;
        for (int lyrcnt = 0; lyrcnt < nlyrs; lyrcnt++)
        {
```

```
        mLayerManager.addLayer(catName, lyr[lyrcnt]);
}

if (mDeformationGrid != null)
        mLayerManager.addLayer(catName, mDeformationGrid);

if (mLegendLayer != null)
        mLayerManager.addLayer(catName, mLegendLayer);




// *** PRODUCE THE COMPUTATION REPORT ***
this.produceComputationReport(mProjectedMasterLayer);


// *** CREATE A THEMATIC MAP USING THE SIZE ERROR ATTRIBUTE ***

// Create a color table for the size error attribute.

BasicStyle bs =
        (BasicStyle)mProjectedMasterLayer.getStyle(BasicStyle.class);
bs.setFillColor(Color.WHITE);

SizeErrorStyle errorStyle = new SizeErrorStyle();

errorStyle.setAttributeName("Ошибка размерности");

errorStyle.addColor(new BasicStyle(new Color(91, 80, 153)));
errorStyle.addColor(new BasicStyle(new Color(133, 122, 179)));
errorStyle.addColor(new BasicStyle(new Color(177, 170, 208)));
errorStyle.addColor(new BasicStyle(new Color(222, 218, 236)));
```

```
errorStyle.addColor(new BasicStyle(new Color(250, 207, 187)));
errorStyle.addColor(new BasicStyle(new Color(242, 153, 121)));
errorStyle.addColor(new BasicStyle(new Color(233, 95, 64)));

errorStyle.addLimit(new Double(70));
errorStyle.addLimit(new Double(80));
errorStyle.addLimit(new Double(90));
errorStyle.addLimit(new Double(100));
errorStyle.addLimit(new Double(110));
errorStyle.addLimit(new Double(120));



lyr[0].addStyle(errorStyle);
errorStyle.setEnabled(true);
lyr[0].getStyle(BasicStyle.class).setEnabled(false);


AppContext.sizeErrorLegend.setVisible(true);

try
{
        AppContext.layerViewPanel.getViewport().zoomToFullExtent();
} catch (Exception exc) {}


// *** SHOW THE FINISHED PANEL

mCartogramWizard.goToFinishedPanel();
```

```
}

/**
 * Sets the layer manager.
 */
public void setLayerManager (LayerManager lm)
{
        mLayerManager = lm;
}

/**
 * Sets the name of the cartogram master layer.
 */
public void setMasterLayer (String layerName)
{
        mMasterLayer = layerName;
}

/**
 * Sets the name of the cartogram master attribute.
 */
public void setMasterAttribute (String attributeName)
{
        mMasterAttribute = attributeName;
}
```

```
/**
 * Lets define us whether the master attribute is a density value
 * or a population value.
 */
public void setMasterAttributeIsDensityValue (boolean isDensityValue)
{
        mMasterAttributeIsDensityValue = isDensityValue;
}



/**
 * Defines the layers to deform during the
 * cartogram process.
 */
public void setSlaveLayers (Vector slaveLayers)
{
        mSlaveLayers = slaveLayers;
}



/**
 * Defines the layers which should not be deformed.
 */
public void setConstrainedDeformationLayers (Vector layers)
{
        mConstrainedDeforamtionLayers = layers;
```

```
}



/**
 * Defines the grid size in x and y dimensions.
 */
public void setGridSize (int x, int y)
{
        mGridSizeX = x;
        mGridSizeY = y;
}



/**
 * Defines the amount of deformation. This is an integer value between
 * 0 and 100. The default value is 50.
 */
public void setAmountOfDeformation (int deformation)
{
        mAmountOfDeformation = deformation;
}



/**
 * Defines the maximum running time in seconds. The default value is
 * 259200 seconds (3 days).
 */
public void setMaximumRunningTime (int seconds)
{
        mMaximumRunningTime = seconds;
}
```

```
/**
 * Computes the cartogram envelope using the provided layers.
 * The envelope will be larger than the layers in order to allow
 * the cartogram deformation inside this envelope.
 */
private void updateEnvelope ()
{

        // Setting the initial envelope using the master layer.
        Layer lyr = mLayerManager.getLayer(mMasterLayer);
        Envelope masterEnvelope =
                lyr.getFeatureCollectionWrapper().getEnvelope();

        mEnvelope = new Envelope(
                masterEnvelope.getMinX(),
                masterEnvelope.getMaxX(),
                masterEnvelope.getMinY(),
                masterEnvelope.getMaxY());



        // Expanding the initial envelope using the slave and
        // constrained deformation layers.
        if (mSlaveLayers != null)
        {
                Iterator lyrIterator = mSlaveLayers.iterator();
                while (lyrIterator.hasNext())
                {
                        lyr = (Layer)lyrIterator.next();
```

```
                    mEnvelope.expandToInclude(
                            lyr.getFeatureCollectionWrapper().getEnvelope());
                }
        }

        if (mConstrainedDeforamtionLayers != null)
        {
                Iterator lyrIterator = mConstrainedDeforamtionLayers.iterator();
                while (lyrIterator.hasNext())
                {
                        lyr = (Layer)lyrIterator.next();
                        mEnvelope.expandToInclude(
                                lyr.getFeatureCollectionWrapper().getEnvelope());
                }
        }


        // Enlarge the envelope by 5%.
        mEnvelope.expandBy(mEnvelope.getWidth() * 0.05,
                mEnvelope.getHeight() * 0.05);


}       // Cartogram.updateEnvelope




/**
 * Adjusts the grid size in order to be proportional to the
 * envelope. It will not increase the grid size, but it will
 * decrease the grid size on the shorter side.
 */
```

```
private void adjustGridSizeToEnvelope ()
{

	if (mEnvelope == null)
		return;



	double width = mEnvelope.getWidth();
	double height = mEnvelope.getHeight();

	if (width < height)
	{
		// Adjust the x grid size.
		mGridSizeX = (int)Math.round(mGridSizeY * (width / height));
	}
	else if (width > height)
	{
		// Adjust the y grid size.
		mGridSizeY = (int)Math.round(mGridSizeX * (height / width));
	}

}	// Cartogram.adjustGridSizeToEnvelope




/**
 * Projects all layers. Creates a new layer for each projected layer.
 */
private Layer[] projectLayers ()
throws InterruptedException
```

```
{

        // Get the number of layers to project
        // (one master layer and all slave layers).
        int nlyrs = 1;
        if (mSlaveLayers != null)
                nlyrs += mSlaveLayers.size();


        // We store the projected layers in an array.
        Layer[] layers = new Layer[nlyrs];


        // Compute the maximum segment length for the layers.
        mMaximumSegmentLength = this.estimateMaximumSegmentLength();


        // Project the master layer.

        mCartogramWizard.updateRunningStatus(750,
                "Подготовка слоев...",
                "Слой 1 из "+ nlyrs);


        Layer masterLayer = mLayerManager.getLayer(mMasterLayer);
        CartogramLayer.regularizeLayer(masterLayer, mMaximumSegmentLength);
        mProjectedMasterLayer =
                CartogramLayer.projectLayerWithGrid(masterLayer, mGrid);

        layers[0] = mProjectedMasterLayer;
```

```
        if (Thread.interrupted())
        {
                // Raise an InterruptedException.
                throw new InterruptedException(
                        "Прервано пользователем.");
        }


        // Project the slave layers.
        for (int lyrcnt = 0; lyrcnt < (nlyrs - 1); lyrcnt++)
        {
                mCartogramWizard.updateRunningStatus(800 + ((lyrcnt+1)/(nlyrs-1)*150),
                "Подготовка слоев...",
                "Слой "+ (lyrcnt+2) +" из "+ nlyrs);

                Layer slaveLayer = (Layer)mSlaveLayers.get(lyrcnt);
                CartogramLayer.regularizeLayer(slaveLayer, mMaximumSegmentLength);
                layers[lyrcnt+1] =
                        CartogramLayer.projectLayerWithGrid(slaveLayer, mGrid);
        }


        return layers;

}       // Cartogram.projectLayers
```

```
/**
 * Says whether we should create a grid layer or not.
 */
public boolean getCreateGridLayer ()
{
	return mCreateGridLayer;
}



/**
 * Sets the flag for creating or not a grid layer.
 */
public void setCreateGridLayer (boolean createGridLayer)
{
	mCreateGridLayer = createGridLayer;
}



/**
 * Returns the grid layer size. This is the grid which is produced
 * for visual effect only.
 */
public int getGridLayerSize ()
{
	return mGridLayerSize;
}



/**
```

```
 * Changes the size of the grid layer to produce.
 */
public void setGridLayerSize (int gridLayerSize)
{
	mGridLayerSize = gridLayerSize;
}



/**
 * Says whether we should create a legend layer or not.
 */
public boolean getCreateLegendLayer ()
{
	return mCreateLegendLayer;
}


/**
 * Sets the flag which says whether to create a legend layer or not.
 */
public void setCreateLegendLayer (boolean createLegendLayer)
{
	mCreateLegendLayer = createLegendLayer;
}



public double[] getLegendValues ()
{
	return mLegendValues;
}
```

```
public void setLegendValues (double[] legendValues)
{
	mLegendValues = legendValues;
}



public boolean getAdvancedOptionsEnabled ()
{
	return mAdvancedOptionsEnabled;
}

public void setAdvancedOptionsEnabled (boolean enabled)
{
	mAdvancedOptionsEnabled = enabled;
}



public int getDiffusionGridSize ()
{
	return mDiffusionGridSize;
}

public void setDiffusionGridSize (int size)
{
	mDiffusionGridSize = size;
```

```
}


public int getDiffusionIterations ()
{
	return mDiffusionIterations;
}


public void setDiffusionIterations (int iterations)
{
	mDiffusionIterations = iterations;
}



/**
 * Returns the category name for our cartogram layers.
 */
public String getCategoryName ()
{

	if (mCategoryName == null)
	{

		// Create a new category in the layer manager in order to
		// properly separate the cartogram layers. We call the new category
		// «Cartogram x», where x is a serial number.

		int catNumber = 1;
		String categoryName = "Картограмма " + catNumber;
		while (mLayerManager.getCategory(categoryName) != null)
```

```
                {
                        catNumber++;
                        categoryName = "Картограмма " + catNumber;
                }

                mCategoryName = categoryName;

        }

        return mCategoryName;

}



/**
 * Creates a layer with the deformation grid.
 */
private void createGridLayer ()
{

        Envelope env = mEnvelope;

        // Compute the deformation grid size in x and y direction.

        double resolution =
                Math.max((env.getWidth() / (double)(mGridLayerSize + 1)),
                                (env.getHeight() / (double)(mGridLayerSize + 1)));

        int sizeX =
                (int)Math.round(Math.floor(env.getWidth() / resolution)) - 1;
```

```
int sizeY =
        (int)Math.round(Math.floor(env.getHeight() / resolution)) - 1;



// CREATE THE NEW LAYER

// Create a new Feature Schema for the new layer.
FeatureSchema fs = new FeatureSchema();
fs.addAttribute("GEOMETRY", AttributeType.GEOMETRY);
fs.addAttribute("ID", AttributeType.INTEGER);

// Create a new empty Feature Dataset.
FeatureDataset fd = new FeatureDataset(fs);

// Create a Geometry Factory for creating the points.
GeometryFactory gf = new GeometryFactory();



// CREATE ALL FEATURES AND LINES
int j, k;
int i = 0;

// Horizontal lines
for (k = 0; k < sizeY; k++)
{
        // Create an empty Feature.
        BasicFeature feat = new BasicFeature(fs);

        // Create the line string and add it to the Feature.
        Coordinate[] coords = new Coordinate[sizeX];
```

```
        for (j = 0; j < sizeX; j++)
        {
                double x = env.getMinX() + (j * resolution);
                double y = env.getMinY() + (k * resolution);
                coords[j] = mGrid.projectPointAsCoordinate(x, y);
        }

        LineString ls = null;
        if (coords != null)
                ls = gf.createLineString(coords);

        if (ls != null)
        {
                feat.setGeometry(ls);

                // Add the other attributes.
                Integer idobj = new Integer((int)i);
                feat.setAttribute("ID", idobj);
                i++;

                // Add Feature to the Feature Dataset.
                fd.add(feat);
        }

}


// Vertical lines
for (j = 0; j < sizeX; j++)
{
        // Create an empty Feature.
        BasicFeature feat = new BasicFeature(fs);
```

```
            // Create the line string and add it to the Feature.
            Coordinate[] coords = new Coordinate[sizeY];
            for (k = 0; k < sizeY; k++)
            {
                  double x = env.getMinX() + (j * resolution);
                  double y = env.getMinY() + (k * resolution);
                  coords[k] = mGrid.projectPointAsCoordinate(x, y);
            }

            LineString ls = null;
            if (coords != null)
                  ls = gf.createLineString(coords);

            if (ls != null)
            {
                  feat.setGeometry(ls);

                  // Add the other attributes.
                  Integer idobj = new Integer((int)i);
                  feat.setAttribute("ID", idobj);
                  i++;

                  // Add Feature to the Feature Dataset.
                  fd.add(feat);
            }

}
```

```
        // Create the layer.
        mDeformationGrid =
                new Layer("Таблица деформации", Color.GRAY, fd, mLayerManager);


}       // Cartogram.createGridLayer




/**
 * Creates an optional legend layer.
 */
private void createLegendLayer()
{

        // The master layer.
        Layer masterLayer = mLayerManager.getLayer(mMasterLayer);



        double distanceBetweenSymbols =
                (masterLayer.getFeatureCollectionWrapper().getEnvelope().
                        getWidth() / 10);


        // Estimate legend values if there are none.

        double attrMax = CartogramLayer.maxValueForAttribute(
                        masterLayer, mMasterAttribute);
```

```
if (mLegendValues == null)
{

        double attrMin = CartogramLayer.minValueForAttribute(
                masterLayer, mMasterAttribute);

        double attrMean = CartogramLayer.meanValueForAttribute(
                masterLayer, mMasterAttribute);



        int nvalues = 3;

        double maxLog = Math.floor(Math.log10(attrMax));
        double maxValue = Math.pow(10, maxLog);
        double secondValue = Math.pow(10, (maxLog-1));

        mLegendValues = new double[nvalues];
        mLegendValues[0] = secondValue;
        mLegendValues[1] = maxValue;
        mLegendValues[2] = attrMax;
}




// CREATE THE NEW LAYER

// Create a new Feature Schema for the new layer.
FeatureSchema fs = new FeatureSchema();
fs.addAttribute("GEOMETRY", AttributeType.GEOMETRY);
fs.addAttribute("ID", AttributeType.INTEGER);
```

```
fs.addAttribute("VALUE", AttributeType.DOUBLE);
fs.addAttribute("AREA", AttributeType.DOUBLE);
fs.addAttribute("COMMENT", AttributeType.STRING);

// Create a new empty Feature Dataset.
FeatureDataset fd = new FeatureDataset(fs);

// Create a Geometry Factory for creating the points.
GeometryFactory gf = new GeometryFactory();




// CREATE THE FEATURES FOR THE LEGEND LAYER.

int nvals = mLegendValues.length;

double totalArea = CartogramLayer.totalArea(masterLayer);
double valuesSum = CartogramLayer.sumForAttribute(
        masterLayer, mMasterAttribute);

double x = mEnvelope.getMinX();
double y = mEnvelope.getMinY();

int id = 1;

int valcnt;
for (valcnt = 0; valcnt < nvals; valcnt++)
{
        double valsize = totalArea / valuesSum * mLegendValues[valcnt];
        double rectsize = Math.sqrt(valsize);
```

```
// Create the coordinate points.
Coordinate[] coords = new Coordinate[5];
coords[0] = new Coordinate(x, y);
coords[1] = new Coordinate((x+rectsize), y);
coords[2] = new Coordinate((x+rectsize), (y-rectsize));
coords[3] = new Coordinate(x, (y-rectsize));
coords[4] = new Coordinate(x, y);

// Create geometry.
LinearRing lr = gf.createLinearRing(coords);
Polygon poly = gf.createPolygon(lr, null);

// Create the Feature.
BasicFeature feat = new BasicFeature(fs);
feat.setAttribute("GEOMETRY", poly);
feat.setAttribute("ID", id);
feat.setAttribute("VALUE", mLegendValues[valcnt]);
feat.setAttribute("AREA", valsize);

if (valcnt == 0)
        feat.setAttribute("COMMENT", "Главное значение");
else if (valcnt == 1)
        feat.setAttribute("COMMENT",
                "Максимум (" + attrMax + ")");

// Add the Feature to the Dataset.
fd.add(feat);


// Change the coordinates.
x += rectsize + distanceBetweenSymbols;
```

```
            id++;

        }


        // Create the layer.
        mLayerManager.setFiringEvents(false);
        mLegendLayer =
            new Layer("Legend", Color.GREEN, fd, mLayerManager);
        LabelStyle legendLabels = mLegendLayer.getLabelStyle();
        legendLabels.setAttribute("VALUE");
        legendLabels.setEnabled(true);
        legendLabels.setFont(new Font(null, Font.PLAIN, 10));
        mLayerManager.setFiringEvents(true);

}       // Cartogram.createLegendLayer




/**
 * Creates the computation report and stores it in the object attribute.
 */
public void produceComputationReport (Layer projectedMasterLayer)
{

        StringBuffer rep = new StringBuffer();

        rep.append("Отчёт\n\n");

        rep.append("Параметры:\n");
```

```
rep.append("Слои: " + mMasterLayer + "\n");
rep.append("Атрибуты: " + mMasterAttribute + "\n");

String attrType = "Плотность";
if (mMasterAttributeIsDensityValue) attrType = "Density value";
rep.append("Тип: " + attrType + "\n");

String transformationQuality = "";
if (mAdvancedOptionsEnabled)
        transformationQuality = "disabled";
else
        transformationQuality = "" + mAmountOfDeformation + " из 100";
rep.append("Качество: " + transformationQuality + "\n");

rep.append("Размер картограммы: "+ mGridSizeX +" x "+ mGridSizeY +"\n");
rep.append("Искажений: "+ mDiffusionGridSize +"\n");
rep.append("Итераций: "+ mDiffusionIterations +"\n\n");



rep.append("CARTOGRAM LAYER & ATTRIBUTE STATISTICS:\n");
Layer masterLayer = mLayerManager.getLayer(mMasterLayer);
int nfeat = masterLayer.getFeatureCollectionWrapper().getFeatures().size();
rep.append("Число: "+ nfeat +"\n");

double mean = CartogramLayer.meanValueForAttribute(
        masterLayer, mMasterAttribute);
rep.append("Атрибуты: " + mean + "\n");

double min = CartogramLayer.minValueForAttribute(
        masterLayer, mMasterAttribute);
rep.append("Минимум: " + min + "\n");
```

```
double max = CartogramLayer.maxValueForAttribute(
        masterLayer, mMasterAttribute);
rep.append("Максимум: " + max + "\n\n");



rep.append("Слоев:\n");
Vector simLayers = mCartogramWizard.getSimultaneousLayers();
if (simLayers == null || simLayers.size() == 0)
{
        rep.append("Нет\n\n");
}
else
{
        Iterator simLayerIter = simLayers.iterator();
        while (simLayerIter.hasNext())
        {
                Layer lyr = (Layer)simLayerIter.next();
                rep.append(lyr.getName() +"\n");
        }
        rep.append("\n");
}


rep.append("Построено слоев:\n");
Vector constLayers = mCartogramWizard.getConstrainedDeformationLayers();
if (constLayers == null || constLayers.size() == 0)
{
        rep.append("Нет\n\n");
}
else
```

```
        {
                Iterator constLayerIter = constLayers.iterator();
                while (constLayerIter.hasNext())
                {
                        Layer lyr = (Layer)constLayerIter.next();
                        rep.append(lyr.getName() +"\n");
                }
                rep.append("\n");
        }



        // Compute the cartogram error.
        double meanError = CartogramLayer.computeCartogramSizeError(
                projectedMasterLayer, mMasterAttribute, masterLayer, "Ошибка
размерности");

        rep.append("Ошибки\n");
        rep.append("Ошибки снижают качество результата.\n");
        rep.append("Ошибок: "+ meanError +"\n");

        double stdDev = CartogramLayer.standardDeviationForAttribute(
                projectedMasterLayer, "Ошибка размерности");
        rep.append("Standard deviation: "+ stdDev +"\n");

        double pctl25 = CartogramLayer.percentileForAttribute(
                projectedMasterLayer, "Ошибка размерности", 25);
        rep.append("25th percentile: "+ pctl25 +"\n");

        double pctl50 = CartogramLayer.percentileForAttribute(
                projectedMasterLayer, "Ошибка размерности", 50);
        rep.append("50th percentile: "+ pctl50 +"\n");
```

```
        double pctl75 = CartogramLayer.percentileForAttribute(
                projectedMasterLayer, "Ошибка размерности", 75);
        rep.append("75th percentile: "+ pctl75 +"\n");




        // Compute the number of features between the 25th and 75th
        // percentile and the percentage.

        FeatureCollectionWrapper fcw =
                projectedMasterLayer.getFeatureCollectionWrapper();

        Iterator featIter = fcw.iterator();
        int nFeaturesInStdDev = 0;
        int nFeatures = fcw.size();
        while (featIter.hasNext())
        {
                Feature feat = (Feature)featIter.next();

                double value =
                        CartogramFeature.getAttributeAsDouble(feat, "Ошибка
размерности");

                if (value >= (meanError - stdDev) && value <= (meanError + stdDev))
                        nFeaturesInStdDev++;
        }


        double percFeaturesInStdDev =
                (double)nFeaturesInStdDev / (double)nFeatures * (double)100;
```

```
        int pfint = (int)Math.round(percFeaturesInStdDev);

        rep.append("Без ошибок: "+
                nFeaturesInStdDev +" of "+ nFeatures +" ("+
                pfint +"%)\n\n");




        long estimatedTime = System.nanoTime() - mComputationStartTime;
        estimatedTime /= 1000000000;
        rep.append("Время: "+ estimatedTime +" секунд\n");


        mComputationReport = rep.toString();

}

public String getComputationReport ()
{
        return mComputationReport;
}
public double estimateMaximumSegmentLength ()
{

        // Check the input variables. Otherwise, return a default value.
        double defaultValue = 500.0;
        if (mEnvelope == null) return defaultValue;
        if (mMasterLayer == null) return defaultValue;

        // Compute the edge length of the square having the same area as
```

```
        // the cartogram envelope.

        double envArea = mEnvelope.getWidth() * mEnvelope.getHeight();
        if (envArea <= 0.0) return defaultValue;

        double edgeLength = Math.sqrt(envArea);



        // Get the number of features and the features per edge.

        Layer layer = mLayerManager.getLayer(mMasterLayer);
        if (layer == null) return defaultValue;

        int nfeat = layer.getFeatureCollectionWrapper().getFeatures().size();
        double featuresPerEdge = Math.sqrt((double)nfeat);



        // Compute the length per feature.
        // 1/10 of the length per feature is our estimate for the
        // maximum segment length.

        double lengthPerFeature = edgeLength / featuresPerEdge;

        return (lengthPerFeature / 10);

    }   // estimateMaximumSegmentLength
```

```
	public void setMissingValue (String value)
	{
		mMissingValue = value;
	}



}	// Cartogram



package ch.epfl.anamorf;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Stroke;
import java.awt.geom.NoninvertibleTransformException;
import java.util.Vector;
import com.vividsolutions.jump.feature.Feature;
import com.vividsolutions.jump.workbench.model.Layer;
import com.vividsolutions.jump.workbench.ui.Viewport;
import com.vividsolutions.jump.workbench.ui.renderer.style.BasicStyle;
import com.vividsolutions.jump.workbench.ui.renderer.style.Style;
import com.vividsolutions.jump.workbench.ui.renderer.style.StyleUtil;
/**
 * The SizeErrorStyle produces the thematic map based on the SizeError
 * attribute.
 */
public class SizeErrorStyle implements Style
{
	boolean _enabled = false;
```

```
String _attrName;
Vector _limits;
Vector _colors;
BasicStyle _defaultStyle;
private Stroke _fillStroke = new BasicStroke(1);
public SizeErrorStyle ()
{
        _limits = new Vector();
        _colors = new Vector();
        _defaultStyle = new BasicStyle(Color.ORANGE);
}
public Object clone()
{
        return null;
}
public void initialize (Layer layer)
{
}
public boolean isEnabled ()
{
        return _enabled;
}
public void paint (Feature f, Graphics2D g, Viewport viewport)
        throws NoninvertibleTransformException
{
                BasicStyle s = this.getStyleForFeature(f);

StyleUtil.paint(
                f.getGeometry(),
                g,
                viewport,
                s.isRenderingFill(),
```

```
_fillStroke,
s.getFillColor(),
s.isRenderingLine(),
                s.getLineStroke(),
                s.getLineColor());
        }
public void setEnabled (boolean enabled)
{
        _enabled = enabled;
}
public void setDefaultStyle (BasicStyle defaultStyle)
{
        _defaultStyle = defaultStyle;
}
public int getNumberOfColors ()
{
        return _colors.size();
}
public BasicStyle getColorAtIndex (int index)
{
        return (BasicStyle)_colors.get(index);
}
public void addColor (BasicStyle color)
{
        _colors.add(color);
}
        public void setColorAtIndex (BasicStyle color, int index)
{
        _colors.set(index, color);
}
        public int getNumberOfLimits ()
{
```

```
        return _limits.size();
}
public Double getLimitAtIndex (int index)
{
        return (Double)_limits.get(index);
}
public void addLimit (Double limit)
{
        _limits.add(limit);
}
public void setLimitAtIndex (Double limit, int index)
{
        _limits.set(index, limit);
}
        public void setAttributeName (String attrName)
{
        _attrName = attrName;
}
        private BasicStyle getStyleForFeature (Feature f)
{
                // Get the attribute value.
        Double value = (Double)f.getAttribute(_attrName);
                        boolean valueFound = false;
        int limitIndex = 0;
        BasicStyle s = null;
        while (valueFound == false && limitIndex < _limits.size())
        {
                Double limit = (Double)_limits.get(limitIndex);
                if (value.doubleValue() <= limit.doubleValue())
                        valueFound = true;
                        limitIndex++;
        }
```

```
                return (BasicStyle)_colors.get(limitIndex);
                }
}
```

Приложение В. Код программы Маршрутоид на языке C

```
#ifndef NEURAL_ACTIVATION_H
#define NEURAL_ACTIVATION_H

namespace neural {
  namespace activation {
    const std::function<double (double)> sigmoid_func =
      [](double in ) -> double {
      if (in < -45.0) {
            return 0.0;
      } else if (in > 45.0) {
            return 1.0;
      } else {
            return 1.0 / (1 + exp(-in));
      }
    };

    const std::function<double (double)> sigmoid_deriv =
      [](double in) -> double {
      return in * (1 - in);
    };

    const std::function<double (double)> tanh_func =
      [](double in) -> double {
      if (in < -10.0) {
            return -1.0;
      } else if (in > 10.0) {
            return 1.0;
```

```
    } else {
        return tanh(in);
    }
  };

  const std::function<double (double)> tanh_deriv =
    [](double in) -> double {
    return (1 + in) * (1-in);
  };

  enum Functions {
    SIGMOID,
    TANH
  };
 }
}
#endif
#include "neural/Layer.h"
#include <cassert>
namespace neural {
 Layer::Layer(int neuron_count, shared_ptr<Layer> previous) :
  prev(previous),
  next()
 {
  if (prev) {
    input_count = prev->size();
  }
  init_neurons(neuron_count, input_count);
 }
 Layer::Layer(vector<vector<double> > neuron_data, shared_ptr<Layer> previous) :
  prev(previous),
  next()
```

```
{
  if (prev) {
    input_count = prev->size();
  }
  init_neurons(neuron_data);
}

Layer::Layer(int neuron_count, int inputs) :
  prev(NULL),
  next(NULL),
  input_count(inputs)
{
  init_neurons(neuron_count, input_count);
}
Layer::Layer(vector<vector<double> > neuron_data, int inputs) :
  prev(NULL),
  next(NULL),
  input_count(inputs)
{
  init_neurons(neuron_data);
}
Layer::Layer(vector<Neuron> neuron_vector, shared_ptr<Layer> previous) :
  prev(previous),
  next(NULL),
  input_count(previous->size()),
  neurons(neuron_vector)
{}
Layer::Layer(vector<Neuron> neuron_vector, int inputs) :
  prev(NULL),
  next(NULL),
  input_count(inputs),
  neurons(neuron_vector)
```

```
{}

Layer* Layer::read(istream &s, shared_ptr<Layer> previous) {
  Layer* fail = new Layer(0,0);
  if(!s.good()) return fail;
  std::string keyword;
  s >> keyword;
  if(keyword != "LAYER") return fail;

  s >> keyword;
  if(keyword != "inputs") return fail;
  int inputs;
  s >> inputs;
  if(inputs != previous->size()) return fail;

  s >> keyword;
  if(keyword != "neurons") return fail;
  int neuron_count;
  s >> neuron_count;
  // consume newline
  char c = s.get();
  if (c != '\n') return fail;

  vector<Neuron> neuron_vector = readNeurons(s, neuron_count);
  if (neuron_vector.size() == 0) return fail;
  delete fail;
  return new Layer(neuron_vector, previous);
}

Layer* Layer::read(istream &s, int input_size) {
  Layer* fail = new Layer(0,0);
  if(!s.good()) return fail;
```

```
std::string keyword;
s >> keyword;
if(keyword != "LAYER") {
  cerr << "Missing LAYER keyword";
  return fail;
}

s >> keyword;
if(keyword != "inputs") {
  cerr << "Missing inputs" << endl;
  return fail;
}
int inputs;
s >> inputs;
if(inputs != input_size) {
  cerr << "Wrong input size!" << endl;
  return fail;
}

s >> keyword;
if(keyword != "neurons") {
  cerr << "Missing neuron count" << endl;
  return fail;
}
int neuron_count;
s >> neuron_count;
// consume newline
char c = s.get();
if (c != '\n') {
  cerr << "Missing newline after neuron count" << endl;
  return fail;
}
```

```
  vector<Neuron> neuron_vector = readNeurons(s, neuron_count);
  if (neuron_vector.size() == 0) {
    cerr << "Error reading neurons" << endl;
    return fail;
  }
  delete fail;
  return new Layer(neuron_vector, input_size);
}

void Layer::setNextLayer(shared_ptr<Layer> n) {
    next = n;
}

void Layer::updateOutputs(vector<double> inputs) {
  output.clear();
  for (vector<Neuron>::iterator it = neurons.begin(); it != neurons.end(); it++) {
    it->updateOutput(inputs);
    output.push_back(it->Output());
  }
  if (next) {
    next->updateOutputs(this->output);
  }
}

void Layer::updateDeltas(vector<double> summed_weighed_deltas) {
  assert(summed_weighed_deltas.size() == this->size());
  vector<double> newDeltas(input_count);
  for (int i = 0; i < size(); i++) {
    neurons[i].updateDelta(summed_weighed_deltas[i]);
    for (int j = 0; j < newDeltas.size(); j++) {
        newDeltas[j] += neurons[i].Delta(j);
```

```
    }
  }
  if (prev) {
    prev->updateDeltas(newDeltas);
  }
}

void Layer::updateWeights(vector<double> inputs, double learning_rate) {
  for (vector<Neuron>::iterator it = neurons.begin(); it != neurons.end(); it++) {
    it->updateWeights(inputs, learning_rate);
  }
  if(next) {

    next->updateWeights(this->output, learning_rate);
  }
}

bool Layer::write(ostream &s) const {
  if (!s.good()) return false;
  s << "LAYER\n"
    << "inputs " << input_count << "\n"
    << "neurons "     << size() << "\n";
  bool success = true;
  for (vector<Neuron>::const_iterator it = neurons.begin(); it != neurons.end(); it++) {
    success &= it->write(s);
  }
  return success;
}

void Layer::init_neurons(int neuron_count, int inputs) {
  for (int i = 0; i < neuron_count; i++) {
```

```
    neurons.emplace_back(inputs);
   }
  }
  void Layer::init_neurons(vector<vector<double> > neuron_data) {
   for (vector<vector<double> >::iterator it = neuron_data.begin(); it != neuron_data.end(); it++) {
    assert(it->size() == input_count);
    neurons.emplace_back(*it);
   }
  }

  vector<Neuron> Layer::readNeurons(istream &s, int count) {
   vector<Neuron> neuron_vector;
   Neuron current(0);
   for(int i = 0; i < count; i++) {
    current = Neuron::read(s);

    if (current.InputSize() == 0 || !s.good()) {
        return vector<Neuron>();
    }
    neuron_vector.push_back(current);
   }
   return neuron_vector;
  }
}
#ifndef NEURAL_LAYER_H
#define NEURAL_LAYER_H

#include <memory>
#include <vector>
#include <fstream>
#include <iostream>
#include "Neuron.h"
```

```
using namespace std;

namespace neural {


  class Layer {
  public:

    Layer(int neuron_count, shared_ptr<Layer> previous);


    Layer(vector<vector<double> > neuron_data, shared_ptr<Layer> previous);

    Layer(vector<Neuron> neuron_vector, shared_ptr<Layer> previous);


    Layer(int neuron_count, int inputs);


    Layer(vector<vector<double> > neuron_data, int inputs);


    Layer(vector<Neuron> neuron_vector, int inputs);


    static Layer* read(istream &s, shared_ptr<Layer> previous);


    static Layer* read(istream &s, int input_size);

    void setNextLayer(shared_ptr<Layer> n);
```

```
  void updateOutputs(vector<double> inputs);

  void updateDeltas(vector<double> summed_weighed_deltas);

  void updateWeights(vector<double> inputs, double learning_rate);

  inline int size() const { return neurons.size(); };

  bool write(ostream &s) const;

  inline vector<double> Output() { return output; };

  inline shared_ptr<Layer> nextLayer() const { return next; };
private:
  vector<double> output;
  void init_neurons(int neuron_count, int inputs);

  void init_neurons(vector<vector<double> > neuron_data);
  static vector<Neuron> readNeurons(istream &s, int count);
  shared_ptr<Layer> prev;
  shared_ptr<Layer> next;
  int input_count;
  vector<Neuron> neurons;
```

```
  };
}
#endif
#include "neural/Network.h"
#include <cassert>
namespace neural {
  Network::Network(int input, int output, vector<int> hidden) :
    layerCount(hidden.size() + 1),
    inputLayer(input),
    firstHidden(),
    outputLayer()
  {
    if (hidden.size() > 0) {

      firstHidden = shared_ptr<Layer>(new Layer(hidden.front(), input));
      shared_ptr<Layer> lastHidden(firstHidden);
      for (int i = 1; i < hidden.size(); i++) {
           shared_ptr<Layer> tmp(new Layer(hidden[i], lastHidden));
           lastHidden->setNextLayer(tmp);
           lastHidden = tmp;
      }
      outputLayer = shared_ptr<Layer>(new Layer(output, lastHidden));
      lastHidden->setNextLayer(outputLayer);
    } else {

      outputLayer = shared_ptr<Layer>(new Layer(output, input));
    }
  }

  Network::Network(int input, shared_ptr<Layer> hidden, shared_ptr<Layer> output) :
    layerCount(1),
    inputLayer(input),
```

```
  firstHidden(hidden),
  outputLayer(output)
{
  if (firstHidden) {
    shared_ptr<Layer> currentLayer = firstHidden;
    while(currentLayer->nextLayer()) {
        layerCount++;
        currentLayer = currentLayer->nextLayer();
    }
    assert(currentLayer == outputLayer);
  }
}

Network Network::read(string &filename) {
  ifstream file(filename.c_str(), ios_base::in | ios_base::binary);
  if (!file.is_open()) {
    return Network(0,0);
  }
  Network result = read(file);
  file.close();
  return result;
}

Network Network::read(istream &s) {
  Network fail = Network(0,0);
  if(!s.good()) {
    return fail;
  }
  string keyword;
  s >> keyword;
  if(keyword != "NETWORK") {
    {
```

```
      return fail;
  }
}

s >> keyword;
if(keyword != "input_size") {
  {
      return fail;
  }
}
int input_size;
s >> input_size;

s >> keyword;
if(keyword != "layers") {
  return fail;
}
int layers;
s >> layers;

char c = s.get();
if(c != '\n') {
  return fail;
}

shared_ptr<Layer> first(Layer::read(s, input_size));
if (first->size() == 0) {
  return fail;
}
layers--;
if (layers == 0) {
  return Network(input_size, shared_ptr<Layer>(NULL), first);
```

```
  }
  shared_ptr<Layer> current = shared_ptr<Layer>(Layer::read(s, first));
  if (current->size() == 0) {
    return fail;
  }
  first->setNextLayer(current);
  layers--;
  while(layers > 0) {
    current->setNextLayer(shared_ptr<Layer>(Layer::read(s, current)));
    current = current->nextLayer();
    if (current->size() == 0) {
        return fail;
    }
    layers--;
  }
  return Network(input_size, first, current);
}

double Network::trainSingle(vector<double> input, vector<double> expected_output, double
learning_rate) {
  assert(input.size() == inputLayer.size());
  assert(expected_output.size() == outputLayer->size());
  inputLayer = input;
  shared_ptr<Layer> startLayer;
  if (firstHidden) {
    startLayer = firstHidden;
  } else {
    startLayer = outputLayer;
  }

  startLayer->updateOutputs(inputLayer);
  vector<double> deltas;
```

```
  for( int i = 0; i < outputLayer->size(); i++) {
    deltas.push_back(expected_output[i] - outputLayer->Output()[i]);
  }
  outputLayer->updateDeltas(deltas);
  startLayer->updateWeights(inputLayer, learning_rate);



  startLayer->updateOutputs(inputLayer);



  double mse = 0.0;
  for( int i = 0; i < outputLayer->size(); i++) {
    mse += (expected_output[i] - outputLayer->Output()[i]) * (expected_output[i] - outputLayer-
>Output()[i]);
  }
  return mse / (double) outputLayer->size();
 }

 vector<double> Network::run(vector<double> input) {
  shared_ptr<Layer> startLayer;
  if (firstHidden) {
    startLayer = firstHidden;
  } else {
    startLayer = outputLayer;
  }
  startLayer->updateOutputs(input);
  return outputLayer->Output();
 }

 bool Network::write(string &filename) const {

  ofstream file(filename.c_str(), ios_base::out | ios_base::binary);
```

```
  if ( !file.is_open() ) {
    return false;
  }
  bool success = write(file);
  file.close();
  return success;
 }

 bool Network::write(ostream &s) const {
  if (!s.good() ) {
    return false;
  }
  s << "NETWORK" << "\n"
    << "input_size " << inputLayer.size() << "\n"
    << "layers " << layerCount << "\n";
  shared_ptr<Layer> currentLayer;
  if (firstHidden) {
    currentLayer = firstHidden;
  } else {
    currentLayer = outputLayer;
  }

  do {
    currentLayer->write(s);
    currentLayer = currentLayer->nextLayer();
  } while (currentLayer->nextLayer());
  outputLayer->write(s);
  return true;
 }
}
#ifndef NEURAL_NETWORK_H
#define NEURAL_NETWORK_H
```

```
#include "Layer.h"
#include <vector>
#include <memory>
#include <fstream>
#include <iostream>

using namespace std;

namespace neural {
 /**
  * A back-propagation neural network.
  */
 class Network {
 public:
  /**
   * Construct a new Neural Network.
   * @param input the number of input neurons
   * @param output the number of output neurons
   * @param hidden an optional vector containing the number of neurons for each hidden layer
   */
  Network(int input, int output, vector<int> hidden = vector<int>());
  static Network read(string &filename);
  static Network read(istream &s);

  /**
   * Train the net with a single test case
   * @param learning_rate controls the speed of learning (see Neuron::updateWeights() )
   * @return the error for this case after back propagation
   */
  double trainSingle(vector<double> input, vector<double> expected_output, double learning_rate
= 0.3);
```

```
    //! Run the neural network for the given input
    vector<double> run(vector<double> input);
    inline int Layers() const { return layerCount; };
    inline int Inputs() const { return inputLayer.size(); };
    inline int Outputs() const { return outputLayer->size(); };
    inline vector<double> Output() const { return outputLayer->Output(); };
    bool write(string &filename) const;
    bool write(ostream &s) const;
  private:
    Network(int input, shared_ptr<Layer> hidden, shared_ptr<Layer> output);
    int layerCount;
    //! Input layer just consists of data
    vector<double> inputLayer;
    // Layers in between are accessed via prev and next pointers
    shared_ptr<Layer> firstHidden;
    shared_ptr<Layer> outputLayer;
    double calculateError();
  };
}

#endif
#include "neural/Neuron.h"
#include <cstdlib>
#include <cassert>

namespace neural {
  Neuron::Neuron(int inputSize,
                 std::function<double (double)> activation_func,
                 std::function<double (double)> deriv_func) :
    activationFunction(activation_func),
    derivFunction(deriv_func)
```

```
{
 initWeightsRandom(inputSize);
}

Neuron::Neuron(std::vector<double> w,
                std::function<double (double)> activation_func,
                std::function<double (double)> deriv_func) :
  activationFunction(activation_func),
  derivFunction(deriv_func)
{
  initWeights(w);
}

Neuron Neuron::read(std::istream &s,
                    std::function<double (double)> activation_func,
                    std::function<double (double)> deriv_func)
{
  if (!s.good()) return Neuron(0);
  std::string keyword;
  s >> keyword;
  if(keyword != "NEURON") return Neuron(0);

  s >> keyword;
  if (keyword != "size") return Neuron(0);
  int dataSize;
  s >> dataSize;

  s >> keyword;
  if (keyword != "data") return Neuron(0);

  char c = s.get();
  if (c != ' ') return Neuron(0);
```

```
  std::vector<double> w;
  double current = 0;
  for (int i = 0; i < dataSize; i++) {
    s.read(reinterpret_cast<char*>( &current), sizeof(current));
    w.push_back(current);
  }

  do {
    c = s.get();
  } while (s.good() && c != '\n');

  return Neuron(w, activation_func, deriv_func);
}

void Neuron::updateOutput(std::vector<double> inputs) {
  assert(inputs.size() + 1 == weights.size());
  output = weights.back();
  for (int i = 0; i < inputs.size(); i++) {
    output += inputs[i] * weights[i];
  }
  output = activationFunction(output);
}

void Neuron::updateDelta(double delta_sum) {
  delta = derivFunction(output) * delta_sum;
}

void Neuron::updateWeights(std::vector<double> input, double learning_rate) {
  assert(input.size() + 1 == weights.size());
  for(int i = 0; i < weights.size() - 1; i++) {
    weights[i] += input[i] * delta * learning_rate;
  }
```

```
  weights[weights.size() - 1] = delta * learning_rate;
 }

 bool Neuron::write(std::ostream &s) const {
  assert(sizeof(double) == sizeof(char*));
  if ( !s.good() ) return false;
  s << "NEURON" << "\n"
   << "size " << weights.size() << "\n"
   << "data ";
  double current = 0.0;
  for ( int i = 0; i < weights.size(); i++) {
   current = weights[i];
   s.write(reinterpret_cast<char*>(&current), sizeof(current));
   if (!s.good()) {
        return false;
   }
  }
  s << std::endl;
  return true;
 }

 void Neuron::initWeightsRandom(int inputSize) {
  for (int i = 0; i < inputSize + 1; i++) {
   weights.push_back((((double) rand()) / ((double) (RAND_MAX/2))) - 1); // Random value between -0.5 and 0.5
  }
 }

 void Neuron::initWeights(std::vector<double> w) {
  for (std::vector<double>::iterator it = w.begin(); it != w.end(); it++) {
   weights.push_back(*it);
```

```
    }
  }
}
#ifndef NEURAL_NEURON_H
#define NEURAL_NEURON_H

#include <vector>
#include <functional>
#include <cmath>
#include <fstream>
#include <iostream>

#include "Activation.h"

namespace neural {
  /**
   * This class models a single neuron in one of the @link Layer Layers @endlink that make up a Network
   *
   * Each Neuron holds a set of input weights, an additional bias weight, and an activation function
   * along with that function's derivative.
   */
  class Neuron {
  public:

    /**
     * Create a new Neuron
     * @param inputSize how many inputs the Neuron has, not including its bias
     * @param activation_func a lambda or function pointer describing the Neuron's
     *                        activation function -- defaults to tanh_func
     * @param deriv_func a lambda or function pointer describing the derivative of
     *                 \p activation_func, except input is \p activation_func(x) rather
```

```
 *              than x -- this lets us pass Neuron::Output() to \p deriv_func directly
 *              (defaults to tanh_deriv)
 */
Neuron(int inputSize,
        std::function<double (double)> activation_func = activation::tanh_func,
        std::function<double (double)> deriv_func = activation::tanh_deriv
        );

/**
 * Create a neuron with specific weights (including one for its bias!)
 * @param weights the weights for this neuron, in order -- last one is for the bias
 * @param activation_func a lambda or function pointer describing the Neuron's
 *                  activation function -- defaults to tanh_func
 * @param deriv_func a lambda or function pointer describing the derivative of
 *              \p activation_func, except input is \p activation_func(x) rather
 *              than x -- this lets us pass Neural::Output() to \p deriv_func directly
 *              (defaults to tanh_deriv)
 */
Neuron(std::vector<double> w,
        std::function<double (double)> activation_func = activation::tanh_func,
        std::function<double (double)> deriv_func = activation::tanh_deriv
        );

/**
 * Construct Neuron from input stream \p s with the given activation function
 * (defaults to tanh) -- if anything goes wrong,
 * this will return a Neuron with an InputSize() of 0!
 */
static Neuron read(std::istream &s,
        std::function<double (double)> activation_func = activation::tanh_func,
        std::function<double (double)> deriv_func = activation::tanh_deriv
        );
```

```
    /**
     * Update this Neuron's output value -- use Neuron::Output() to access it
     * @param inputs the input values for this neuron (typically outputs of all neurons in the previous Layer)
     */
    void updateOutput(std::vector<double> inputs);

    /**
     * Get the current output value -- this is only updated by Neuron::updateOutput(), *not* automatically!
     */
    inline double Output() const { return output; }

    /**
     * Update the delta value
     *
     * This is the back-propagation this type of neural network gets its name from.
     * @param delta_sum the summed deltas of the following layer, weighed by the input weight corresponding to this Neuron.
     *   Example (this Neuron has Index 0 in its layer, next layer has 2 neurons):
     *   \f$\text{delta_sum} = \text{next_layer.neurons[0].delta} * \text{next_layer.neurons[0].weights[0]}
     *                + \text{next_layer.neurons[1].delta} * \text{next_layer.neurons[1].weights[0]}\f$
     */
    void updateDelta(double delta_sum);

    /**
     * Get current delta (need to call Neuron::updateDelta() first!)
     * @param wrt (with-regard-to) optionally multiply delta with weight for one of this Neuron's inputs
     */
    inline double Delta(int wrt = -1) const {
      if (wrt < 0 || wrt >= weights.size() - 1) {
```

```
        return delta;
    } else {
        return delta * weights[wrt];
    }
  }

  /**
   * Get the number of input weights for this Neuron -- this is equal to the number of neurons in the previous layer
   */
  inline int InputSize() { return weights.size() - 1; };

  /**
   * Update the weights using the delta calculated with Neuron::updateDelta()
   * @param input the input to this layer
   * @param learning_rate the change in weight will be multiplied with this --
   *                      higher values mean faster convergence, but are more likely
   *                      to overshoot the actual minimum.
   */
  void updateWeights(std::vector<double> input, double learning_rate);

  /**
   * Write this Neuron's data to an output stream. Size is written in ASCII, weights
   * are stored as a binary blob -- the activation function isn't saved at all!
   *
   * @return true if writing to the stream was successfull, false if it was not
   */
  bool write(std::ostream &s) const;

 protected:
  /**
```

```
     * Initialize this Neuron's weights to random values between -0.5 and 0.5 -- there will be one
more
     * weight than \p inputSize to account for the Neuron's bias
     */
    void initWeightsRandom(int inputSize);

    /**
     * Initialize this Neuron's weights (including the bias weight) to the given values
     * @param w a vector containing all weights for this neuron, including the bias weight
     */
    void initWeights(std::vector<double> w);
    std::function<double (double)> activationFunction;
    std::function<double (double)> derivFunction;
    //! Size will be inputSize + 1 -- the additional item is the bias
    std::vector<double> weights;
    double output;
    double delta;
  };
}
#endif
```