

ВВЕДЕНИЕ

В течение многих лет существовало мнение, что психически больные не хотели бы использовать технологии для лечения их состояния, в отличие от, например, людей с астмой или сердечной недостаточностью. Некоторые также утверждали, что лечение должно выполняться лицом к лицу, чтобы быть эффективным, и что технологии могут только испугать больного или ухудшить его паранойю.

Однако, результаты недавних исследований ряда престижных институтов, включая Гарвард, опровергают эти утверждения. Опыты показывают, что психически больные, даже те, кто тяжело болен, например, шизофренией, могут успешно контролировать свое состояние с помощью смартфонов, компьютеров и носимых датчиков. И эти инструменты только начало. В течение нескольких лет, новое поколение технологий обещает произвести революцию в практике психиатрии.

Когда пациент посещает психиатра или терапевта, большая часть времени первичного визита уходит на осмотр его симптомов, таких как сон, уровень энергии, аппетит и способность фокусироваться. Что также может быть очень сложным; депрессия, как множество других психических заболеваний, влияет на способность человека думать и запоминать.

Больной, скорее всего, уйдет со списком упражнений и рецептом лекарств. Есть вероятность, что медикаментозное лечение не будет эффективным, по крайней мере, в течение нескольких недель, план упражнений будет проигнорирован, что отрицательно отобразится на ход прогресса. К сожалению, врач ничего не узнает о пациенте до последующей встречи с ним.

Технологии могут положительно повлиять на сложившуюся ситуацию, предоставляя возможность в режиме реального времени следить за больным вне офиса. Вместо того чтобы полагаться только на упомянутые пациентом симптомы, доктор может взглянуть на данные о поведении больного с его персонального телефона и носимых датчиков. Психиатр может даже порекомендовать пациенту начать использовать такое устройство перед первым визитом.

Современные технологии и методы обработки данных, в частности машинное обучение, позволяют предоставить множество полезной информации из данных, которые, на первый взгляд, имеют мало общего со здоровьем человека.

Активное развитие технологий искусственного интеллекта, в настоящее время, привело к тому, что эти технологии широко применяются во многих

областях. Компьютер как обычный вычислитель способен быстро и точно вычислять заданные параметры. Объединяя эти преимущества с моделями человеческого разума, технологии искусственного интеллекта улучшают человеческую жизнь, и даже, помогают выполнять задачи за пределами человеческих способностей. Искусственный интеллект – это комплексная и сложная наука, имеющая много пересечений с другими науками.

Искусственные нейронные сети дают многообещающие перспективы в развитии, а программное обеспечение имеет огромное преимущество от их использования. Кроме того, каждая реализуемая задача имеет неограниченный и нестандартный набор методов решения. В магистерской диссертации рассматривается возможность применения нейронной сети при решении задачи распознавания психологического состояния человека.

Актуальность магистерской диссертации заключается в том, что на сегодняшний день людей с психологическими проблемами и тяжелыми расстройствами гораздо больше, чем специалистов, способных с этим справиться. Частично эта проблема решается разработкой унифицированных протоколов лечения и доказательных моделей психотерапии, которые призваны решить как можно более широкий круг проблем как можно эффективнее. Для ЭВМ усвоить эти четкие критерии и последовательность действий не составляет труда.

ГЛАВА 1

АНАЛИЗ МЕТОДОВ И СПОСОБОВ РАСПОЗНАВАНИЯ ПСИХОЛОГИЧЕСКОГО СОСТОЯНИЯ ЧЕЛОВЕКА

1.1 Цифровая психотерапия

Современные средства коммуникации предоставляют большой выбор форм общения. Благодаря видеоконференциям и мессенджерам общаться с психологом или психотерапевтом можно из дома, офиса и из вагона метро. Разрабатываются отдельные протоколы и этические стандарты онлайн-психотерапии, а специальные кодировки сохраняют конфиденциальность переписки, телефонных разговоров или видеозвонков. Это все значительно облегчает клиентам и пациентам процесс обращения за помощью и снижает риск отговорок и сопротивления.

Сегодня мобильные приложения позволяют фиксировать собственный прогресс в самопознании или предостерегают от потенциальных опасностей. Например, в MoodKit можно отслеживать собственное негативное мышление, вести журнал настроения и планировать активности, которые помогут по-новому выстраивать свою жизнь изо дня в день. Некоторые приложения, такие как Mobilyze, отслеживают местоположение пользователя и активность его социальных взаимодействий онлайн и в реальной жизни, чтобы вовремя распознать первые признаки депрессии. И это не говоря уже о многочисленных приложениях для медитации и поддержания здорового образа жизни.

Одно из наиболее перспективных направлений в психотерапии нового времени – это использование виртуальной реальности. Клинические психологи и нейробиологи активно исследуют возможности применения VR-очков в лечении депрессии и последствий психических травм, а также при подготовке специалистов помогающих профессий. Например, в Университете Барселоны был проведен эксперимент, в рамках которого люди на камеру описывали свои психологические проблемы, а затем при помощи очков виртуальной реальности смотрели на себя со стороны и проводили самим себе психологические консультации. Наиболее эффективных результатов удалось достичь тогда, когда участники перевоплощались в Зигмунда Фрейда: эта иллюзия, по всей видимости, вселяла в них уверенность в собственной компетентности.

В другом VR-исследовании, проведенном совместно командами Университетского колледжа Лондона и Университета Барселоны, испытуемые управляли персонажем, чтобы тот утешил расстроенного ребенка. Затем

в формате виртуальной реальности они сами оказывались на месте персонажа, по отношению к которому проявляют тепло и поддержку. Ощувив на себе собственное неподдельное сострадание, участники встали на путь отказа от самокритики, научились по-новому обращаться со своими переживаниями, и в результате у них значительно снизилась выраженность депрессивных симптомов [4].

Кроме того, на основе виртуальной реальности разрабатываются компьютерные программы и мобильные приложения для оказания психологической помощи. Так, Virtual Reality Medical Center позволяет симулировать ситуации, вызывающие у пациента тревогу. Одновременно с этим устройство биологической обратной связи фиксирует физиологические показатели стресса, такие как пульс, дыхательный ритм и кожно-гальваническая реакция, и пациент учится усилием воли ослаблять выраженность своей тревоги и паники в трудных ситуациях – а затем переносит этот опыт произвольной саморегуляции в реальную жизнь.

Эндрю Ён (Andrew Ng), сооснователь проекта по созданию самообучающегося искусственного интеллекта Google Brain, возглавит работу над чат-ботом-психотерапевтом Woebot, которым занимаются ученые Стэнфордского университета. Этот чат-бот позволяет пациентам с депрессией и тревожными расстройствами вести переписку в ключе когнитивно-поведенческой психотерапии (КПТ) – наиболее структурированного и эффективного направления по работе с такими состояниями.

Этот подход основывается на идее того, что источником психологических проблем и страданий являются не сами события, которые с нами происходят, а наше отношение к ним. В связи с этим в работе становится главным не то, что вы пережили в детстве или что вас пугает в будущем, а то, как вы себя ведете и чувствуете прямо сейчас. Если изменить свои текущие мысли и поведение, это поможет выстроить более адекватную и реалистичную картину мира и чувствовать себя лучше в любой момент времени. Именно поэтому Woebot подсказывает, когда вы по отношению к себе несправедливы или делаете слишком обобщенные выводы. Скажем, после ссоры с супругом или супругой вы пожалуетесь чат-боту, что вас никто не любит и не понимает, а он вам ответит, что это когнитивная ошибка, которая искажает ваше мышление. В подтверждение своей правоты он еще и приведет целый ряд примеров из вашей же жизни, чтобы опровергнуть ваши деструктивные убеждения. И это только один из аспектов его работы и постоянного самообучения.

Сами разработчики Woebot утверждают, что вовсе не стремятся лишить живых психотерапевтов их рабочих мест, а лишь стараются предоставить

желающим альтернативу. И все же идея роботов-терапевтов встречает неоднозначные отклики в научном и практическом сообществе.

На сегодняшний день людей с психологическими проблемами и тяжелыми расстройствами гораздо больше, чем специалистов, способных с этим справиться. Частично эта проблема решается разработкой унифицированных протоколов лечения и доказательных моделей психотерапии, которые призваны решить как можно более широкий круг проблем как можно эффективнее. Для робота усвоить эти четкие критерии и последовательность действий не составляет труда. И в этом смысле разработчики AI-психотерапии вряд ли станут останавливаться на достигнутом [5].

1.2 Источники поведенческих данных

Анализ больших данных позволяет получить точную информацию о человеке. Большие данные представляют собой огромный объем информации разного типа: картинки, видео, текст, геоданные, веб-журналы, машинный код. Вся информация находится в различных хранилищах и трудно поддается анализу с помощью традиционных методов. Для этого используются специализированные технологии, включая искусственный интеллект и машинное обучение. Рассмотрим возможные источники данных, которые могут быть использованы для сбора поведенческой информации с целью определения психологического состояния человека:

GPS данные смартфона могут показать передвижения человека, которые, в свою очередь, отражают психическое здоровье человека. Посредством корреляции GPS измерений смартфона пациента и его симптомов депрессии, исследования 2016 года Центра Технологий Поведенческого Вмешательства в северо-западном Университете в Чикаго, было найдено, что когда люди в депрессии, они имеют тенденцию находиться дома больше чем, когда они чувствуют себя хорошо. Аналогично, люди входящие в маниакальную стадию биполярного расстройства, могут быть активнее и в движении. Консорциум по мониторингу, лечению и предсказанию стадий биполярного расстройства провел многочисленные исследования, которые демонстрируют, что этот тип данных может быть использован для предсказания течения биполярного расстройства [6].

Где GPS данные не доступны, могут быть использованы Wi-Fi и Bluetooth. Исследования Дрора Бен-Зеева, Университет Вашингтона, Сиэтл,

показали, что Bluetooth-радиоприемники в смартфонах могут быть использованы для мониторинга места положения людей страдающих шизофренией в госпитале. Данные собранные через Wi-Fi сети могли бы также показать избегают ли пациенты, склонные к алкоголизму, бары и посещают ли встречи группы поддержки [6].

Данные акселерометра смартфона или фитнес-трекера могут предоставлять более точные детали о передвижениях человека, определять тремор, который может быть побочным эффектом применения наркотиков, и захватывать шаблонные движения в упражнениях. Тестирование приложения названного CrossCheck недавно продемонстрировало как этот тип данных, в комбинации с другой информацией полученной с телефонов, может содействовать прогнозированию симптомов шизофрении, предоставляя информацию о сне и шаблонах поведения. Статья из Американского журнала психиатрии, Ипсит Вахия и Даниэль Сивел, описывает как у них получилось лечить пациентов с особо сложным случаем депрессии используя данные акселерометра. Пациент сообщал, что он был физически активен и проводил мало времени в постели, но данные с его фитнес-трекера показали, что его воспоминания были ложны, доктора, таким образом, верно диагностировали его состояние как депрессию скорее, чем, скажем, расстройство сна.

Отслеживание частоты телефонных звонков и текстовых сообщений может показать насколько человек общителен и указать на его психологические изменения. Когда одна из исследовательских групп Monarca взглянула на журнал входящих и исходящих текстовых сообщений и телефонных звонков, они пришли к выводу что изменения в этом журнале могли бы быть полезны для отслеживания депрессии также как и мании в биполярном расстройстве [7].

Физиологические данные, такие как пульс, гальваническая реакция кожи могут также пролить свет на умственное благополучие человека. Такого рода данные уже собираются некоторыми устройствами. Множество исследований показали, что изменчивость пульса может быть использована для отслеживания тяжести биполярного расстройства и шизофрении. Гальваническая реакция кожи, измерение электропроводимости кожи, которая зависит от состояния потовых желез и контролируется компонентом нервной системы, могут свидетельствовать о биологической активности, которая может быть сверхактивной при тревожном расстройстве.

В конечном счете, психиатрические расстройства это расстройства головного мозга, и использование электроэнцефалографии (ЭЭГ) для отслеживания активности головного мозга уже давно принятая практика в психиатрических исследованиях и лечении биологической обратной связи.

Более новые, потребительские ЭЭГ гаджеты в наше время могут быть приобретены в магазине, но ещё не понятно насколько они эффективны.

Мобильный рынок предлагает пользователям смартфонов массу интересных приложений для поддержания здорового образа жизни и отслеживания собственных спортивных достижений. Сейчас подобные приложения могут измерять пульс или считать шаги, но делают это приблизительно и требуют от пользователей ввода определенных данных.

С помощью машинного обучения можно значительно улучшить показатели фитнес-приложений путем постоянного отслеживания физической активности человека без дополнительных действий с его стороны. То есть, пользователю не потребуется переключать режимы приложения, когда он переходит на бег или езду на велосипеде. Однако это можно реализовать при условии использования сенсоров, интегрированных с приложением.

Подобная задача решается в приложении для людей, подверженных приступам мигрени, эпилепсии или обморокам. Приложение используется для мониторинга за состоянием пациентов и предупреждения критических ситуаций, отслеживает их нейробиологические сигналы через сенсоры смартфонов и обрабатывает информацию с помощью модели машинного обучения. Если возникает риск приступа, приложение «сигнализирует» об этом, отправляя сообщение близким, которые могут прийти на помощь [8].

Группа исследователей под руководством Питера Глур из Массачусетского технологического института создала новую систему, которая позволяет людям оценивать свое эмоциональное состояние на протяжении длительных промежутков времени, просто нося умные часы на руке. Они написали приложение для часов Pebble 2 и смартфонов под управлением Android. Приложение на часах собирает данные об уровне активности и сердцебиения пользователя, а на смартфоне данные о его перемещениях, погоде, уровне освещенности вокруг и времени.

Для оценки настроения на основе этих данных авторы использовали алгоритм машинного обучения типа «случайный лес». Исследователи решили представить варианты эмоционального состояния в виде двумерного пространства «активность-удовольствие». Каждая из этих двух величин была разбита на три уровня, и, таким образом, они формировали девять комбинаций, описывающих основные типы эмоций (рисунок 1.1).

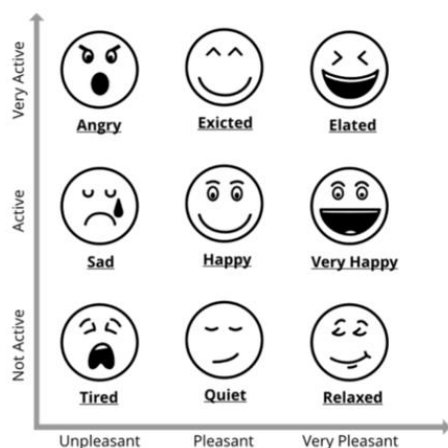


Рисунок 1.1 – Основные типы эмоций

Приложение объединяет множество данных и формирует на их основе эмоцию, которую в данный момент испытывает человек. Четыре раза в день оно сообщает ему эту оценку, после чего пользователь может подтвердить ее или изменить, если она не соответствует его настроению. Разработчики протестировали приложение на 60 добровольцах, которые участвовали в исследовании в течение двух месяцев. За счет применения машинного обучения модель, натренированная на отдельных людях достигла точности определения настроения в 86,68 процента, а общая модель 85,1 процента.

Исследователи из MIT и ранее разрабатывали системы для измерения человеческих эмоций, в том числе и на основе носимых устройств. Недавно они научили умные часы определять эмоциональный тон беседы за счет измерения биометрических показателей владельца, а также записи речи собеседников. А в прошлом году они создали систему, которая за счет радиоволн измеряет пульс и дыхание человека, и на основе этих данных предсказывает его эмоциональное состояние [9].

1.3 Существующие решения

Чтобы помочь человеку справиться в сложной психологической ситуации, компании ежегодно разрабатывают и совершенствуют виртуальных помощников – приложения, которые способны оказать поддержку и помощь. Так, Siri, виртуальный помощник Apple, может вызвать службу спасения, если распознает в вопросе суицидальную направленность. Рассмотрим некоторые приложения, которые оказывают психологическую помощь носителям мобильных устройств.

7 Cups. Приложение дает возможность бесплатно пообщаться с профессиональными слушателями, которые оказывают эмоциональную поддержку, консультируют и проводят терапию, чтобы помочь преодолеть депрессию, тревогу и стресс. Вы можете выражать свои мысли без страха быть осужденным знакомыми или родными. Сообщения абсолютно анонимны и удаляются из беседы.

Слушателями являются волонтеры, которые работают бесплатно. Создатели приложения утверждают, что слушатели предоставляют свои услуги через их приложение, только потому что хотят помогать. Кроме того, все слушатели проходят подготовку и должны получить проходной балл на последнем испытании, чтобы работать на 7 Cups.

На данный момент в системе есть более 70000 обученных слушателей и лицензированных врачей. Каждый слушатель имеет список категорий, на которых он специализируется, начиная от панических атак и издевательства до пищевых расстройств. Также есть возможность выбрать слушателей на основе языка или страны.

Чат-бот Karim. Приложение было создано для решения проблемы, связанной с недоступностью медицинской и психологической помощи беженцам, пережившим серьезные жизненные трудности. Он ведет прием на арабском языке, оценивает состояние по речи и по сообщениям, после чего дает совет. Проект был разработан стартапом X2AI из Кремниевой долины, в основе чат-бота лежит искусственный интеллект по имени Karim.

Чатбот обрабатывает естественную речь и анализирует эмоциональное состояние собеседника по его сообщениям, после чего предлагает поддержку, совет и помощь.

Для того, чтобы беженцы смогли пообщаться с Каримом, X2AI заключила партнерство с организацией Field Innovation Team (FIT), которая специализируется на технологиях помощи в бедствиях. Чатбот не позиционируется как психотерапевт так как это может вызвать у людей предубеждение. Карим ведет себя как друг и начнет диалог издалека, например, спрашивает о любимых фильмах, а затем задает более личные вопросы. Пока что система работает в тестовом режиме.

Чат-бот Tess. С помощью этого чатбота пациенты могут получить поддержку удаленно, в любое время, в том числе когда их лечащий врач недоступен. Авторы приложения справедливо указывают, что разговоры с обычным психотерапевтом часто следуют ограниченному количеству шаблонов и путей, что оставляет много возможностей для автоматизации. Боты X2AI обнаруживают закономерности в формулировках фраз, скорости набора текста, длине предложений, грамматических ошибках, выявляя корреляции с

различными эмоциональными состояниями. Это позволяет системе замечать скрытые эмоции, как это делают люди-терапевты.

На рисунке 1.2 изображено мобильное приложение чат-бот Tess.

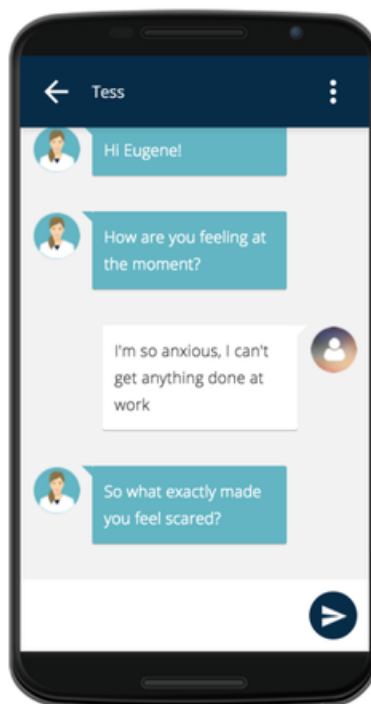


Рисунок 1.2 – Чат-бот Tess

X2AI обеспечивают связь с «терапевтом», который адаптируется к каждому пользователю, с которым взаимодействует. Tess обнаруживает изменения в настроении человека. Благодаря обработке естественного языка бот может определить, достигает ли человек опасного уровня депрессии, где уже требуется вмешательство специалистов.

Когда Tess обнаруживает, что пользователь действительно находится в бедственном положении, например, говорит о самоубийстве, то передает консультационную сессию реальному психологу-человеку. Была разработана специальная версия Tess, предназначенная для работы с ветеранами войны, у которых есть выраженные посттравматические стрессовые расстройства.

X2AI предоставляет ряд персонализированных услуг в области психического здоровья, такие как психотерапия, психологический коучинг и даже когнитивная поведенческая терапия.

X2AI создали также бота Nema, который специализируется на лечении детей с диабетом и «Сару», призванную помочь подросткам справиться с одиночеством. Некоторые из ботов предназначены для удовлетворения потребностей конкретного клиента (X2AI работает с несколькими крупными

американскими и европейскими поставщиками медицинских услуг), в то время как другие стремятся удовлетворить потребности кризисных территорий (министерство здравоохранения Ливана и Мировая продовольственная программа ООН выразили заинтересованность в осуществлении более масштабных программ с использованием психологических чатботов). Другие боты в настоящее время разрабатываются, чтобы помочь людям, пострадавшим от насилия в Бразилии и Нигерии.

Sentrian – система удаленного наблюдения за пациентами, которая прогнозирует ухудшение состояния хронических больных. Для этого система собирает данные с носимых устройств пациентов и регистрирует малейшие физиологические отклонения. Сам пациент может их и не заметить. Если возникает угроза обострения, то система отправляет отчет лечащему врачу.

На рисунке 1.3 изображено веб-приложение Sentrian.

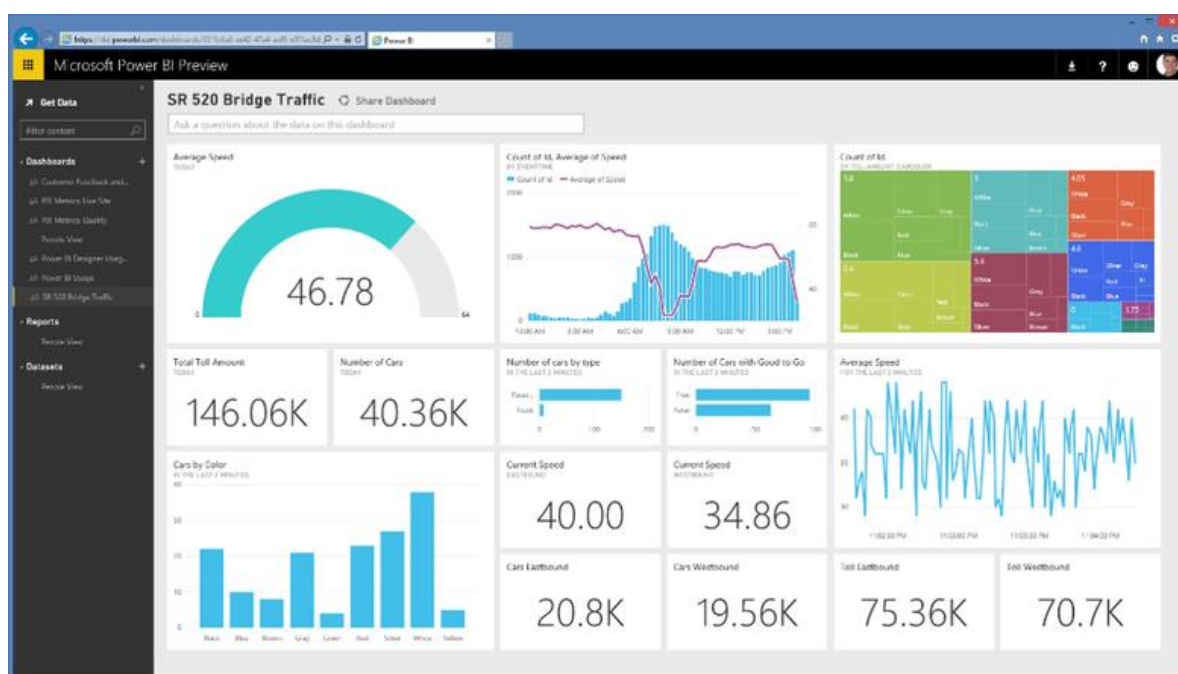


Рисунок 1.3 – Приложение Sentrian

Sentrian сотрудничает с госпиталями и врачами, которые наблюдают за хроническими больными. Совместно с медиками разработчики подбирают носимые устройства и датчики для конкретного пациента.

Искусственный интеллект анализирует данные, полученные с сенсоров, согласно индивидуальным настройкам. То есть, если пациент пережил инфаркт, то система более пристально отслеживает артериальное давление и пульс.

Лечащий врач непосредственно участвует в настройке платформы искусственного интеллекта. Поэтому разработчики назвали эту технологию

«машинное обучение под управлением врача» — Clinician-Directed Machine Learning. Она позволяет лечащему врачу вводить параметры наблюдения за пациентом, используя не программный код, а интерфейс на естественном языке.

CareSkore. Приложение для прогнозирования течения болезни.

CareSkore использует машинное обучение для прогнозирования течения болезни и вероятности смертности. CareSkore объединяет клинические, социально-экономические, демографические и поведенческие данные пациентов, чтобы нарисовать целостную картину, которую могут использовать врачи и страховые компании, чтобы обеспечить лучший профилактический уход.

CareSkore также помогает пациентам, которые могут использовать систему для информирования врачей о новых симптомах или задавать вопросы о своем состоянии. Компания планирует использовать умные датчики, чтобы пациенты только в крайне редких случаях обращались в больницу. Датчики в режиме реального времени будут собирать всю информацию о здоровье и передавать лечащему врачу. Система сможет на основе данных датчиков давать рекомендации по лечению.

1.4 Техническое задание

Разрабатываемое программное средство предназначено для исполнения следующих процессов:

- сбор и хранение поведенческой информации пациента;
- предоставление возможности отслеживать поведение пациента через интернет;
- автоматизация процесса распознавания психологического состояния человека;

Целью проекта является:

- повышение эффективности исполнения работы психотерапевта за счет предоставления доктору объективной информации о поведении пациента вне офиса;
- повышение качества определения диагноза за счет оперативности представления, полноты, достоверности и удобства форматов отображения информации;
- выбор информационной модели нейронной сети и алгоритма классификации психологического состояния пациента на основе собранной поведенческой информации;
- полная информационная открытость и создание банков данных.

Основные функциональные возможности программного обеспечения:

- создание нового пациента;
- создание нового доктора;
- возможность отслеживать звонки пациента;
- возможность отслеживать сообщения пациента;
- возможность отслеживать перемещения пациента;
- предоставление детальной информации о пациенте;
- определение возможного психологического состояния пациента.

Веб-сервер будет получать данные на вход, преобразовывать их в соответствии с бизнес-требованиями и осуществлять вывод данных. Данные на вход и выход будут передаваться по сети с использованием протокола HTTP.

Входные данные:

- это запросы клиентов (в основном веб-браузеров)
- данные поступающие со смартфона пациента (история звонков, сообщений и передвижений).

Выходные данные:

- HTML-код подготовленных веб-страниц;
- данные в формате JSON.

Требования к составу и параметрам технических и программных средств:

1. Скорость работы интернета, состояние сети и компьютеров будет достаточно высоким, чтобы избежать задержек в обработке данных. максимальное число параллельных работающих пользователей приложения, поддержка которого ожидается от приложения в любой момент времени определяется разработчиком.

2. Сервер на базе Microsoft. Система будет основана на технологии .NET. Это означает, что для системного хостинга может использоваться только сервер на базе Microsoft.

3. Каждое рабочее место будет оснащено ПК с операционной системой Microsoft Windows. При эксплуатации системы на рабочих местах пользователей устанавливаются Google Chrome 40, Microsoft Edge, Safari 9.

4. Мобильное Android-устройство начиная с версии 9.0.

ГЛАВА 2

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1 Проектирование системы

В этой главе будет спроектирована архитектура системы, а также база данных для приложения. Также, необходимо рассмотреть используемые технологии, которые будут применяться при разработке данного программного обеспечения. Существует огромное количество языков программирования и технологий, выбор наиболее удовлетворительных при заданных условиях не всегда оказывается простым. Необходимо рассмотреть все преимущества и недостатки, и, применительно к данной задаче, использовать наиболее подходящие технологические решения.

Клиент-сервер (англ. Client-server) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Физически клиент и сервер – это программное обеспечение. Обычно они взаимодействуют через компьютерную сеть посредством сетевых протоколов и находятся на разных вычислительных машинах, но могут выполняться также и на одной машине. Программы расположенные на сервере ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent или потоковое мультимедиа) или сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями, просмотр веб-страниц во всемирной паутине) [26].

Преимущества:

1. Отсутствие дублирования кода программы-сервера программами-клиентами.
2. Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
3. Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.

Недостатки:

1. Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть. Неработоспособным сервером следует считать сервер, производительности которого не хватает на обслуживание всех клиентов, а также сервер, находящийся на ремонте, профилактике и т.п.

2. Поддержка работы данной системы требует отдельного системного администратора.

3. Высокая стоимость оборудования.

Трехуровневая архитектура, или трехзвенная архитектура (англ. three-tier или англ. Multitier architecture) – архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиентского приложения (обычно называемого «тонким клиентом» или терминалом), сервера приложений, к которому подключено клиентское приложение, и сервера базы данных, с которым работает сервер приложений.

Клиент – это интерфейсный (обычно графический) компонент, который представляет первый уровень, собственно приложение для конечного пользователя. Первый уровень не должен иметь прямых связей с базой данных (по требованиям безопасности), быть нагруженным основной бизнес-логикой (по требованиям масштабируемости) и хранить состояние приложения (по требованиям надежности). На первый уровень может быть вынесена и обычно выносятся простейшая бизнес-логика: интерфейс авторизации, алгоритмы шифрования, проверка вводимых значений на допустимость и соответствие формату, несложные операции (сортировка, группировка, подсчет значений) с данными, уже загруженными на терминал.

Сервер приложений располагается на втором уровне. На втором уровне сосредоточена большая часть бизнес-логики. Вне его остаются фрагменты, экспортируемые на терминалы (см. выше), а также погруженные в третий уровень хранимые процедуры и триггеры [27].

Сервер базы данных обеспечивает хранение данных и выносятся на третий уровень. Обычно это стандартная реляционная или объектно-ориентированная СУБД. Если третий уровень представляет собой базу данных вместе с хранимыми процедурами, триггерами и схемой, описывающей приложение в терминах реляционной модели, то второй уровень строится как программный интерфейс, связывающий клиентские компоненты с прикладной логикой базы данных.

Достоинства трёхзвенной архитектуры клиент-сервер:

- масштабируемость;
- конфигурируемость – изолированность уровней друг от друга позволяет (при правильном развертывании архитектуры) быстро и простыми средствами

переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней;

- высокая безопасность;
- высокая надёжность;
- низкие требования к скорости канала (сети) между терминалами и сервером приложений;
- низкие требования к производительности и техническим характеристикам терминалов, как следствие снижение их стоимости.

Терминалом может выступать не только компьютер, но и, например, мобильный телефон.

Недостатки:

- более высокая сложность создания приложений;
- сложнее в разворачивании и администрировании;
- высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования;
- высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений [28].

Для взаимодействия приложения с базой данных будет использоваться принцип разделения ответственности на команды и запросы – CQRS. Принцип предполагает разделение приложения на две отдельные части. Одна часть системы имеет дело с командами, которые отлавливают запросы на изменение состояния, в то время как другая часть системы работает с запросами на извлечение данных. Поэтому для обновления и чтения информации будут использоваться разные модели.

Команды:

- изменяют состояние системы;
- ничего не возвращают;
- хорошо описывают предметную область, как действие пользователя над системой;
- контекст команды хранит нужные для её выполнения данные.

Если возникает необходимость в изменении данных, то выполняется следующая цепочка задач:

- 1) сервис создает команду, которая включает в себя всю необходимую информацию, и отправляет ее диспетчеру команд;
- 2) диспетчер команд находит нужный обработчик и отправляет ему полученную от сервиса команду;
- 3) обработчик генерирует запрос в базу данных, используя информацию размещенную команде.

Запросы:

- не изменяют состояние системы;
- возвращают результат;
- контекст запроса хранит нужные для её выполнения данные.

Если возникает необходимость в извлечении данных, то выполняется следующая цепочка задач:

- 1) сервис создает запрос, который включает в себя всю необходимую информацию для извлечения данных, и отправляет его диспетчеру запросов;
- 2) диспетчер запросов находит нужный обработчик и отправляет ему полученный от сервиса запрос;
- 3) обработчик генерирует запрос в базу данных, используя информацию, размещенную в объекте запроса, и возвращает полученный из базы данных результат [26].

2.2 Архитектура системы

Программное средство состоит из 5 элементов (см. рисунок 2.1):

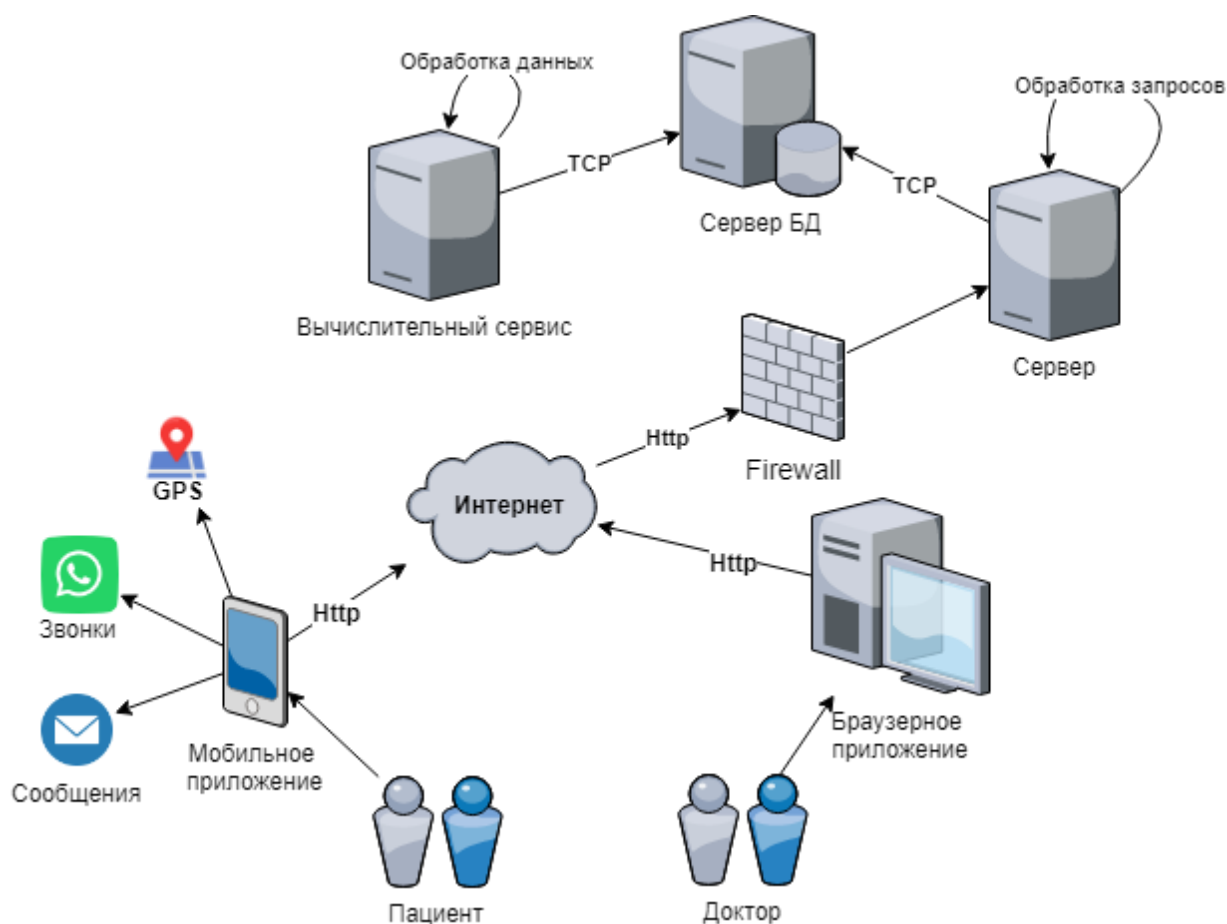


Рисунок 2.1 – Архитектура системы

1. Мобильное приложение, которое устанавливается на телефон пациента, для сбора поведенческой информации. Было написано с использованием языка

Java для мобильной платформы Android. Приложение аккумулирует и отправляет всю информацию о пациенте на веб-сервис.

2. Веб-сервис для получения и предоставления данных о пациенте. Был разработан с использованием технологии ASP.NET Web API 2.0, доступ к ресурсам которого предоставляется через HTTP запросы. Получает всю поведенческую информацию пациента от мобильного приложения и сохраняет её в базу данных.

3. Вычислительный сервис для обработки больших данных пациента. Был разработан с использованием технологии .NET. При поступлении новых данных о пациенте производит сложные вычисления для предсказаний возможного психологического состояния. Запускается на отдельной машине, чтобы снять нагрузку с основного сервиса.

4. Сервер базы данных является хранилищем всей информации о пациенте и пользователях системы.

5. Браузерное приложение, которое позволяет врачу получить доступ к поведенческой информации конкретного пациента, а также увидеть предсказанное системой его психологическое состояние. Приложение было написано с использованием технологий ReactJS, HTML 5.0 и CSS 3.0.

2.3 Проектирование базы данных

В данной системе будет использована реляционная база данных, так как приложение имеет точно описанные доменные модели, что способствует использованию типизированных схем данных на уровне БД. В случае с растущими нагрузками и обрабатываемым объемом данных администратор БД может легко найти уязвимые места и добавить необходимые индексы для улучшения производительности, что в случае с не реляционными БД является достаточно сложной трудоемкой задачей.

База данных состоит из следующих сущностей:

1. User – сущность пользователя системы. Хранит в себе логин пользователя, хеш пароля, соль пароля, адрес электронной почты.
2. Role – хранит имя пользовательской роли (Patient/Doctor);
3. UserRole – промежуточная сущность связи «многие-ко-многим». Хранит роли конкретного пользователя в системе;
4. Contact – сущность контактной информации пользователя. Хранит в себе страну, язык, адрес, город, почтовый индекс, имя и фамилию пользователя.
5. Diagnosis – хранит имя пользовательской роли (Patient/Doctor);

6. UserDiagnosis – промежуточная сущность связи «многие-ко-многим». Хранит историю диагнозов пациента.

7. UserCall – сущность звонка пользователя. Хранит в себе ссылку на пользователя, дату и длительность звонка, имя кому был адресован звонок, имя от кого исходил звонок.

8. UserMessage – сущность сообщения пользователя. Хранит в себе ссылку на пользователя, дату сообщения, имя адресата и адресанта сообщения.

9. UserLocation – сущность геолокации пользователя. Хранит в себе ссылку на пользователя, дату сообщения, имя адресата и адресанта сообщения.

На рисунке 2.2 можно увидеть схему базы данных, спроектированную для хранения всей необходимой информации в системе.

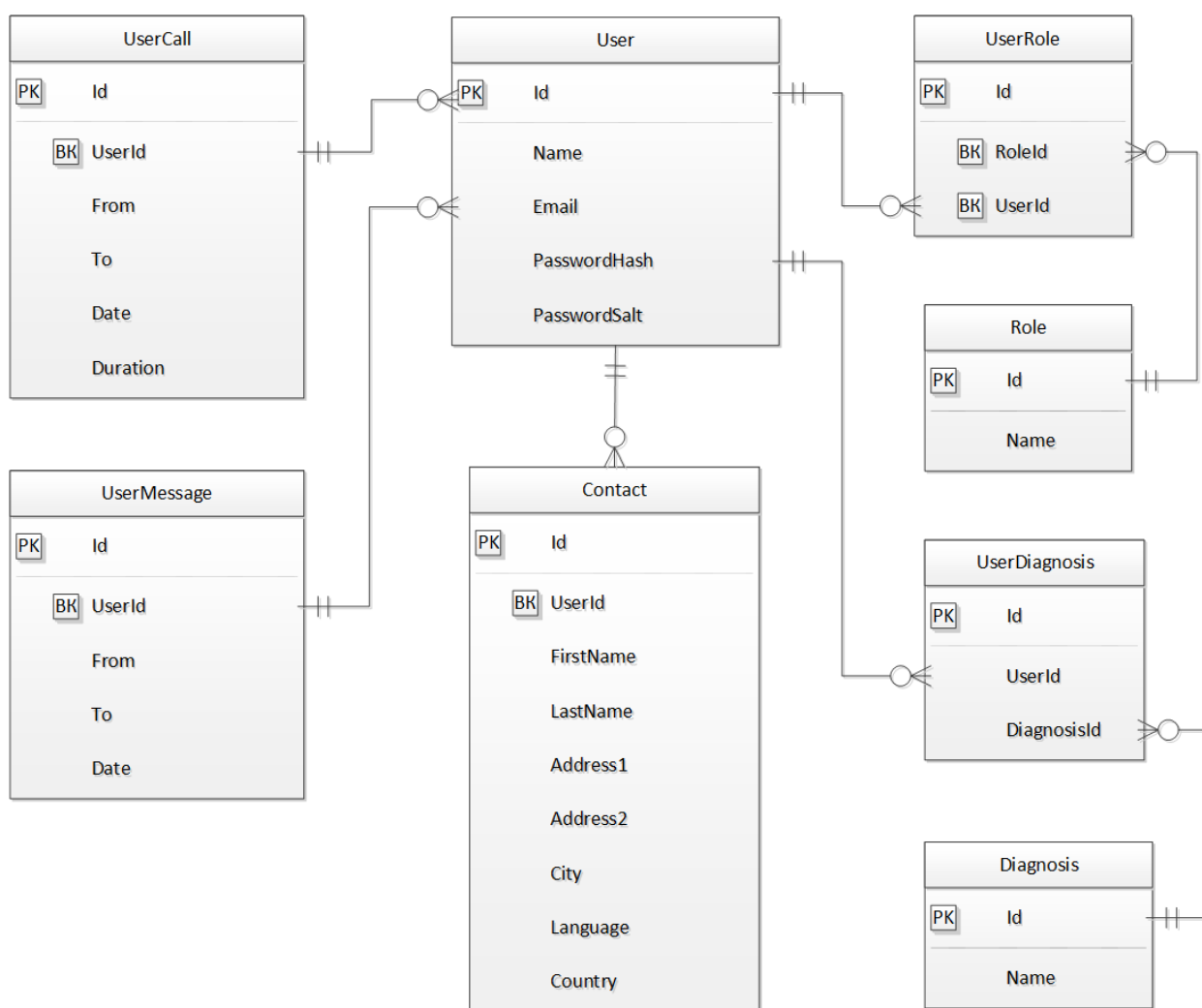


Рисунок 2.2 – Схема базы данных

2.4 Сбор поведенческой информации

Рассмотрим данные, которые могут быть собраны с помощью современного мобильного телефона, для определения психологического состояния человека (см. рисунок 2.3):



Рисунок 2.3 – Данные мобильного телефона

1. GPS данные смартфона показывают передвижения человека. Когда люди в депрессии, они имеют тенденцию находиться дома больше чем, когда они чувствуют себя хорошо. Аналогично, люди входящие в маниакальную стадию биполярного расстройства, могут быть активнее в движении.

2. Bluetooth-радиоприемники в смартфонах могут быть использованы для мониторинга места положения людей.

3. Данные собранные через Wi-Fi сети могут показать избегают ли пациенты, склонные к алкоголизму, бары и посещают ли встречи группы поддержки.

4. Частота телефонных звонков и текстовых сообщений показывает насколько человек общителен и указывает на его психологические изменения. Изменения в журнале входящих и исходящих текстовых сообщений и

телефонных звонков могут быть использованы для отслеживания депрессии также как и мании в биполярном расстройстве.

ГЛАВА 3

ПОСТРОЕНИЕ КЛАССИФИКАТОРА С ИСПОЛЬЗОВАНИЕМ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Задание состоит из нескольких пунктов:

1. Неформальное описание. Задача состоит в том, чтобы классифицировать психологическое состояние пациента на основе частоты звонков, сообщений и информации о передвижении. Сам процесс должен быть именно автоматическим: классификация происходит без помощи специально обученных экспертов, которые могут определить, попадает тот или иной пациент в определенный диагноз, никакого подбора правил вручную.

2. Неформальное описание. Предполагается наличие коллекции $S = \{s_1 \dots s_{|S|}\}$ заранее классифицированных пациентов, т.е. с известным значением целевой функции, конечное множество пациентов $P = \{p_1 \dots p_{|P|}\}$ и неизвестная целевая функция Φ , которая для каждой пары <пациент, диагноз> определяет, соответствуют ли они друг другу $\Phi: S \times P \rightarrow 0,1$. Задача состоит в том, чтобы найти максимально близкую к функции Φ функцию Φ' . Функцию Φ' называют классификатором.

3.1 Нейронные сети

Нейронные сети – это одно из направлений исследований в области искусственного интеллекта, основанное на попытках воспроизвести нервную систему человека. А именно: способность нервной системы обучаться и исправлять ошибки, что должно позволить смоделировать, хотя и достаточно грубо, работу человеческого мозга.

Биологическая модель нейрона.

Нейронная сеть или нервная система человека – это сложная сеть структур человека, обеспечивающая взаимосвязанное поведение всех систем организма.

Биологический нейрон – это специальная клетка, которая структурно состоит из ядра, тела клетки и отростков. Одной из ключевых задач нейрона является передача электрохимического импульса по всей нейронной сети через доступные связи с другими нейронами. Притом, каждая связь характеризуется некоторой величиной, называемой силой синаптической связи. Эта величина определяет, что произойдет с электрохимическим импульсом при передаче его

другому нейрону: либо он усилится, либо он ослабится, либо останется неизменным.

Биологическая нейронная сеть обладает высокой степенью связности: на один нейрон может приходиться несколько тысяч связей с другими нейронами. Но, это приблизительное значение и в каждом конкретном случае оно разное. Передача импульсов от одного нейрона к другому порождает определенное возбуждение всей нейронной сети. Величина этого возбуждения определяет реакцию нейронной сети на какие-то входные сигналы. Например, встреча человека со старым знакомым может привести к сильному возбуждению нейронной сети, если с этим знакомым связаны какие-то яркие и приятные жизненные воспоминания. В свою очередь сильное возбуждение нейронной сети может привести к учащению сердцебиения, более частому морганию глаз и к другим реакциям. Встреча же с незнакомым человеком для нейронной сети пройдет практически незаметной, а значит и не вызовет каких-либо сильных реакций [9-10].

На рисунке 3.1 представлена сильно упрощенную модель биологической нейронной сети.

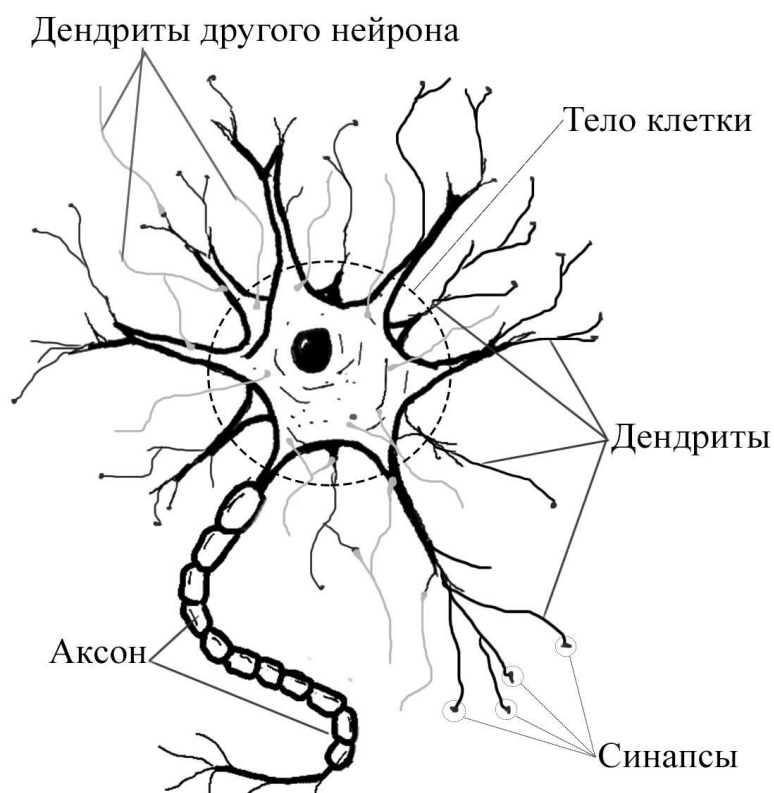


Рисунок 3.1 – Модель биологической нейронной сети

Каждый нейрон состоит из тела клетки, которое содержит ядро. От тела клетки ответвляется множество коротких волокон, называемых дендритами. Длинные дендриты называются аксонами. Аксоны растягиваются на большие расстояния, намного превышающее то, что показано в масштабе этого рисунка. Обычно аксоны имеют длину 1 см (что превышает в 100 раз диаметр тела клетки), но могут достигать и 1 метра [11].

Нейронные сети в искусственном интеллекте – это упрощенные модели биологических нейронных сетей.

На этом сходство заканчивается. Структура человеческого мозга гораздо более сложная, чем описанная выше, и поэтому воспроизвести ее хотя бы более-менее точно не представляется возможным.

У нейронных сетей много важных свойств, но ключевое из них – это способность к обучению. Обучение нейронной сети в первую очередь заключается в изменении «силы» синаптических связей между нейронами.

Математическая модель нейрона.

Нейрон представляет собой единицу обработки информации в нейронной сети. На рисунке 3.2 показана модель нейрона, лежащего в основе искусственных нейронных сетей.

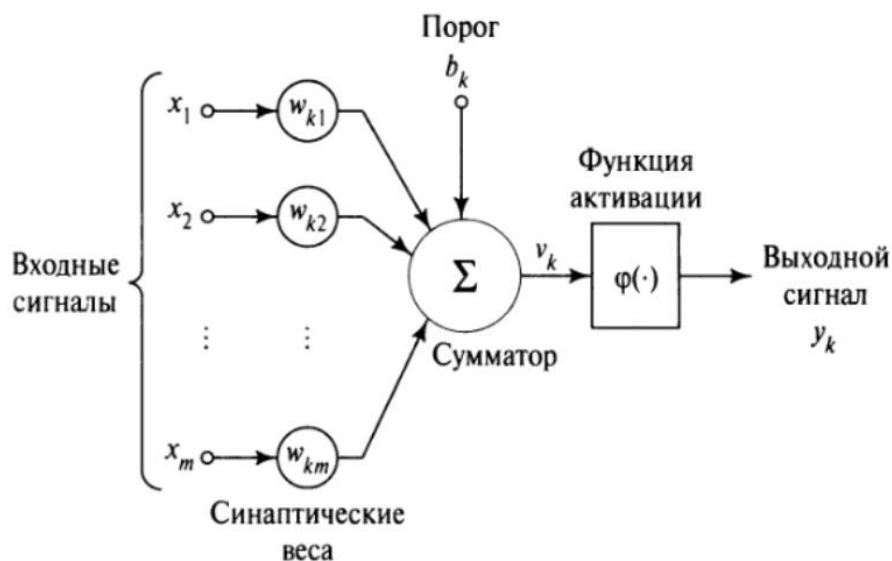


Рисунок 3.2 – Нелинейная модель нейрона

В этой модели можно выделить три основных элемента.

1. Набор синапсов или связей, каждый из которых характеризуется своим весом. В частности, сигнал x_i на входе синапса j , связанного с нейроном k , умножается на вес w_{kj} .

2. Сумматор складывает входные сигналы, взвешенные относительно соответствующих синапсов нейрона.

3. Функция активации ограничивает амплитуду выходного сигнала нейрона [12].

Пороговый элемент, который обозначен символом b_k на рисунке 1.1, отражает увеличение или уменьшение входного сигнала, подаваемого на функцию активации.

В математическом представлении функционирование нейрона k можно описать следующей парой уравнений

$$v_k = \sum_{i=0}^n w_{ki} * x_i, \quad (3.1)$$

$$y_k = \varphi(v_k + b_k), \quad (3.2)$$

где x_1, x_2, \dots, x_n – входные сигналы;
 $w_{k1}, w_{k2}, \dots, w_{km}$ – синоптические веса нейрона k ;
 v_k – линейная комбинация входных сигналов;
 b_k – порог;
 φ – функция активации;
 y_k – выходной сигнал.

Функция активации представленная как $\varphi(u)$, определяет выходной сигнал нейрона в зависимости от значения сумматора v . На рисунке 3.3 изображена сигмоидальная функция активации.

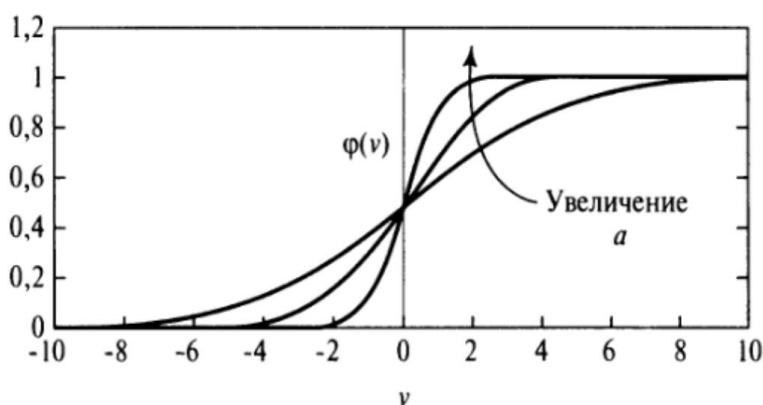


Рисунок 3.3 – Сигмоидальная функция для различных параметров a

Примером сигмоидальной функции может служить логистическая функция, задаваемая следующим выражением:

$$\varphi(u) = \frac{1}{1 + \exp(-av)}, \quad (3.3)$$

где a – параметр наклона сигмоидальной функции.

В пределе, когда параметр наклона сигмоидальной функции достигает бесконечности, сигмоидальная функция вырождается в пороговую [17].

Многослойная нейронная сеть прямого распространения представлена на рисунке 3.4.

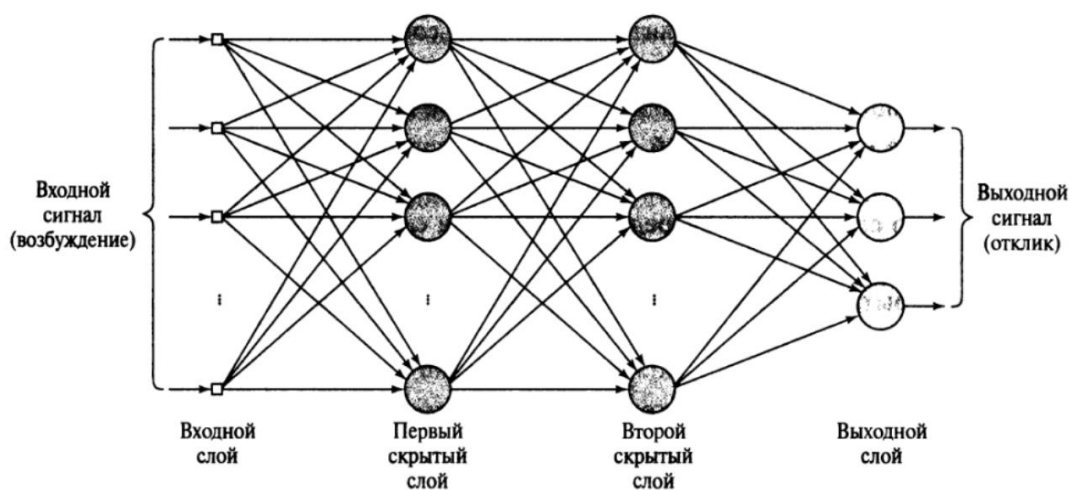


Рисунок 3.4 – Структурный граф многослойного персептрона

Многослойный персептрон характеризуется наличием одного или нескольких скрытых слоев, узлы которых называются скрытыми нейронами, или скрытыми элементами. Функция последних заключается в посредничестве между внешним входным сигналом и выходом нейронной сети. Добавляя один или несколько скрытых слоев, мы можем выделить статистики высокого порядка [13].

Многослойный персептрон имеет три отличительных признака:

- каждый нейрон сети имеет нелинейную функцию активации;
- сеть содержит один или несколько слоев скрытых нейронов, не являющихся частью входа или выхода сети;
- сеть обладает высокой степенью связанности, реализуемых по средствам синоптических соединений.

Для этого типа сети выделяют два типа сигналов (рисунок 3.5).



Рисунок 3.5 – Направление двух основных потоков сигнала для многослойного персептрона

1. Функциональный сигнал. Это входной сигнал, поступающий в сеть и передаваемый вперед от нейрона к нейрону по всей сети. Такой сигнал достигает конца сети в виде выходного сигнала.

2. Сигнал ошибки. Сигнал ошибки берет свое начало на выходе сети и распространяется в обратном направлении от слоя к слою [14].

Сети с прямой связью являются универсальным средством аппроксимации функций, что позволяет их использовать в решении задач классификации. Как правило, нейронные сети оказываются наиболее эффективным способом классификации, потому что генерируют фактически большое число регрессионных моделей, которые используются в решении задач классификации статистическими методами.

К сожалению, в применении нейронных сетей в практических задачах возникает ряд проблем. Во-первых, заранее не известно, какой сложности (размера) может потребоваться сеть для достаточно точной реализации отображения. Эта сложность может оказаться чрезмерно высокой, что потребует сложной архитектуры сетей. Было доказано, что простейшие однослойные нейронные сети способны решать только линейно разделимые задачи. Это ограничение преодолимо при использовании многослойных нейронных сетей. В общем виде можно сказать, что в сети с одним скрытым слоем вектор, соответствующий входному образцу, преобразуется скрытым слоем в некоторое новое пространство, которое может иметь другую размерность, а затем гиперплоскости, соответствующие нейронам выходного слоя, разделяют его на классы. Таким образом сеть распознает не только характеристики исходных данных, но и «характеристики» характеристик, сформированные скрытым слоем.

Процесс обучения.

Определение процесса обучения предполагает следующую последовательность событий:

- в нейронную сеть поступают сигналы из внешней среды;
- в результате этого изменяются свободные параметры нейронной сети;
- после изменения внутренней структуры нейронная сеть отвечает на возбуждение уже иным образом.

Выделяют две основных парадигмы обучения нейронных сетей.

1. Обучение с учителем, то есть процесс обучения, при котором обучение происходит путем предоставления сети последовательности обучающих примеров с правильными откликами.

2. Обучение без учителя. Парадигма обучения без учителя самим названием подчеркивает отсутствие руководителя, контролирующего процесс настройки весовых коэффициентов. При использовании такого подхода не существует маркированных примеров, по которым проводится обучение сети.

Концептуально участие учителя можно рассматривать как наличие знаний об окружающей среде, представленных в виде пар вход-выход. При этом сама среда неизвестна обучаемой нейронной сети.

На основе встроенных знаний учитель может сформировать и передать обучаемой нейронной сети желаемый отклик, соответствующий данному входному вектору. Этот желаемый результат представляет собой оптимальные действия, которые должна выполнить нейронная сеть. Параметры сети корректируются с учетом обучающего вектора и сигнала ошибки. Сигнал ошибки – это разность между желаемым сигналом и текущим откликом нейронной сети. Корректировка параметров выполняется пошагово с целью имитации нейронной сетью поведения учителя. После окончания обучения учителя можно отключить и позволить нейронной сети работать со средой самостоятельно.

Для обучения без учителя можно воспользоваться правилом конкурентного обучения. Например, можно использовать нейронную сеть, состоящую из двух слоев – входного и выходного. Входной слой получает доступные данные. Выходной слой состоит из нейронов, конкурирующих друг с другом за право отклика на признаки, содержащиеся во входных данных.

В простейшем случае нейронная сеть действует по принципу «победитель получает все». При такой стратегии нейрон с наибольшим суммарным входным сигналом «побеждает» в соревновании и переходит в активное состояние. При этом все остальные нейроны отключаются [15].

Рекуррентные нейронные сети.

Рекуррентная нейронная сеть отличается от сети прямого распространения наличием, по крайней мере, одной обратной связи. Наличие

обратных связей в сети, показанной на рисунке 3.6, позволяет использовать информацию о предыдущих событиях для анализа последующих.

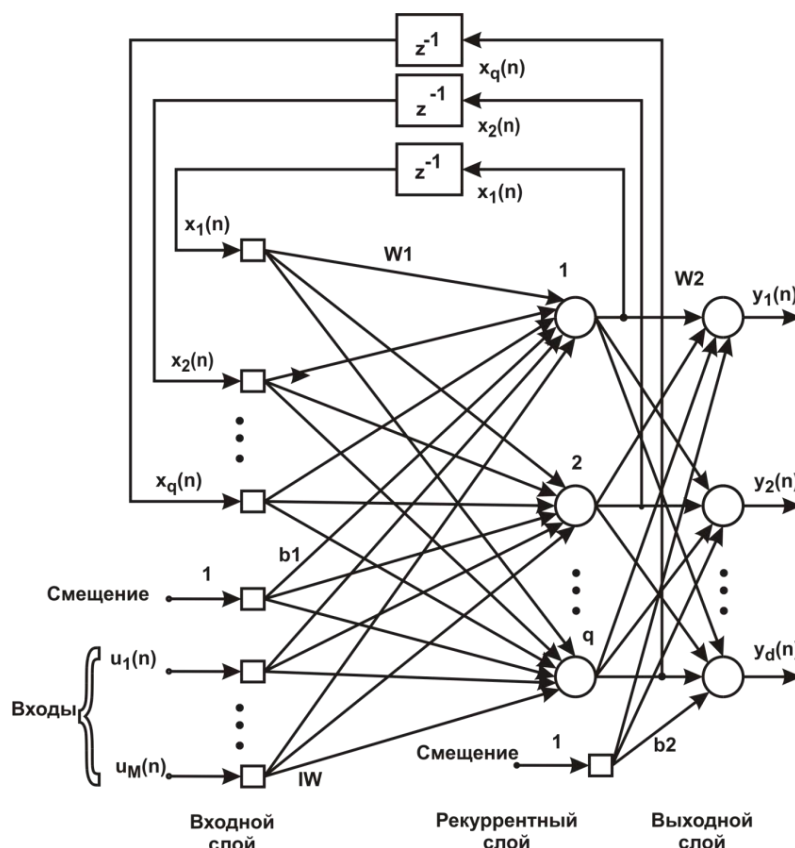


Рисунок 3.6 – Рекуррентная сеть со скрытыми нейронами

Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. Поэтому рекуррентные нейронные сети применимы в таких задачах, где нечто целостное разбито на сегменты, например: распознавание речи или эмоций.

Трудность рекуррентной сети заключается в том, что если учитывать каждый шаг времени, то становится необходимым для каждого шага времени создавать свой слой нейронов, что вызывает серьёзные вычислительные сложности. Кроме того многослойные реализации оказываются вычислительно неустойчивыми, так как в них как правило исчезают или зашкаливают веса. Если ограничить расчёт фиксированным временным окном, то полученные модели не будут отражать долгосрочных трендов. Различные подходы пытаются усовершенствовать модель исторической памяти и механизм запоминания и забывания [16-18].

Даже в улыбке – одной из самых простых эмоций – есть несколько моментов: от нейтрального выражения лица до того момента, когда будет полная улыбка. Они идут друг за другом последовательно. Чтоб это хорошо

понимать, нужно уметь наблюдать за тем, как это происходит, передавать то, что было на предыдущем кадре в следующий шаг работы системы [19].

3.2 Алгоритм построения классификатора

Алгоритм построения классификатора на основе нейронных сетей (см. рисунок 3.7) состоит из пяти этапов:

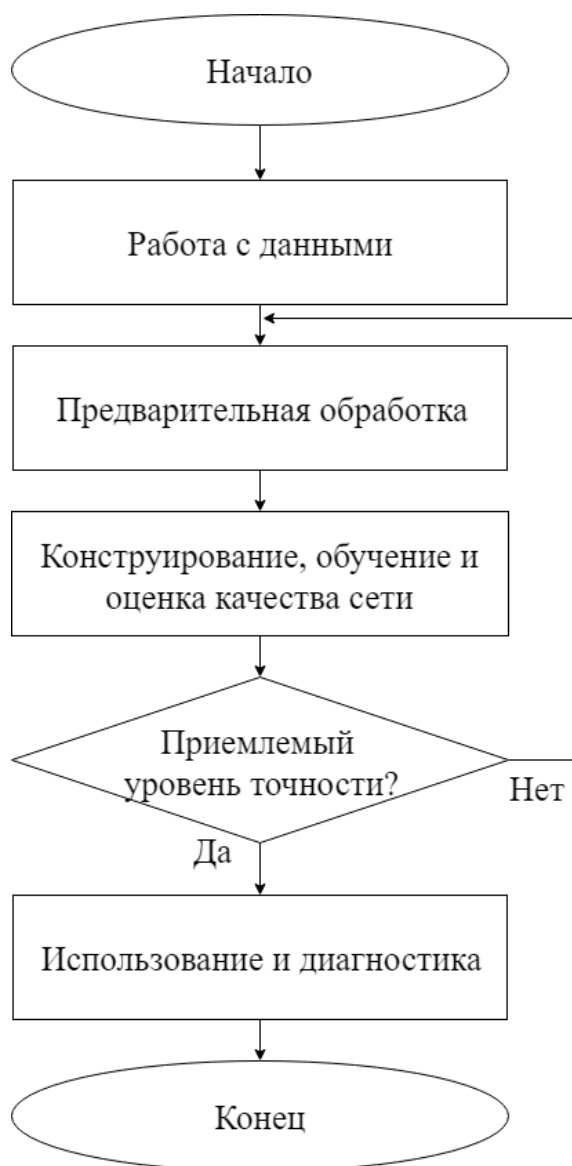


Рисунок 3.7 – Алгоритм построения классификатора

1. Работа с данными:

– составить базу данных из примеров, характерных для данной задачи;

- разбить всю совокупность данных на два множества: обучающее и тестовое (возможно разбиение на 3 множества: обучающее, тестовое и подтверждающее).

2. Предварительная обработка:

- выбрать систему признаков, характерных для данной задачи, и преобразовать данные соответствующим образом для подачи на вход сети (нормировка, стандартизация и т.д.). В результате желательно получить линейно отделяемое пространство множества образцов;

- выбрать систему кодирования выходных значений (классическое кодирование, 2 на 2 кодирование и т.д.).

3. Конструирование, обучение и оценка качества сети:

- выбрать топологию сети: количество слоев, число нейронов в слоях и т.д.

- выбрать функцию активации нейронов (например «сигмоида»);

- выбрать алгоритм обучения сети;

- оценить качество работы сети на основе подтверждающего множества или другому критерию, оптимизировать архитектуру (уменьшение весов, прореживание пространства признаков);

- остановится на варианте сети, который обеспечивает наилучшую способность к обобщению и оценить качество работы по тестовому множеству.

4. Использование и диагностика:

- выяснить степень влияния различных факторов на принимаемое решение (эвристический подход);

- убедиться, что сеть дает требуемую точность классификации (число неправильно распознанных примеров мало);

- при необходимости вернуться на этап 2, изменив способ представления образцов или изменив базу данных;

- практически использовать сеть для решения задачи.

Для того, чтобы построить качественный классификатор, необходимо иметь качественные данные. Никакой из методов построения классификаторов, основанный на нейронных сетях или статистический, никогда не даст классификатор нужного качества, если имеющийся набор примеров не будет достаточно полным и представительным для той задачи, с которой придется работать системе [20-22].

Подготовка исходных данных.

Для построения классификатора необходимо определить, какие параметры влияют на принятие решения о том, к какому классу принадлежит образец. При этом могут возникнуть две проблемы. Во-первых, если количество параметров мало, то может возникнуть ситуация, при которой один и тот же набор исходных данных соответствует примерам, находящимся в

разных классах. Тогда невозможно обучить нейронную сеть, и система не будет корректно работать (невозможно найти минимум, который соответствует такому набору исходных данных). Исходные данные обязательно должны быть непротиворечивы. Для решения этой проблемы необходимо увеличить размерность пространства признаков (количество компонент входного вектора, соответствующего образцу). Но при увеличении размерности пространства признаков может возникнуть ситуация, когда число примеров может стать недостаточным для обучения сети, и она вместо обобщения просто запомнит примеры из обучающей выборки и не сможет корректно функционировать. Таким образом, при определении признаков необходимо найти компромисс с их количеством.

Далее необходимо определить способ представления входных данных для нейронной сети, т.е. определить способ нормирования. Нормировка необходима, поскольку нейронные сети работают с данными, представленными числами в диапазоне $0 \dots 1$, а исходные данные могут иметь произвольный диапазон или вообще быть нечисловыми данными. При этом возможны различные способы, начиная от простого линейного преобразования в требуемый диапазон и заканчивая многомерным анализом параметров и нелинейной нормировкой в зависимости от влияния параметров друг на друга.

Кодирование выходных значений.

Задача классификации при наличии двух классов может быть решена на сети с одним нейроном в выходном слое, который может принимать одно из двух значений 0 или 1, в зависимости от того, к какому классу принадлежит образец. При наличии нескольких классов возникает проблема, связанная с представлением этих данных для выхода сети. Наиболее простым способом представления выходных данных в таком случае является вектор, компоненты которого соответствуют различным номерам классов. При этом i -я компонента вектора соответствует i -му классу. Все остальные компоненты при этом устанавливаются в 0. Тогда, например, второму классу будет соответствовать 1 на 2 выходе сети и 0 на остальных. При интерпретации результата обычно считается, что номер класса определяется номером выхода сети, на котором появилось максимальное значение. Например, если в сети с тремя выходами мы имеем вектор выходных значений (0.2, 0.6, 0.4), то мы видим, что максимальное значение имеет вторая компонента вектора, значит класс, к которому относится этот пример, – 2. При таком способе кодирования иногда вводится также понятие уверенности сети в том, что пример относится к этому классу. Наиболее простой способ определения уверенности заключается в определении разности между максимальным значением выхода и значением другого выхода, которое является ближайшим к максимальному. Соответственно чем выше

уверенность, тем больше вероятность того, что сеть дала правильный ответ. Этот метод кодирования является самым простым, но не всегда самым оптимальным способом представления данных.

Известны и другие способы. Например, выходной вектор представляет собой номер кластера, записанный в двоичной форме. Тогда при наличии 8 классов нам потребуется вектор из 3 элементов, и, скажем, 3 классу будет соответствовать вектор 011. Но при этом в случае получения неверного значения на одном из выходов мы можем получить неверную классификацию (неверный номер кластера), поэтому имеет смысл увеличить расстояние между двумя кластерами за счет использования кодирования выхода по коду Хемминга, который повысит надежность классификации.

Другой подход состоит в разбиении задачи с k классами на $k * (k - 1) / 2$ подзадач с двумя классами (2 на 2 кодирование) каждая. Под подзадачей в данном случае понимается то, что сеть определяет наличие одной из компонент вектора. Т.е. исходный вектор разбивается на группы по два компонента в каждой таким образом, чтобы в них вошли все возможные комбинации компонент выходного вектора. Число этих групп можно определить как количество неупорядоченных выборок по два из исходных компонент. Из комбинаторики

$$A_k^n = \frac{k!}{n!(k-n)!} = \frac{k!}{2!(k-2)!} = \frac{k(k-1)}{2} \quad (3.4)$$

где k – количество классов;
 n – количество классов в подзадаче;
 A – количество выходов (подзадач).

Тогда, например, для задачи с четырьмя классами мы имеем 6 выходов (подзадач), распределение которых можно увидеть в таблице 3.1.

Таблица 3.1 – Распределение выходов нейронной сети по подзадачам

N подзадачи (выхода)	Компоненты выхода
1	1 – 2
2	1 – 3
3	1 – 4
4	2 – 3
5	2 – 4
6	3 – 4

Где 1 на выходе говорит о наличии одной из компонент. Тогда мы можем перейти к номеру класса по результату расчета сетью следующим образом:

определяем, какие комбинации получили единичное (точнее близкое к единице) значение выхода (т.е. какие подзадачи у нас активировались), и считаем, что номер класса будет тот, который вошел в наибольшее количество активированных подзадач (см. таблицу 3.2).

Таблица 3.2 – Распределение классов по выходам нейронной сети

N класса	Активные выходы
1	1, 2, 3
2	1, 4, 5
3	2, 4, 6
4	3, 4, 6

Данное кодирование во многих задачах дает лучший результат, чем классический способ кодирования [23-24].

Объем сети.

Правильный выбор объема сети имеет большое значение. Построить небольшую и качественную модель часто бывает просто невозможно, а большая модель будет просто запоминать примеры из обучающей выборки и не производить аппроксимацию, что, естественно, приведет к некорректной работе классификатора. Существуют два основных подхода к построению сети – конструктивный и деструктивный. При первом из них вначале берется сеть минимального размера, и постепенно увеличивают ее до достижения требуемой точности. При этом на каждом шаге ее заново обучают. Также существует так называемый метод каскадной корреляции, при котором после окончания эпохи происходит корректировка архитектуры сети с целью минимизации ошибки. При деструктивном подходе вначале берется сеть завышенного объема, и затем из нее удаляются узлы и связи, мало влияющие на решение. При этом полезно помнить следующее правило: число примеров в обучающем множестве должно быть больше числа настраиваемых весов. Иначе вместо обобщения сеть просто запомнит данные и утратит способность к классификации – результат будет неопределен для примеров, которые не вошли в обучающую выборку [25].

ГЛАВА 4

СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА

4.1 Мобильное приложение

Основной целью мобильного приложения является сбор поведенческой информации пациента, такой как история звонков, сообщений и перемещений. Данная часть системы не нуждается в наличие графического интерфейса и должна работать в фоне. Поэтому было принято решение создать Android службу (сервис).

Службы (Сервисы) в Android работают как фоновые процессы и представлены классом `android.app.Service`. Они не имеют пользовательского интерфейса и нужны в тех случаях, когда не требуется вмешательства пользователя. Сервисы работают в фоновом режиме, выполняя сетевые запросы к веб-серверу, обрабатывая информацию, запуская уведомления и т.д. Служба может быть запущена и будет продолжать работать до тех пор, пока кто-нибудь не остановит её или пока она не остановит себя сама. Сервисы предназначены для длительного существования, в отличие от активностей. Они могут работать, постоянно перезапускаясь, выполняя постоянные задачи или выполняя задачи, требующие много времени [29].

Android даёт службам более высокий приоритет, чем бездействующим активностям, поэтому вероятность того, что они будут завершены из-за нехватки ресурсов, заметно уменьшается. Если система должна преждевременно завершить работу запущенного сервиса, он может быть настроен таким образом, чтобы запускаться повторно, как только станет доступно достаточное количество ресурсов. В крайних случаях приоритет сервиса может быть повышен до уровня активности, работающей на переднем плане.

Запущенные сервисы всегда имеют больший приоритет, чем бездействующие или невидимые активности, поэтому менее вероятно, что их работа завершится преждевременно при распределении ресурсов. Единственная причина, почему Android может досрочно остановить Сервис, – выделение дополнительных ресурсов для компонентов, работающих на переднем плане (как правило, для активностей). Если такое случится, сервис автоматически перезапустится, когда будет достаточно доступных ресурсов.

Чтобы определить службу, необходимо создать новый класс, расширяющий базовый класс `Service`:

```
public class BehavioralService extends Service {
    public BehavioralService () {...}
    ...
}
```

Также необходимо зарегистрировать сервис в манифесте приложения в секции application:

```
<service
    android:name=".BehavioralService"
    android:enabled="true"
    android:exported="true" >
</service>
```

На рисунке 4.1 можно увидеть жизненный цикл Android службы:

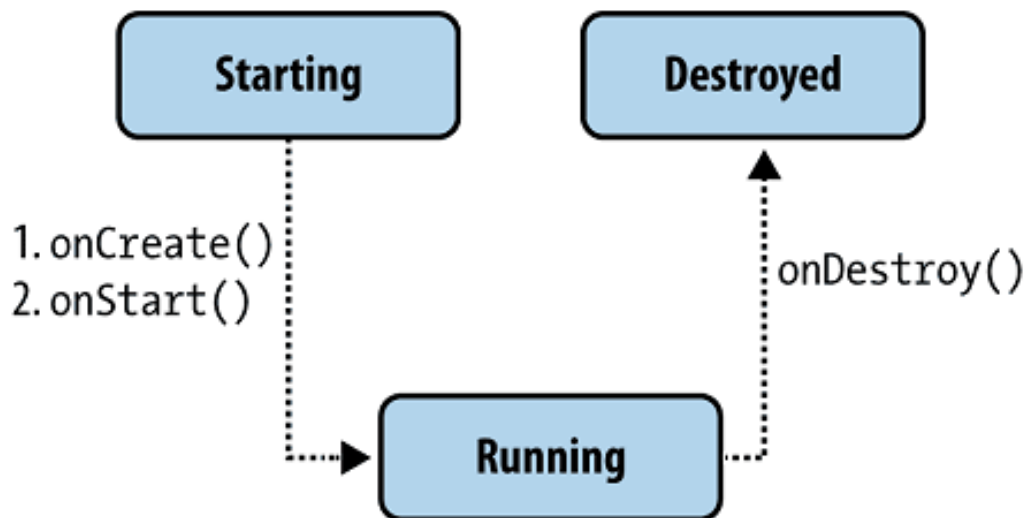


Рисунок 4.1 – Жизненный цикл Android службы

Жизнь службы начинается с вызова метода `onCreate`, внутри которого производим начальную инициализацию. Первое, что необходимо сделать, это получить доступ к идентификатору/имени пользователя учетной записи Google через Android:

```
public String getUsername() {
    AccountManager manager = AccountManager.get(this);
    Account account = manager.getAccountByType("com.google");
    String email = account.name.get(0);
}
```

```

        return email.split("@");
    }

```

Активная жизнь службы начинается с вызова метода `onStart`. Внутри данного метода производим запуск процесса, который будет периодически отправлять поведенческую информацию пациента на сервер:

```

Boolean stop = false;
new android.os.Handler().postDelayed(
    new Runnable() {
        public void run() {
            if (!stop)
                behaviourData.sync();
        }
    }, 5000);

```

При остановке работы службы вызывается метод `onDestroy`, где выполняется освобождение всех ресурсов. В теле метода устанавливаем флаг `stop` в значение `true`. В результате чего будет остановлена синхронизации данных с сервером.

```

@Override
public void onDestroy() {
    stop = true;
}

```

Для доступа к истории звонков и сообщений был использован поставщик контента `Android`. Поставщик контента предоставляет данные внешним приложениям в виде одной или нескольких таблиц, аналогичных таблицам в реляционной базе данных. Строка представляет собой экземпляр некоторого типа собираемых поставщиком данных, а каждый столбец в этой строке – это отдельный элемент данных, собранных для экземпляра.

В листинге ниже, используя массив из ключей, указываем какую информацию о звонках нам необходимо получить:

```

String[] projection = new String[]{
    CallLog.Calls._ID,
    CallLog.Calls.DATE,
    CallLog.Calls.NUMBER,
    CallLog.Calls.CACHED_NAME,

```

```
        CallLog.Calls.DURATION,  
    };
```

Для доступа к данным из поставщика контента используется клиентский объект `ContentResolver`:

```
Cursor cursor = ContentResolver.query(  
    CallLog.Calls.CONTENT_URI,  
    projection,  
);
```

Таким же образом запрашиваем историю сообщений:

```
String[] projection = new String[]{  
    SmsLog.Sms._ID,  
    SmsLog.Sms.DATE,  
    SmsLog.Sms.NUMBER,  
    SmsLog.Sms.CACHED_NAME,  
};
```

```
Cursor cursor = ContentResolver.query(  
    SmsLog.Sms.CONTENT_URI,  
    projection,  
);
```

Также необходимо запросить разрешение на чтение истории звонков и сообщение в манифесте приложения:

```
<uses-permission android:name="android.permission.READ_CALLS" />  
<uses-permission android:name="android.permission.READ_SMS" />
```

Смартфоны и планшеты на базе Android постоянно фиксируют местоположение пользователя и хранят эти данные. Координаты они определяют несколькими способами: через спутники GPS, башни сотовой связи или точки Wi-Fi. История хранится с самого первого дня, когда у пользователя появилось устройство на Android, и не очищается автоматически.

Android предоставляет приложениям доступ к службам определения местоположения с помощью классов в пакете `android.location`. Центральным компонентом структуры местоположения является системная служба

LocationManager, которая предоставляет API-интерфейсы для определения местоположения и ориентации устройства (если доступно).

В листинге ниже мы запрашиваем экземпляр LocationManager из системы, используя который, мы получаем последнее зарегистрированное телефоном местоположение пользователя:

```
mLastLocation = getSystemService(Context.LOCATION_SERVICE)
    .getLastLocation(mGoogleApiClient);
store.saveLocation(mLastLocation)
```

Google Maps не предоставляет историю перемещений пользователя, так как это нарушает правила конфиденциальности. Поэтому нам доступно только последнее зарегистрированное телефоном местоположение.

Когда пользователь первый раз открывает приложение, он видит экран авторизации (см. рисунок 4.2). Пользователю предлагается пройти авторизацию с использованием Google аккаунта.

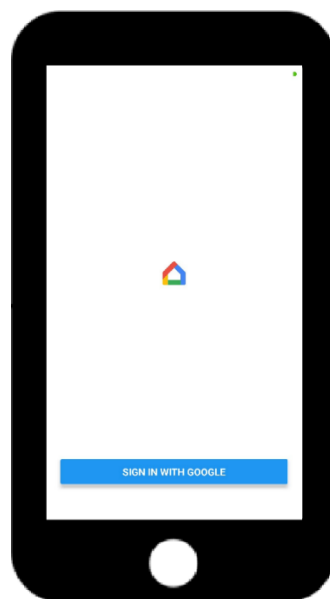


Рисунок 4.2 – Экран авторизации

При нажатии на кнопку «SIGN IN WITH GOOGLE», приложение отправляет запрос на google-сервис, чтобы получить токен для доступа к информации пользователя. Далее происходит перенаправление пользователя на страницу авторизации google. Если пользователь имеет несколько аккаунтов, google предлагает выбрать один из имеющихся (см. рисунок 3.5).

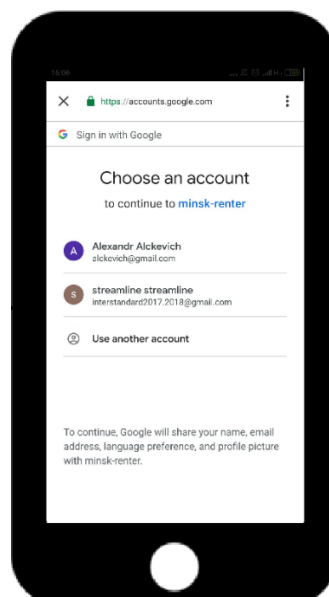


Рисунок 4.3 – Страница авторизации Google

Когда пользователь успешно проходит авторизацию Google, наше приложение получает токен доступа к его информации. Используя токен, мы можем запросить адрес электронного почтового ящика, чтобы зарегистрировать пользователя в системе.

После того как пользователь проходит успешно авторизацию, запускается Android служба, которая работает в фоне.

4.2 Сервер

Для реализации сервера была использована технология ASP.Net Web API 2 на платформе .NET Framework 4.5. Для доступа к объектам реляционной базы данных в зависимости проекта был добавлен Entity Framework версии 6 с помощью NuGet package manager.

Для создания API-интерфейсов, которые предоставляют службы, и данные было реализовано несколько объектов-контроллеров, которые обрабатывают HTTP-запросы. На рисунке 4.4 можно увидеть контроллеры, которые определяют контракт доступа к данным сервера.

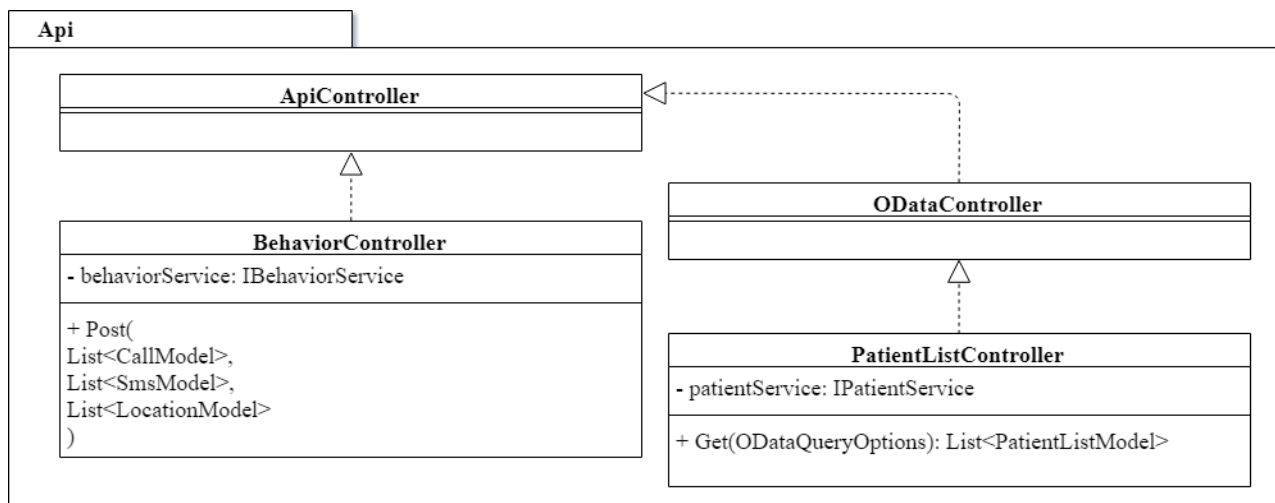


Рисунок 4.4 – Серверные контроллеры

Для представления данных в приложении были созданы следующие объекты-модели:

- a) CallModel – представляет данные о звонке пациента;
- b) SmslModel – представляет данные о сообщении пациента;
- c) LocationModel – представляет данные о местоположении пациента;
- b) PatientListModel – модель представления пациента, используется для вывода списка пациентов в веб-приложении для доктора.

Объекты представления сериализуются в формат JSON, а затем записываются в тело HTTP запроса/ответа.

BehaviorController перехватывает HTTP запросы со стороны клиента, проверяет пришедшие данные и, если данные валидны, перенаправляет обработку запроса объекту реализующему интерфейс IBehaviorService. На рисунке 4.5 можно увидеть серверные сервисы.

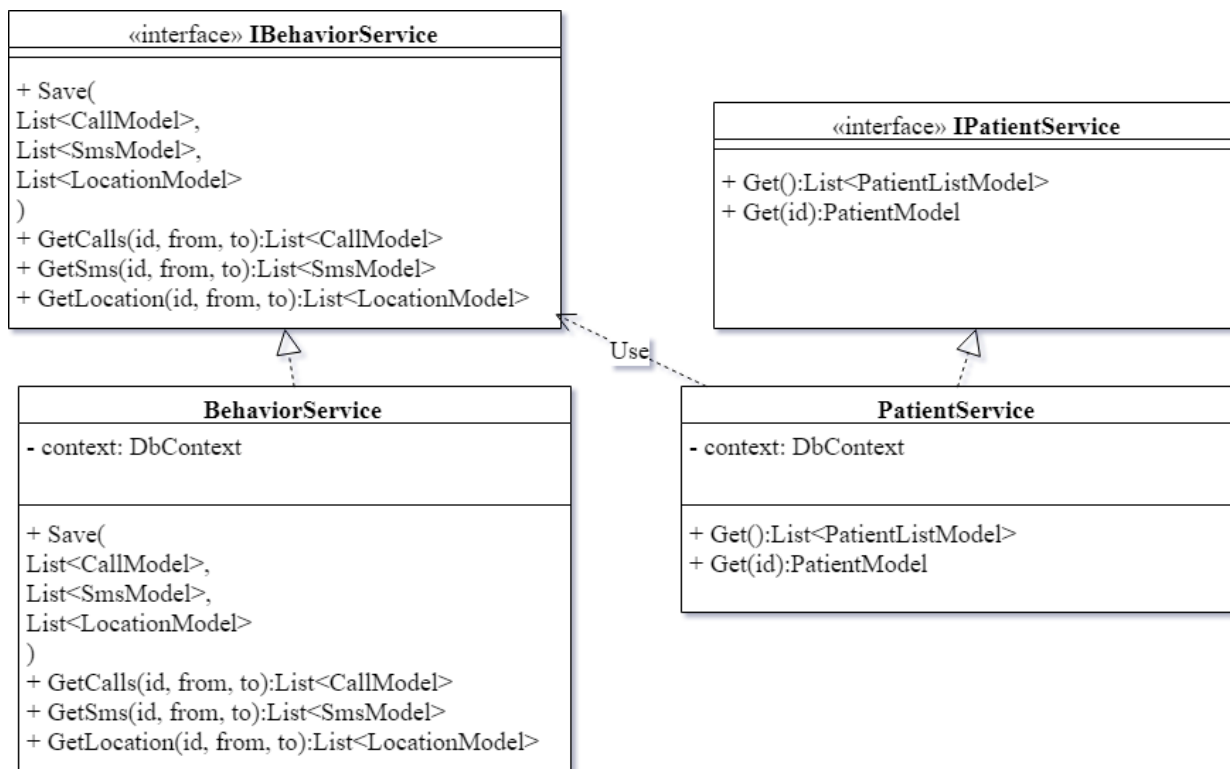


Рисунок 4.5 – Серверные сервисы

Для сохранения поведенческой информации пациента необходимо по маршруту `serverUrl/api/behavior` отправить HTTP запрос с POST методом, в теле которого должна быть размещена вся необходимая информация в формате JSON.

`PatientListController` наследуется от `ApiController`, что позволяет, при обращении к API контроллера использовать возможности протокола OData (Open Data Protocol). OData расширяет RESTful API стандарт и позволяет клиентам формировать запрос, используя параметры фильтрации, разбиения на страницы, сортировки данных.

Для реализации аутентификации между клиентским и серверным приложением был использован подход, основанный на токене.

Когда пользователь успешно выполняет вход в систему со своими учетными данными, сервер использует секретный ключ для шифрования необходимых пользовательских данных и генерирует, на основе этих данных, токен. Этот токен возвращается клиенту и должен быть сохранен в локальном хранилище браузера или в cookies.

Всякий раз, когда пользователь захочет получить доступ к защищенному ресурсу, он должен отправить токен в заголовке авторизации с использованием схемы Bearer (`Authorization: Bearer <token>`). Затем сервер использует

секретный ключ для дешифрования и проверки токена, и если он действителен, пользователь получает доступ к защищенным ресурсам.

Аутентификация на основе токенов является механизмом аутентификации без состояния, поскольку пользовательское состояние никогда не сохраняется в памяти сервера. Такой подход уменьшает необходимость многократного запроса к базе данных, поскольку токен содержит всю необходимую информацию. Технически, после того, как токен подписан, он действителен навсегда, если явно не установлен срок действия.

Так как модули должны принимать запросы, поступающие из любого источника, был использован механизм CORS (Cross Origin Resource Sharing), который обеспечивает междоменное управление доступом и безопасную междоменную передачу данных.

Для включения механизма аутентификации и защиты данных каждый модуль должен содержать следующую настройку:

```
public static void ConfigureOAuth(IApplicationBuilder app, IContainer container)
{
    var oAuthServerOptions = new OAuthAuthorizationServerOptions()
    {
        AllowInsecureHttp = true,
        TokenEndpointPath = new PathString("/api/token"),
        AccessTokenExpireTimeSpan = tokenexpiration,
        Provider = container
            .Resolve<CustomAuthorizationServerProvider>(),
        RefreshTokenProvider = container.Resolve<RefreshTokenProvider>()
    };
    // Token Generation
    app.UseOAuthAuthorizationServer(oAuthServerOptions);
}
```

4.3 Вычислительный сервис

Сервис производит сложные вычисления над большими данными и сильно загружает систему на которой работает. Поэтому было принято решение запускать его на отдельной от сервера машине.

Так как новая информация о пациенте поступает в базу данных постоянно, было бы не целесообразно проводить вычисления каждый раз при

её изменении. Поэтому было принято решение запускать сервис каждые двадцать четыре часа. При запуске производится проверка данных пациента на наличие новой информации, и выполняется пересчет психологического состояния. Для запуска такого рода рекуррентной задачи была использована сторонняя библиотека Hangfire [30].

Hangfire – многопоточный и масштабируемый планировщик задач, построенный по клиент-серверной архитектуре на стеке технологий .NET (в первую очередь Task Parallel Library и Reflection), с промежуточным хранением задач в БД. Hangfire полностью функционален в бесплатной (LGPL v3) версии с открытым исходным кодом.

Как можно видеть на рисунке 4.6, процесс-клиент добавляет задачу в БД, процесс-сервер периодически опрашивает БД и выполняет задачи.

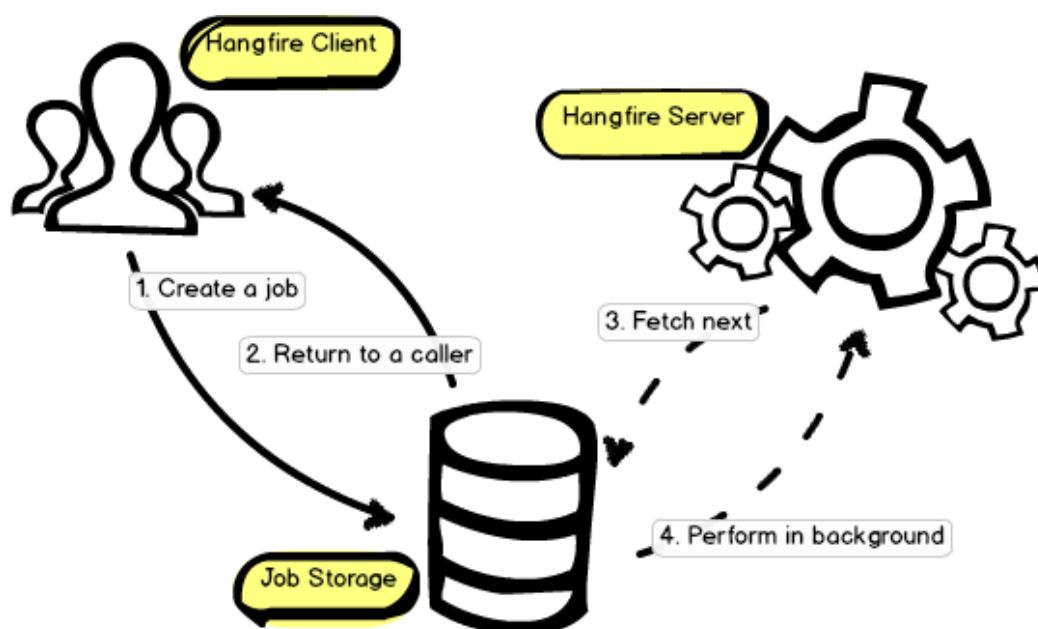


Рисунок 4.6 – Основные процессы в Hangfire

Hangfire не привязан к конкретному типу приложения .NET. Вы можете использовать его в веб-приложениях ASP.NET, веб-приложениях, отличных от ASP.NET, в консольных приложениях или службах Windows. Вот требования:

- .NET Framework 4.5;
- постоянное хранилище;
- библиотека Newtonsoft.Json версии выше 5.0.1.

С точки зрения клиента, работа с задачей происходит по принципу «добавил в очередь и забыл» на клиенте не происходит ничего, помимо сохранения задачи в БД.

Код для фонового выполнения необязательно должен находиться в одной сборке с клиентом. Схема зависимостей сборок представлена на рисунке 4.7.

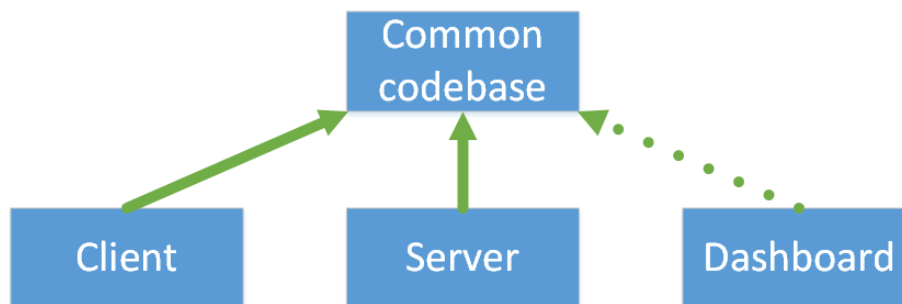


Рисунок 4.7 – Схема зависимостей сборок

Клиент и сервер должны иметь доступ к общей сборке, при этом для встроенного веб-интерфейса доступ необязателен. При необходимости, можно заменить реализацию уже сохраненной в БД задачи – путем замены сборки, на которую ссылается приложение-сервер. Это удобно для повторяемых по расписанию задач, но, работает при условии полного совпадения контракта в старой и новой сборках.

Все необходимые таблицы создаются автоматически в БД при первом запуске сервиса Hangfire-сервер (см. рисунок 4.8).

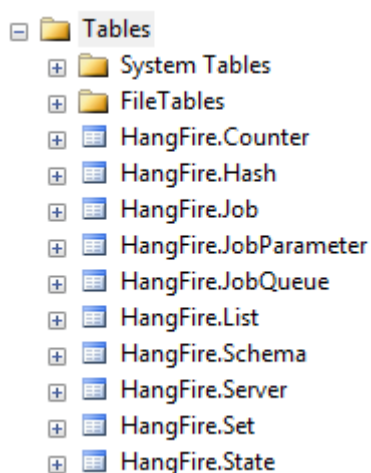


Рисунок 4.8 – Встроенное веб-приложение Hangfire

После старта сервиса Hangfire-сервер начинает читать задачи из БД и выполнять их. Hangfire имеет встроенное веб-приложение, которое позволяет управлять обработкой задач (см. рисунок 4.9).

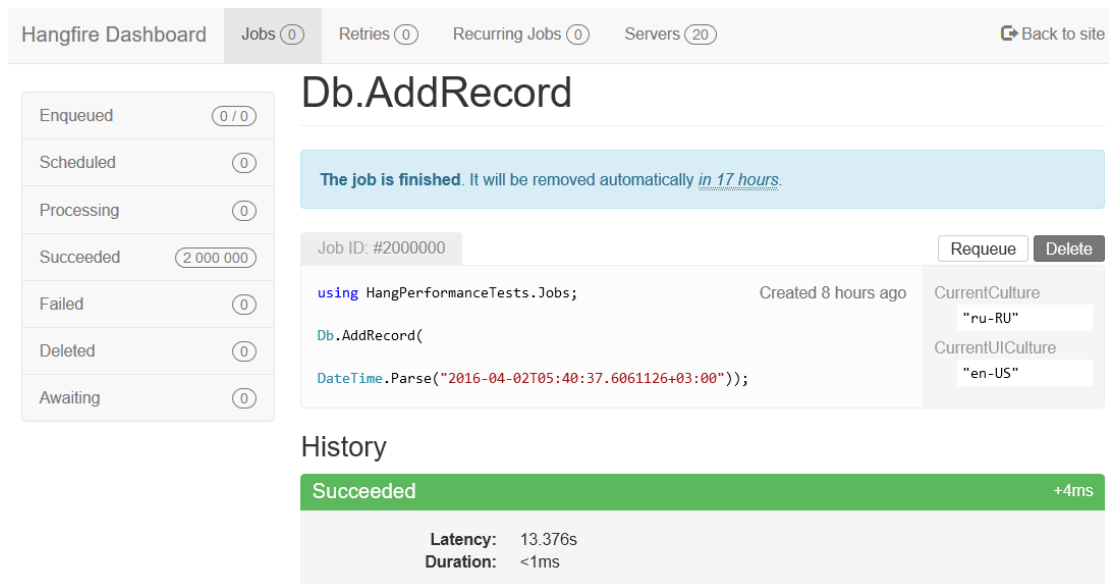


Рисунок 4.9 – Встроенное веб-приложение Hangfire

Hangfire-сервер содержит свой пул потоков, реализованный через Task Parallel Library.

4.4 Определение психологического состояния человека

Представление данных.

Представление данных для компьютерной обработки. В соответствии с заданием, необходимо было построить классификатор, который способен обучаться на основе заранее классифицированных данных. В качестве обучающей выборки было взято 222 пациента: 111 с психологическими отклонениями, остальные 111 – без. Однако поведенческие данные в первичной виде нельзя передать в нейронную сеть, поэтому первым этапом решения задачи автоматической классификации пациентов является преобразование данных к виду пригодному для алгоритмов машинного обучения.

Функция ошибки представлена в виде меры близости двух векторов. В качестве меры близости была реализована функция вычисления половины квадрата Евклидова расстояния, которая выражена следующим образом:

$$E = \sum_i e_i^2 = \sum_i (d_i - y_i)^2, \quad (4.1)$$

где d – вектор ожидаемых выходных значений нейронной сети;
 y – вектор полученных выходных значений сети;

E – мера близости (ошибка).

Производная функции 2 выглядит следующим образом:

$$\frac{dE}{dy_j} = y_j - d_i. \quad (4.2)$$

где d – вектор ожидаемых выходных значений нейронной сети;

y – вектор полученных выходных значений сети;

E – мера близости (ошибка).

Ниже представлен листинг реализации функции ошибки:

```
public class LeastSquareMethod : IProximityMeasure
{
    public double Compute(double[] d, double[] y)
    {
        double E = 0;
        for (var i = 0; i < d.Length; i++)
        {
            E += (y[i] - d[i]) * (y[i] - d[i]);
        }
        return E * 0.5;
    }

    public double ComputePartialDerivative(
        double[] d,
        double[] y,
        int Index
    )
    {
        return y[Index] - d[Index];
    }
}
```

Метод `Compute` вычисляет меру близости двух векторов (значение ошибки).

Метод `ComputePartialDerivative` вычисляет значение частной производной функции для выходных векторов по индексу переменной из y .

Функция Активации.

В качестве функции активации была использована сигмоидальная функция, реализация которой представлена в листинге ниже:

```

public double Compute(double x)
{
    double r = (1 / (1 + Math.Exp(-1 * alpha * x)));
    return r;
}

public double ComputeFirstDerivative(double x)
{
    return alpha * this.Compute(x) * (1 - this.Compute(x));
}

```

Compute – вычисляет значение сигмоидальной функции по переданному аргументу.

ComputeFirstDerivative – вычисляет значение производной сигмоидальной функции по переданному аргументу.

Нейрон.

Реализация элементарной единицы нейронной сети выглядит следующим образом:

```

public interface INeuron
{
    public List<Link> OutgoingLinks { get; set; }
    public List<Link> IncomingLinks { get; set; }

    public double dEdS { get; set; }
    public double LastNET { get; set; }
    public double NET { get; set; }
    public double OUT { get; set; }
    public double Threshold { get; set; }

    public IActivationFunction ActivationFunction { get; set; }
    public double Activate();
    public void ReceiveImpulse(double impulse);
}

```

Для обеспечения возможности распространения сигнала в обе стороны, нейрон содержит в себе ссылки как на входные (IncomingLinks), так и на выходные (OutgoingLinks) нейроны.

LastNET – сумматор нейрона, храни последнее вычисленное значение, необходимое для алгоритма обучения;

OUT – выход нейрона, храни последнее вычисленное значение;

dEdS – частная производная функции ошибки по сумматору нейрона (градиент), вычисляется в зависимости от того на каком слое находится текущий перон;

Threshold – пороговое значение, используется для увеличения или уменьшения входного сигнала, подаваемого на функцию активации;

ReceiveImpulse – наращивает значение NET на переданное значение impulse;

ActivationFunction – функция активации, которая вызывается во время активации нейрона, чтобы рассчитать выход нейрона;

Activate – используя функцию активации, вычисляет OUT по значению NET и отправляет полученное значение выходным нейронам; NET обнуляется, а его старое состояние заносится в LastNet.

Синапс.

Синапс служит связующим звеном для двух нейронов, его реализация выглядит следующим образом:

```
public class Link
{
    public double dw { get; set; }
    public double Weight { get; set; }
    public INeuron Origin { get; set; }
    public INeuron Destination { get; set; }

    public void SendImpulse(double impulse)
    {
        Destination.ReceiveImpulse(impulse * Weight);
    }
}
```

Origin – ссылка на нейрон начала связи.

Destination – ссылка на нейрон конца связи.

Weight – весовой коэффициент связи двух нейронов. Значение импульса первого нейрона умножается на весовой коэффициент связи для расчета входного значения второго нейрона.

Dw – хранит последнее изменение весового коэффициента во время обучающего процесса для вычисления момента инерции.

SendImpulse – перехватывает сигнал входного нейрона, корректирует его в соответствии с весовым коэффициентом и отправляет значение выходному нейрону.

Слой нейронов.

Алгоритм активации слоя нейронной сети представлен на рисунке 4.10.

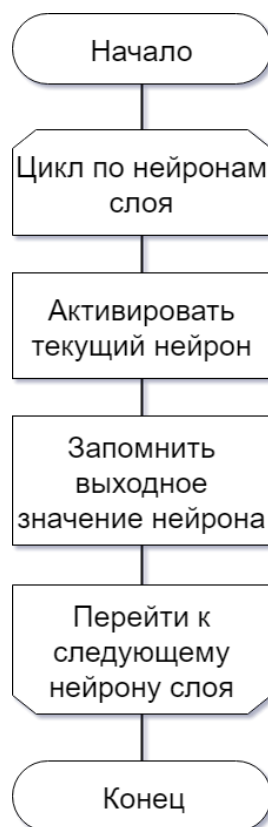


Рисунок 4.10 – Алгоритм активации слоя нейронной сети

Структурная единица нейронной сети представлена в коде следующим образом:

```
public class Layer
{
    public double[] LastOutput { get; set; }
    public int InputDimension { get; set; }
    public List<Neuron> Neurons { get; set; }

    public void Activate()
    {
        var output = new double[Neurons.Count];
        for (var i = 0; i < Neurons.Count; i++)
```

```

        {
            output[i] = Neurons[i].Activate();
        }
        LastOutput = output;
    }
}

```

Neurons – список нейронов, которые принадлежать текущему слою;
 InputDimension – размерность входного вектора; количество входных связей каждого нейрона в слое;

LastOutput – хранит последний вектор выходных значений слоя;

Activate – активирует все нейроны, которые принадлежать текущему слою, и сохраняет вектор выходных значений слоя в поле LastOutput.

Сеть.

Нейронная сеть представлена в коде следующим образом:

```

public class NeuronNetwork : INeuralNetwork
{
    List<Layer> layers { get; set; }
    List<Link> input { get; set; }

    public double[] Compute(double[] input)
    {
        for (var i = 0; i < input.Length; i++)
        {
            this.input[i].SendImpulse(input [i]);
        }
        this.layers.ForEach(l => l.Activate());
        return this.layers[this.layers.Count - 1].LastOutput;
    }
}

```

Compute – отправляет входной вектор на первый слой сети, затем итеративно, начиная с первого, активирует слои таким образом, что выходной вектор предыдущего слоя становится входным для следующего. При завершении распространения входного сигнала в прямом направлении, возвращает выходной вектор последнего слоя сети.

Алгоритм обучения.

Для обучения нейронной сети, предварительно необходимо сформировать обучающую выборку, каждый элемент которой выглядит следующим образом:

```
public class TrainingSample
{
    public double[] Sample { get; set; }
    public double[] Answer { get; set; }
}
```

Sample – хранит значения входных параметров.

Answer – хранит ожидаемые выходные значения сети.

Стохастический градиентный спуск подразумевает корректировку весовых коэффициентов сети после подачи каждого обучающего примера. Для того, чтобы быть в курсе динамики состояния сети, обучающий алгоритм, используя события, сообщает подписчику о текущих результатах обучения. Классом TrainAlgorithmConfig описываются параметры алгоритма обучения:

```
public class TrainAlgorithmConfig
{
    public long Epoches { get; set; }
    public IProximityMeasure ErrorFunction { get; set; }
    public double TrainingSpeed { get; set; }
    public double InertialFactor { get; set; }
    public double StimulatingFactor { get; set; }
}
```

Epoches – количество эпох. Алгоритм машинного обучения может просматривать обучающую выборку множество раз, при этом один полный проход по выборке называется эпохой обучения. Следовательно, можно варьировать количество прокруток одних и тех же данных при обучении, используя данный параметр.

ErrorFunction – функция ошибки. Используется для оценки работы сети во время её обучения.

TrainingSpeed – коэффициент скорости обучения.

InertialFactor – коэффициент инерции. Определяет меру влияния предыдущих подстроек на текущую.

StimulatingFactor – коэффициент стимуляции. Определяет меру влияния предыдущего значения весового коэффициента при расчете нового.

В качестве метода обучения нейронной сети был выбран метод обратного распространения ошибки, алгоритм которого представлен на рисунке А.1.

Суть метода заключается в том, что мы подаем сигнал на вход нейронной сети, далее, сравнивая выходной сигнал с желаемым, приступаем к корректировке внутренних составляющих сети. Ниже представлен код, который вычисляет градиент для каждого нейрона в выходном слое:

```
var neuron = layer.Neurons[i];  
var function = neuron.ActivationFunction;  
  
neuron.dEdS = config.ProximityMeasure  
    .ComputePartialDerivative(answer, layer.output, i)  
    * function.ComputeFirstDerivative(neuron.LastNET);
```

где i – порядковый номер нейрона в выходном слое.

Для вычисления градиента нейрона внутреннего слоя, необходимо, чтобы был известен градиент каждого нейрона, с которым текущий нейрон объединен выходными связями:

```
var d = function.ComputeFirstDerivative(neuron.LastNET);  
var e = InnerNeuronError(neuron);  
neuron.dEdS = d * e;
```

Внутренне содержимое функции InnerNeuronError, в которой происходит вычисление сигнала ошибки скрытого нейрона, выглядит следующим образом:

```
foreach (var outgoingLink in neuron.OutgoingLinks)  
{  
    result += outgoingLink.Destination.dEdS  
    * outgoingLink.Weight;  
}
```

После вычисления градиента, синаптические веса нейрона корректируются в соответствии с дельта правилом с добавлением к нему момента инерции и стимуляции:

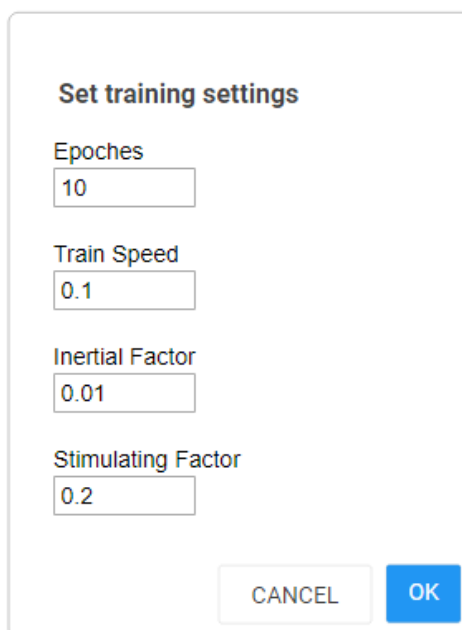
```
var grad = neuron.dEdS * incomingLink.Origin.OUT;  
var inertialMoment = incomingLink.dw * config.InertialFactor;  
  
var correctedWeight = (1 - config.StimulatingFactor)
```

* incomingLink.Weight
- grad * config.TrainingSpeed
+ inertialMoment;

incomingLink.dw = correctedWeight - incomingLink.Weight;
incomingLink.Weight = correctedWeight;

Обучение классификатора.

Перед началом обучения классификатора, необходимо установить все параметры алгоритма обучения, изображенные на рисунке 4.11.



Set training settings

Epoches
10

Train Speed
0.1

Inertial Factor
0.01

Stimulating Factor
0.2

CANCEL OK

Рисунок 4.11 – Параметры алгоритма обучения

Далее, при нажатии на кнопку «ОК», должен появиться обозреватель папок, изображенный на рисунке 3.14, через который необходимо выбрать обучающую выборку. Требуется, чтобы выбранная папка состояла из подпапок, имена которых соответствуют именам классов документов расположенных в этих подпапках.

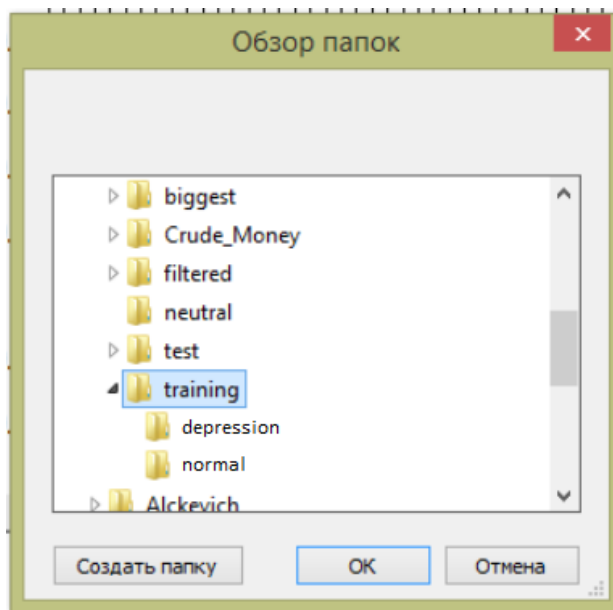


Рисунок 4.12 – Загрузка обучающей выборки

После того, как обучающая выборка будет передана в программу, на экране начнет рисоваться график, отображающий процесс обучения нейронной сети (см. рисунок 4.13).

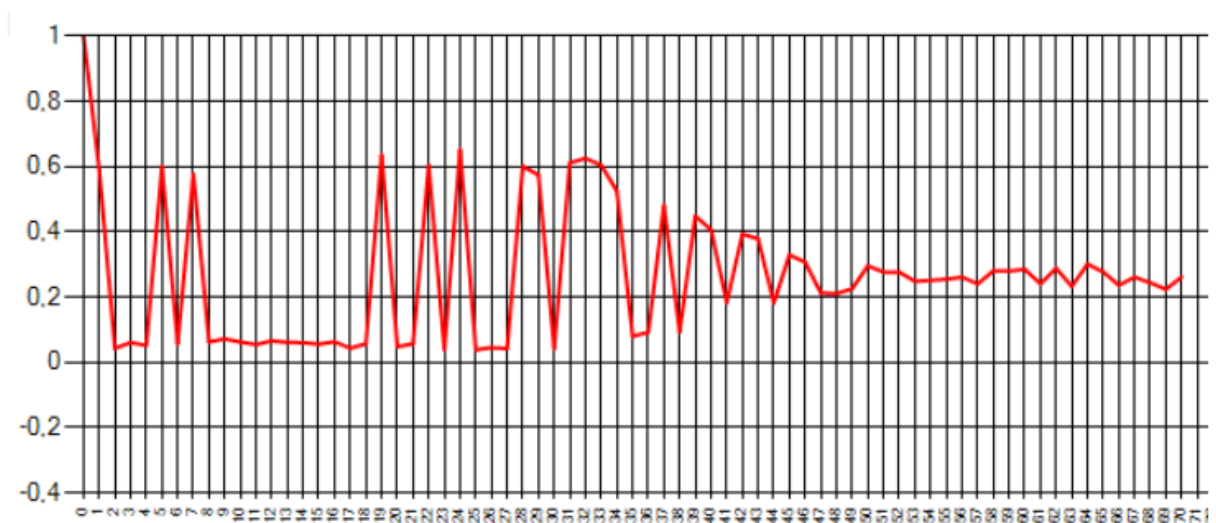


Рисунок 4.13 – График обучения сети

4.5 Веб-приложение

Веб-приложение позволяет врачу получить доступ к поведенческой информации конкретного пациента, а также увидеть предсказанное системой

его психологическое состояние. Приложение было написано с использованием технологий ReactJS, HTML 5.0 и CSS 3.0.

Для удобства перемещения и определения места положения на сайте была создана боковая панель навигации. Как можно увидеть на рисунке 4.14.



Рисунок 4.14 – Боковая панель навигации

Когда пользователь переходит на какую-либо страницу, соответствующая секция в навигационной панели выделяется оранжевым цветом. Для пациентов в боковом меню была выделена секция «Patients».

При нажатии на кнопку «Patients» в боковой панели навигации, система направляет пользователя на страницу со списком пациентов (см. рисунок 4.15).

Hi, Dr. Alex

HOME	Number	Patient Name	Age	Sex	Mental state
PATIENTS	1	William	31	Male	-
SETTINGS	2	Michael	18	Male	Depression
	3	Sophia	23	Female	-
	4	Alexander	27	Male	-
	5	Aiden	25	Male	Depression
	6	Olivia	21	Female	-
	7	Emily	23	Female	-
	8	Liam	32	Male	-
	9	Lily	33	Female	-

Copyright © 1995-2019 Alkevich Alexander

Рисунок 4.15 – Список пациентов

Список пациентов состоит из следующих колонок:

- Number – уникальный номер пациента. Автоматически присваивается пациенту системой при создании;
- Patient Name – имя пациента;
- Age – возраст пациента;
- Sex – пол пациента;
- Mental state – психологическое состояние пациента. Автоматически вычисляется системой на основе поведенческой информации пациента. Как можно увидеть на рисунке 4.15, пациенты с вычисленным диагнозом выделяются в списке желтым цветом.

При наведении курсора мыши на строку с пациентом в таблице, строка подсвечивается оранжевым цветом (см. рисунок 4.16).

	Number	Patient Name	Age	Sex	Mental state
HOME					
PATIENTS					
SETTINGS					
	1	William	31	Male	-
	2	Michael	18	Male	Depression
	3	Sophia	23	Female	-
	4	Alexander	27	Male	-
	5	Aiden	25	Male	Depression
	6	Olivia	21	Female	-
	7	Emily	23	Female	-
	8	Liam	32	Male	-
	9	Lily	33	Female	-

Рисунок 4.16 – Подсветка пациента при наведении курсора мыши

При нажатии на строку пациента в таблице, система направляет пользователя на страницу с детальной информацией о пациенте (см. рисунок 4.17).

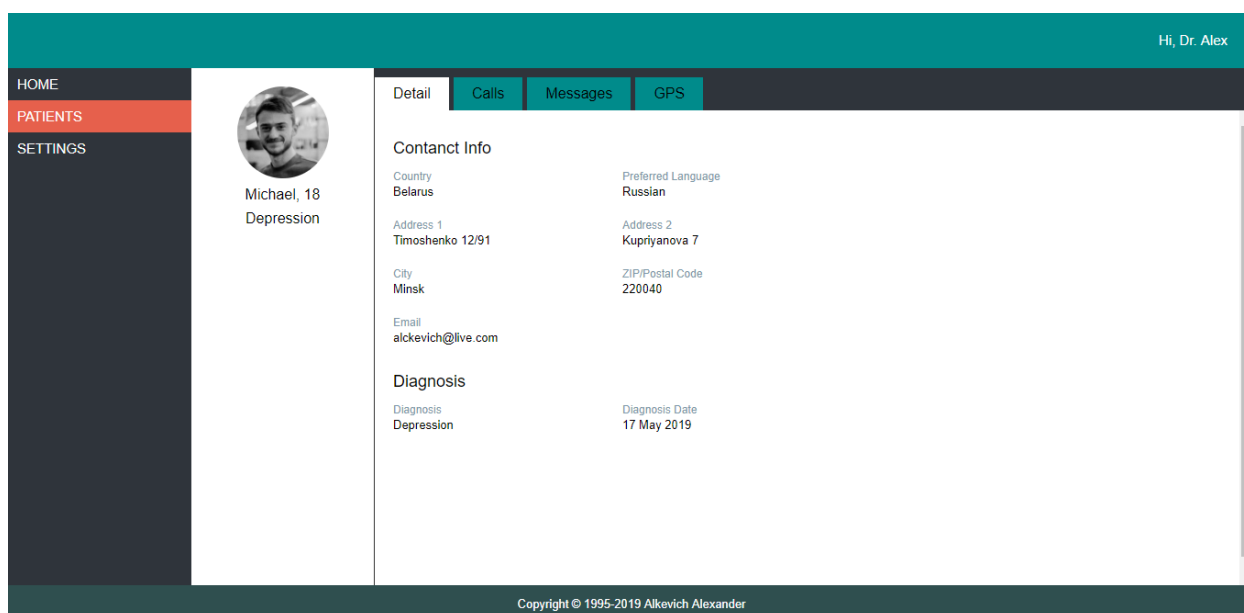


Рисунок 4.17 – Страница с детальной информацией о пациенте

Страница пациента состоит из следующих частей:

1. Боковая панель – содержит фотографию, имя, возраст пациента и его диагноз, который был автоматически присвоен пациенту системой. Панель остается видна, когда пользователь переключается между вкладками страницы пациента.

2. Вкладка «Detail» – содержит контактную информацию (страна, язык, адрес, город, адрес электронной почты, почтовый индекс) и диагноз пациента.

3. Вкладка «Calls» (см. рисунок 4.18) – отображает историю звонков пациента в виде обычного списка и столбчатого графика. При построении столбчатого графика по оси ординат было отложено количество звонков, по оси абсцисс – дни; каждому количеству звонков соответствует столбик. У пользователя есть возможность указать интересуемый временной интервал с помощью полей «From» и «To».

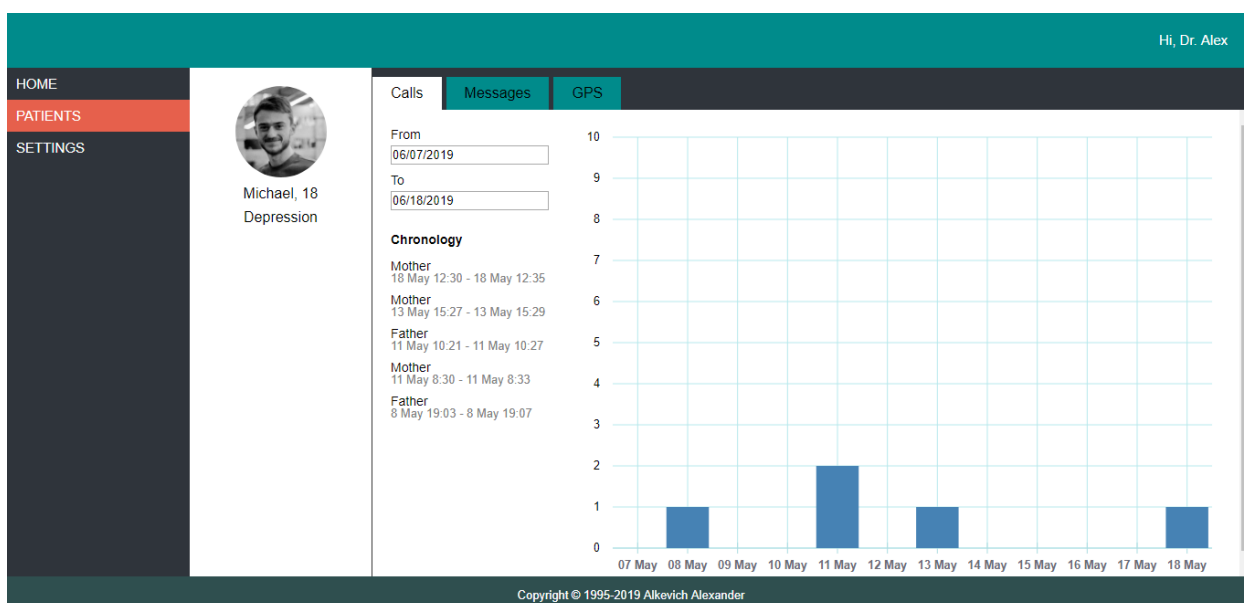


Рисунок 4.18 – История звонков пациента

4. Вкладка «Messages» (см. рисунок 4.18) – отображает историю сообщений пациента в виде обычного списка и столбчатого графика. При построении столбчатого графика по оси ординат было отложено количество сообщений, по оси абсцисс – дни; каждому количеству сообщений соответствует столбик. У пользователя есть возможность указать интересующий временной интервал с помощью полей «From» и «To».

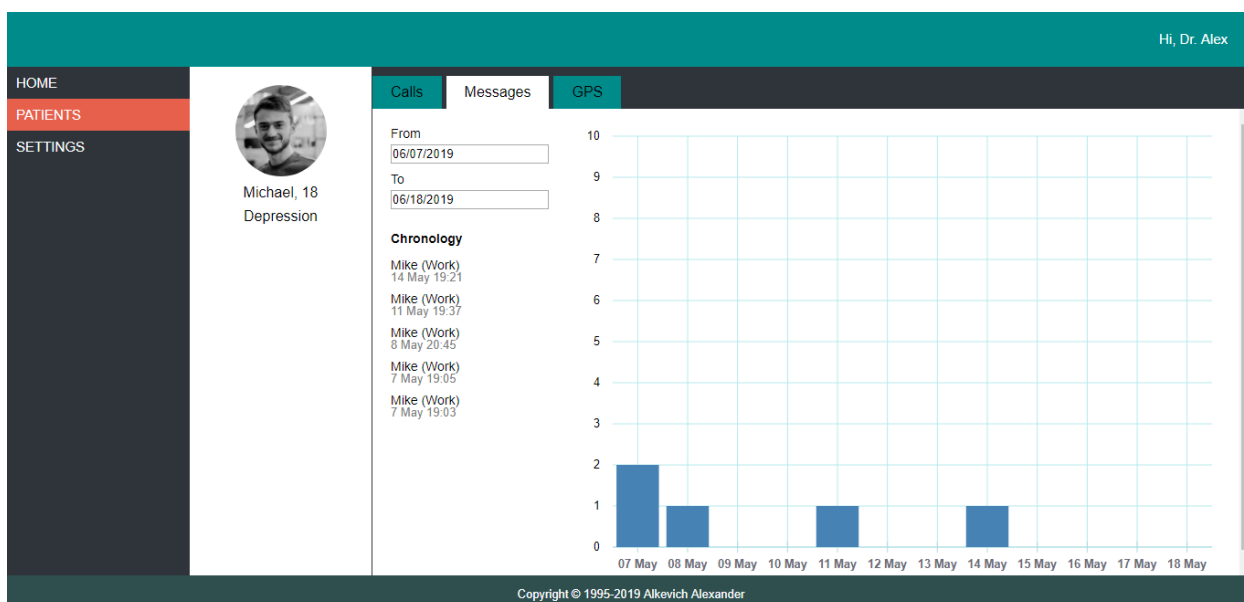


Рисунок 4.19 – История сообщений пациента

5. Вкладка «GPS» (см. рисунок 4.20) – отображает историю перемещений пациента в виде обычного списка и интерактивной карты с метками. У пользователя есть возможность указать интересуемый временной интервал с помощью полей «From» и «To».

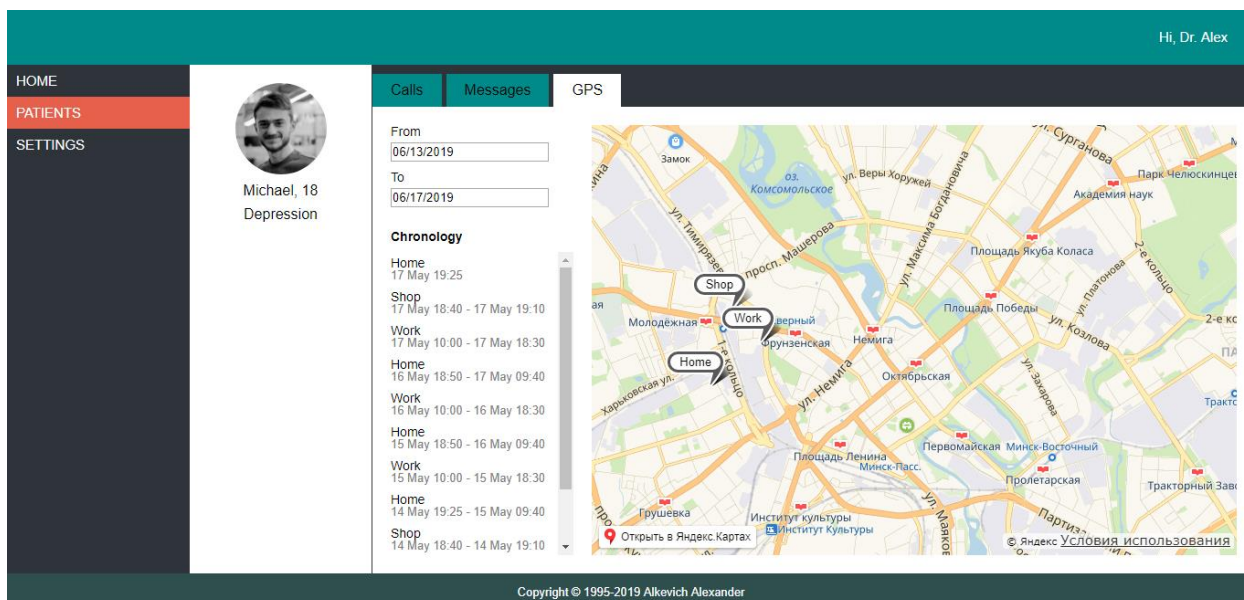


Рисунок 4.20 – История перемещений пациента

ГЛАВА 5

ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

5.1 Автоматизированное тестирование

Из автоматизированных видов в системе используется модульное и интеграционное тестирование.

Алгоритм любого теста имеет примерно следующую схему:

- инициализация класса, метод которого будет тестироваться;
- подготовка тестовых данных;
- передача тестовых данных в тестируемый метод;
- проверка правильности результата.

Для написания автоматизированных тестов были использованы следующие технологии:

1. NUnit – предоставляет богатый набор утверждений в качестве статических методов класса Assert. Если утверждение не выполняется, происходит оповещение об ошибке и тест считается не пройденным.

2. AutoMock – решает проблему внедрения зависимостей при написании модульных тестов. AutoMock сканирует зависимости тестируемого класса и подставляет пустые объекты вместо реальных.

3. DotCover – это инструмент для запуска юнит-тестов в приложениях на платформе .NET. DotCover анализирует покрытие кода тестами, определяет, какие именно тесты покрывают то или иное место в коде, а также подсвечивает покрытый и непокрытый код непосредственно в редакторе Visual Studio. Кроме того, dotCover позволяет агрегировать данные из нескольких сессий анализа покрытия, создавать отчеты в форматах XML, HTML или JSON.

Основные аспекты бизнес-логики приложения покрыты модульными тестами. Как правило, это различные вычислительные операции, обработка данных, управление правами доступами.

Пример тестирования работы нейронной сети:

```
[Test]
public void ShouldClassifyIntersectingSamples() {
    buildOptions = new NeuralNetworkBuilderOptions
    {
        ActivationFunction = new SigmoidFunction(1),
        Layers = new List<int> { 9, 3 }
    }
}
```

```

};

neuralNetwork = new NeuronNetworkBuilder(buildOptions).Build();
new BackPropagationAlgorithm(neuralNetwork, trainOptions)
    .Train(intersectingTrainSamples);

Assert.That( (neuralNetwork
    .ComputeOutput(trainSamples[0].Sample), Is.EqualTo(0));
Assert.That(neuralNetwork
    .ComputeOutput(trainSamples[1].Sample), Is.EqualTo(1));
Assert.That(neuralNetwork
    .ComputeOutput(trainSamples[2].Sample), Is.EqualTo(2));
}

```

Данный модульный тест проверяет, что сеть способна классифицировать образы, входные параметры которых пересекаются.

Работа с базами данных, а также эмуляция HTTP запросов к веб-приложению покрыты интеграционными тестами. Интеграционные тесты позволяют протестировать создание, обновление, удаление данных. С точки зрения веб-приложения, интеграционные тесты позволяют проверить работоспособность различных обработчиков запросов, правильность формата полученных данных из запроса, правильность возвращаемого сервером ответа.

Пример интеграционного теста получения данных из таблицы пациентов из тестовой базы данных:

```

[Test]
public async Task ShouldReturnPatients()
{
    var context = new AppContext();
    queryHandler = new GetPatientListQueryHandler(context);
    var query = new GetPatientListQuery();
    var result = await queryHandler.HandleAsync(query);
    Assert.That(result.Count(), Is.EqualTo(2));
}

```

В начале каждого теста удаляется тестовая база данных. Далее создаётся «чистая» база данных, заполняется нужными данными для конкретного тест-кейса.

Интеграционные тесты, как правило, имеют значительно большее время выполнения, чем модульные. Если рассматривать промышленную разработку, то рекомендуется модульные тесты запускать при каждой сборке приложения перед каждой фиксацией изменений кода, а интеграционные – раз в день, либо перед релиз-сборкой на интеграционном сервере.

5.2 Ручное тестирование

При тестировании системы также применялся ручной подход. В таблице 5.1 представлен тестовый набор для выполнения функционального тестирования веб-приложения.

Таблица 5.1 – Тесты веб-приложения

Описание	Предварительное условие	Действия	Ожидаемый результат
Дата звонков пациента не должна быть меньше значения выбранного в поле «From».	Должен быть создан пациент с историей звонков.	1. открыть историю звонков пациента; 2. указать значение в поле «From».	История звонков отображается начиная с даты «From».
Дата звонков пациента не должна быть больше значения выбранного в поле «To».	Должен быть создан пациент с историей звонков.	1. открыть историю звонков пациента; 2. указать значение в поле «To».	История звонков отображается до даты «To».
Дата сообщений пациента не должна быть меньше значения выбранного в поле «From».	Должен быть создан пациент с историей сообщений.	1. открыть историю сообщений пациента; 2. указать значение в поле «From».	История сообщений отображается начиная с даты «From».

Продолжение таблицы 5.1

Дата сообщений пациента не должна быть больше значения выбранного в поле «To».	Должен быть создан пациент с историей сообщений.	1. открыть историю сообщений пациента; 2. указать значение в поле «To».	История сообщений отображается до даты «To».
Дата перемещений пациента не должна быть меньше значения выбранного в поле «From».	Должен быть создан пациент с историей перемещений.	1. открыть историю перемещений пациента; 2. указать значение в поле «From».	История перемещений отображается начиная с даты «From».
Дата перемещений пациента не должна быть больше значения выбранного в поле «To».	Должен быть создан пациент с историей перемещений.	1. открыть историю перемещений пациента; 2. указать значение в поле «To».	История перемещений отображается до даты «To».

Для управления данными полученными в результате тестирования используется программное обеспечение TestRail. Данный инструмент помогает создавать тест-кейсы, управлять тестовыми наборами и координировать весь процесс тестирования программного обеспечения.

ЗАКЛЮЧЕНИЕ

Реализация интеллектуальной системы, способной следить за психологическим состоянием человека – это решение технически и математически сложной задачи, требующей углубленного изучения предметной области, собственного анализа, умения и навыков при обработке и выборе экспериментальных данных, глубокие знания в области дискретной математики и программировании, а также психологии и многих других областях науки.

Использование нейронных сетей, при разработке систем классификации, позволяет структурировать плохоформализованные данные, ускорить процесс обработки данных, оценить адекватность среды, в которую помещена информационная модель нейронной сети.

Основная цель магистерской диссертации заключалась в создании системы способной в режиме реального времени оценивать психологическое состояние человека, а также в выборе информационной модели нейронной сети и описании алгоритма классификации.

Была выбрана информационная модель на основе многослойного персептрона. Данная модель оптимальна по внутренней структуре и способу управления информационными потоками между нейронами. Кроме того, такая модель способна минимизировать число входных элементов. Модель на основе многослойного персептрона является универсальной моделью и подходит для решения задач разного уровня сложности, в том числе и для решения задачи классификации.

Основные выводы и результаты исследования:

1. На основании изученного материала по искусственным интеллектуальным системам были выявлены наиболее эффективные системы и точные методы классификации, отмечены успешные примеры реализации. Система, которая использовалась в данной работе являлась самообучающейся.

2. Обзор основных видов информационных моделей нейронных сетей позволил выделить две универсальные модели, применимые для широкого круга задач: радиальные нейронные сети прямого распространения и однонаправленные многослойные сети. Выбор информационной модели для решения практической задачи был сделан в пользу рекуррентных многослойных сетей. Данная модель подходит по внутренней структуре и математическому описанию и предусматривает разные подходы к решению задачи.

3. В практической части работы была разработана автоматизированная система по распознаванию психологического состояния человека. Система

предоставляет возможность в режиме реального времени отслеживать поведение человека через интернет, делать предположения о его психологическом состоянии. Система состоит из мобильного приложения, разработанного под платформу Android, двух веб-сервисов на платформе .NET Framework, одного ReactJs веб-приложения и одной реляционной базы данных.

На основании полученных результатов можно сделать вывод о том, что все поставленные задачи были выполнены в полном объеме, следовательно, главная цель была достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Список использованных источников

- [1] Аверин В.А., Психология личности: Учебное пособие. – СПб.: Изд-во Михайлова В.А., 2011. – 89 с.
- [2] Немов Р.С., Психология: Учеб. для студ. высш. пед. учеб. заведений: Пер. с англ. - М. : ООО "И.Д. Вильямс 2013. – 688с.
- [3] Андреева Г.М., Социальная психология: Учеб. для студ. высш. пед. учеб. заведений: Пер. с англ. - М. : ООО "И.Д. Вильямс 2013. – 229с.
- [4] A. Laya, V. I. Bratu, and J. Markendahl, «Who is investing in machine-tomachine communications?» in Proc. 24th Eur. Reg. ITS Conf., Florence, Italy, Oct. 2013, pp. 20–23.
- [5] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira, «Smart cities and the future internet», The Future Internet, Lect. Notes Comput. Sci., vol. 6656, pp. 431–446, 2011.
- [6] L. Atzori, A. Iera, and G. Morabito, «The internet of things: A survey», Comput. Netw., vol. 54, no. 15, pp. 2787–2805, 2010.
- [7] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, «Convergence of MANET and WSN in IoT urban scenarios», IEEE Sens. J., vol. 13, no. 10, pp. 3558–3567, Oct. 2013.
- [8] Уоссермен, Ф. Нейрокомпьютерная техника: Теория и практика: пер. с англ. Ю. А. Зуев / В. А. Точенов – М. : Феникс, 1992. – 184 с.
- [9] Научная библиотека избранных естественно научных изданий [Электронный ресурс]. – Электронные данные. – Режим доступа: http://sernam.ru/book_kir.php?id=25.
- [10] Определение эмоций по речи [Электронный ресурс]. – Режим доступа: <http://www.emotionlabs.ru/content/66/>.
- [11] Выражение эмоций в языке и речи [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/vyrazhenie-emotsiy-v-yazyke-i-rechi>.
- [12] Синтез речи, анализ эмоций и биометрия [Электронный ресурс]. – Режим доступа: <https://hh.ru/article/312409>.
- [13] Распознавание эмоций по речевому сигналу с помощью функций модуляционной теории звуковых сигналов [Электронный ресурс]. – Режим доступа: <http://euroasia-science.ru/tehnicheskie-nauki/raspoznavanie-emocij/>.
- [14] Хайкин, Саймон, Нейронные сети: полный курс, 2-е изд., испр. : Пер. с англ. - М. : ООО "И.Д. Вильямс 2016. – 1104с.

[15] О методах обучения многослойных нейронных сетей прямого распространения [Электронный ресурс]. – Режим доступа: https://www.nbpublish.com/library_get_pdf.php?id=21913.

[16] Цифровая обработка изображений [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.sibsauktf.ru/courses/fulleren/g3.htm>.

[17] Хайкин, Саймон, Нейронные сети: полный курс, 2-е изд., испр. : Пер. с англ. - М. : ООО "И.Д. Вильямс 2016. – 1104с.

[18] Распознавание текста в ABBYY FineReader [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://habrahabr.ru/company/abbyy/blog/225215>.

[19] Методы распознавания текста [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://habrahabr.ru/post/220077>.

[20] Нейронные сети [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.aiportal.ru/articles/neural-networks/1/>.

[21] Бинаризация изображений а [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://recog.ru/blog/applied/15.html>.

[22] Система кодирования лицевых движений [Электронный ресурс]// основные мимические выражения/ URL: <http://dic.academic.ru/dic.nsf/ruwiki/1628394> (дата обращения 16.04.17).

[23] Эмоциональная напряженность методика определения [Электронный ресурс]/URL: <http://pandia.ru/text/80/079/8545.php> (дата обращения 19.04.2017).

[24] Татаренков Д.А. Анализ методов обнаружения лиц на изображении//Технические науки. Молодой ученый. 2015. №4(84).

[25] Халафян А.А. Статистический анализ данных. 3-е изд. учеб./ Бином – Пресс. 2007. 512 с. 40. Царегородцев В. Г. Вычислительные технологии// Вестник КазНУ. ч 3. 2008. с. 308-315.

[26] Использование React.js на вебсайтах [Электронный ресурс]. – Режим доступа: <https://reactjs.org/>.

[27] Новые элементы в HTML 5 [Электронный ресурс]. – Режим доступа: <http://www.ibm.com/developerworks/ru/library/x-html5>.

[28] Бер Бибо, jQuery. Подробное руководство по продвинутому JavaScript: Символ-Плюс, 2010. – 384с.

[29] Code First в Entity Framework [Электронный ресурс]. – Режим доступа: <http://metanit.com/sharp/entityframework/1.2.php>.

[30] Рихтер Дж., CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. – Пер. с англ./ Дж. Рихтер – СПб: Питер, 2015. – 896 с.

Список публикаций соискателя

[1-А] Алькевич, А.С. Цифровая психология / А.С. Алькевич // Материалы 54-ой научной конференции студентов, магистрантов и аспирантов УО «Белорусский государственный университет информатики и радиоэлектроники» – Минск, 2018 – 15 с.

ПРИЛОЖЕНИЕ А
Предупреждающие
знаки безопасности
согласно ГОСТ

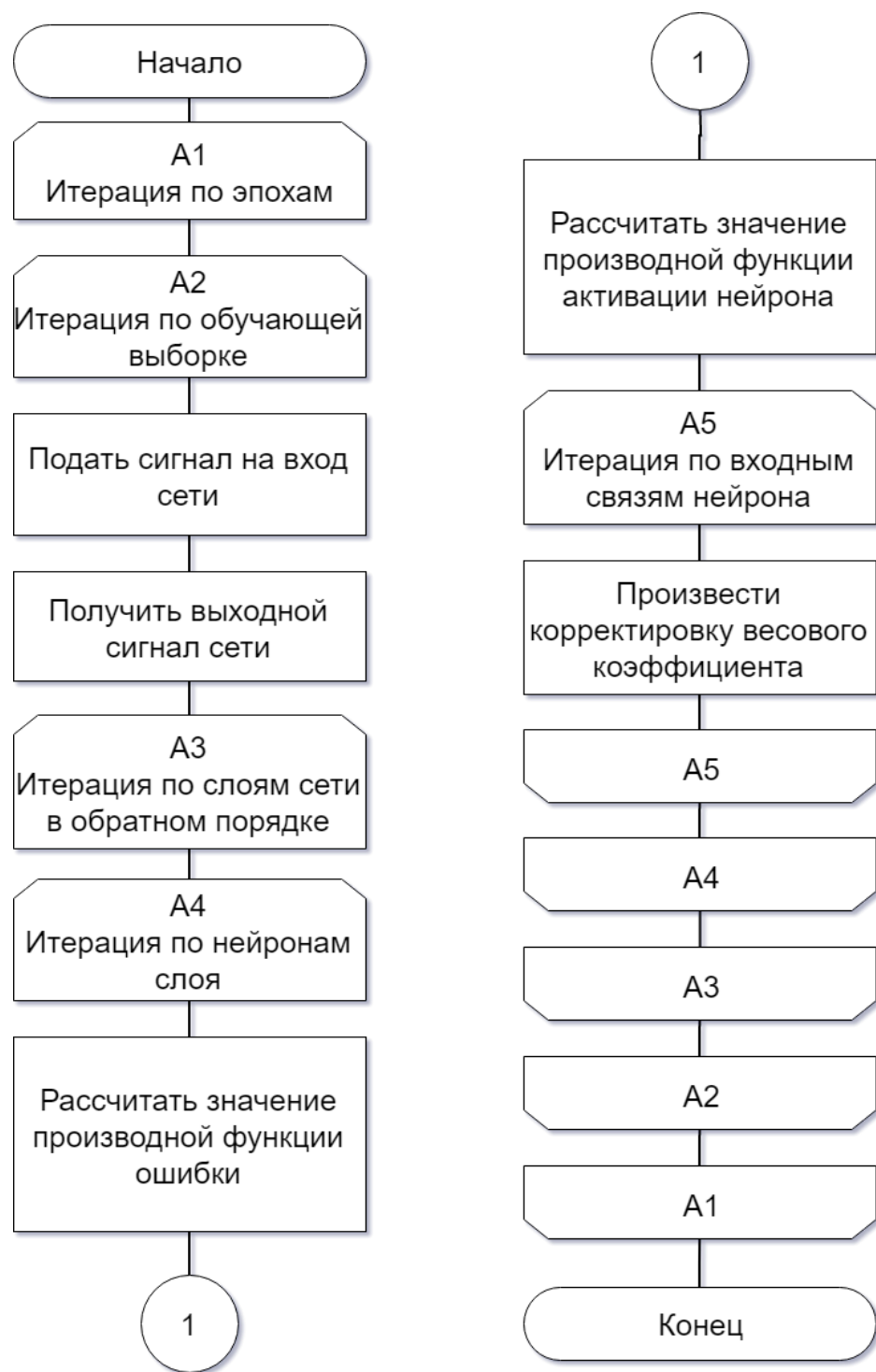


Рисунок А.1 – Алгоритм обратного распространения ошибки