МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Факультет вычислительной математики и кибернетики**

**Кафедра: Математического обеспечения ЭВМ**

Направление подготовки: «Прикладная математика и информатика»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**Тема:**

**«Инструментальная СИАД. Разработка библиотеки нейросетей»**

Заведующий кафедрой: Выполнил:

д.ф.-м.н., проф., студент группы 8403

Стронгин Роман Григорьевич Смирнов Михаил Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись подпись

Научный руководитель:

к.т.н., доц.,

Карпенко Сергей Николаевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Нижний Новгород  
2015

Содержание

[Список сокращений 4](#_Toc421361288)

[Введение 5](#_Toc421361289)

[1. Интеллектуальный анализ данных 7](#_Toc421361290)

[1.1. Понятие ИАД 9](#_Toc421361291)

[1.2. Задачи, решаемые с помощью ИАД 10](#_Toc421361292)

[1.2.1. Классификация по типу извлекаемой информации 10](#_Toc421361293)

[1.2.2. Классификация по способу решения задачи 12](#_Toc421361294)

[1.2.3. Классификация по назначению 13](#_Toc421361295)

[1.3. Методы ИАД 13](#_Toc421361296)

[1.3.1. Деревья решений 15](#_Toc421361297)

[1.3.2. Метод опорных векторов 19](#_Toc421361298)

[1.3.3. Метод ближайших соседей 21](#_Toc421361299)

[1.3.4. Нейронные сети 23](#_Toc421361300)

[1.3.5. Алгоритм Apriopi 32](#_Toc421361301)

[1.3.6. Генетические алгоритмы 33](#_Toc421361302)

[1.4. Этапы решения задач методами ИАД 37](#_Toc421361303)

[1.5. Обзор программных систем в области ИАД 39](#_Toc421361304)

[1.5.1. Oracle Data Mining 39](#_Toc421361305)

[1.5.2. DTREG 40](#_Toc421361306)

[1.5.3. STATISTICA Data Miner 40](#_Toc421361307)

[1.5.4. ORANGE DATA MINING 41](#_Toc421361308)

[1.5.5. NeuroShell 2 42](#_Toc421361309)

[2. Постановка задачи выпускной квалификационной работы 43](#_Toc421361310)

[3. ИСИАД. Общая архитектура 44](#_Toc421361311)

[3.1. Термины и определения 44](#_Toc421361312)

[3.2. Архитектура ИСИАД 46](#_Toc421361313)

[4. Разработка библиотеки нейросетей 50](#_Toc421361314)

[4.1. База данных ИНС 50](#_Toc421361315)

[4.1.1. Таблица NeuroNet 51](#_Toc421361316)

[4.1.2. Таблица NetTopology 52](#_Toc421361317)

[4.1.3. Таблица WeightsMatrix 52](#_Toc421361318)

[4.2. Библиотека ИНС 52](#_Toc421361319)

[4.2.1. Класс ActivateFunctionParameter 54](#_Toc421361320)

[4.2.2. Абстрактный класс ActivateFunction 54](#_Toc421361321)

[4.2.3. Статический класс LibraryOfActivateFunctions 55](#_Toc421361322)

[4.2.4. Абстрактный класс Topology 56](#_Toc421361323)

[4.2.5. Статический класс LibraryOfTopologies 56](#_Toc421361324)

[4.2.6. Класс NeuronInputConnection 56](#_Toc421361325)

[4.2.7. Класс Neuron 57](#_Toc421361326)

[4.2.8. Класс NeuroNet 57](#_Toc421361327)

[4.3. Работа с нейронными сетями 60](#_Toc421361328)

[4.3.1. Просмотр созданных ИНС 60](#_Toc421361329)

[4.3.2. Описание новой ИНС 61](#_Toc421361330)

[4.3.3. Выбор ИНС для решения задачи или обучения 62](#_Toc421361331)

[4.3.4. Решение задачи выбранной ИНС 63](#_Toc421361332)

[Заключение 64](#_Toc421361333)

[Список литературы 65](#_Toc421361334)

[Приложения 66](#_Toc421361335)

[Приложение А. Заголовочные части классов библиотеки активационных функций 66](#_Toc421361336)

[Приложение Б. Класс сигмоидальной активационной функции 67](#_Toc421361337)

[Приложение В. Класс топологии «Персептрон» 68](#_Toc421361338)

[Приложение Г. Заголовочные части классов NeuronInputConnection, Neuron и NeuroNet 69](#_Toc421361339)

# Список сокращений

АФ – Активационная функция

БД – База данных

ГА – Генетический алгоритм

ДР – Дерево решений

ИАД – Интеллектуальный анализ данных

ИИ – Искусственный интеллект

ИНС – Искусственная нейронная сеть

ИСИАД – Инструментальная система интеллектуального анализа данных

МБС – Метод ближайших соседей

МОВ – Метод опорных векторов

МОРО – Метод обратного распространения ошибки

# Введение

В последние десятилетия, благодаря бурному развитию вычислительной техники и информационных технологий в целом, электронные устройства стали использоваться практически в любой сфере деятельности человека. Любое из этих устройств является преобразователем одного массива данных в другой: от суперкомпьютеров, решающих трудоёмкие исследовательские задачи, до носимой электроники, являющейся своеобразным интерфейсом между пользователем и другим, более мощным вычислительным узлом. В связи с этим синтезируется огромное количество информации, как передаваемой «на вход» вычислительным устройствам, так и получаемой от них «на выходе». Возникает желание, проанализировав эти данные, попытаться оптимизировать работу в исследуемой области, увеличить доход или минимизировать затраты. Но уже в начале XXI века объемы информации, хранящейся в базе данных, соответствующей конкретной задаче в отдельно взятой компании достигли размеров, которые человек не в состоянии проанализировать за приемлемое время [1]. Поэтому было решено большую часть рутинной работы, возникающей при анализе больших объемов данных, переложить, опять же, на вычислительную технику. Так возникла целая совокупность алгоритмов и методов, позволяющих выявлять из данных нетривиальные, заранее неизвестные факты о рассматриваемой проблеме. Совокупность всех этих методов получила название «Интеллектуальный Анализ Данных». Стоит заметить, что изначально был введен англоязычный термин для данной области – «Data Mining», который не имеет четкого, устоявшегося перевода на русский язык. Вкратце, данный термин сравнивает извлечение ценной информации из большого объема данных с добычей полезных ископаемых и драгоценных металлов[[1]](#footnote-1). Важно, что методы ИАД позволяют представить результаты вычислений в наглядной (графической) форме, или описать их на естественном языке. Это позволяет использовать ИАД людям, не имеющим специальной математической подготовки. Поэтому, вкупе с эффективностью методов, Интеллектуальный Анализ Данных является актуальной темой для исследований и популярным механизмом анализа больших объемов информации.

Данная работа включает в себя программный проект «Инструментальная Система ИАД», содержащий несколько методов ИАД. Главное достоинство данного проекта заключается в том, что у пользователя ИСИАД есть возможность сравнения результатов работы методов ИАД над учебными данными (т.н. тестовой выборкой) в рамках некоторого критерия качества и последующего выбора метода ИАД для решения своей задачи.

Опишем содержание данной работы.

Первая глава содержит определение понятия ИАД, описание задач, решаемых методами ИАД и представление данных методов. Излагаются основные этапы решения задач с помощью ИАД, приводятся примеры созданных коммерческих и бесплатных систем ИАД.

Вторая глава описывает постановку задачи данной работы.

В третьей главе содержится общее описание проекта ИСИАД, его внутренняя архитектура, а также термины и определения, применяемые в данной работе и проекте ИСИАД. Содержание данной главы есть результат общей работы команды, создавшей ИСИАД.

Четвертая глава посвящена разбору архитектуры одного из методов ИАД, примененных в проекте ИСИАД – нейронных сетей. В ней определяются и излагаются все необходимые для работы метода вспомогательные элементы. Также представляется архитектура библиотеки нейронных сетей и демонстрируются возможности работы данного алгоритма в проекте ИСИАД. Создание библиотеки нейронных сетей есть индивидуальная задача в проекте ИСИАД автора данной работы.

# Интеллектуальный анализ данных

Особому интересу к моделированию интеллектуальных способностей человека в 60­-70-х годах XX века способствовало бурное развитие вычислительной техники. Развивались методы *Искусственного Интеллекта*, молодой науки, зародившейся в конце 40-х годов прошлого столетия. Были предприняты попытки дать формальное определение понятию «интеллект», «интеллектуальная деятельность». На методы ИИ возлагались большие надежды и давались смелые прогнозы относительно скорого создания «искусственного разума» [2]. Однако уже в 80-х годах интерес к ИИ угас вследствие несоответствия действительности данным прогнозам. К сожалению, первоначальная цель и задача ИИ – моделирование процессов интеллектуальной деятельности, а также постановка четкого определения понятия «интеллект» так и не достигнута. В начале XXI века интерес к методам ИИ вновь возрос, были достигнуты успехи во многих направлениях исследований. В частности, благодаря использованию *глубокого обучения* [3] был достигнут большой прогресс в задачах распознавания речи, компьютерного зрения. Были основаны новые области на стыке ИИ и других наук – одной из них является ИАД.

Кроме ИИ на ИАД большое влияние оказали методы математической статистики, теории вероятности и теории баз данных. Потребность в переосмыслении методологии анализа данных возникла благодаря огромному объему и разнообразию данных. Методы статистического анализа выдают вердикты, основанные на усреднении значений в выборке, что в условиях большого разнообразия данных не позволяет выявить из них важную, нетривиальную информацию.

Условно *интеллект* можно определить как операции над тремя сущностями: данными, информацией и знаниями. *Данные* отличаются от *информации* тем, что они явно не несут в себе смысловую нагрузку. Например, данными является массив чисел, каждое из которых представляет собой рост в сантиметрах конкретного жителя некоторого поселка. Информацией же, извлеченной из этих данных, могут быть наименьший, средний, наибольший рост среди жителей, или другие, более сложные конструкции. Следует все же отметить, что граница между данными и информацией довольно размыта. Однако еще более сложно определить, что такое знания. Под *знаниями* будем понимать состояние интеллекта (точнее, его части, отвечающей за решение текущей задачи), в котором он способен осознавать поступающую входную информацию и данные, а также синтезировать новую информацию и данные (относящиеся к текущей задаче). Заметим, что в рамках данного определения знания могут быть ошибочными или не оптимальными (при решении конкретной задачи); также оттуда следует, что знания являются чем-то средним между процессом решения задачи и информацией, требуемой для этого решения. Знания нельзя записать на бумаге (или на любом другом носителе): записанное будет лишь информацией, интерпретацией знаний. Таким образом, только интеллект способен вмещать знания, и только с их помощью он способен преобразовывать входные информацию и данные в выходные (Рисунок 1).

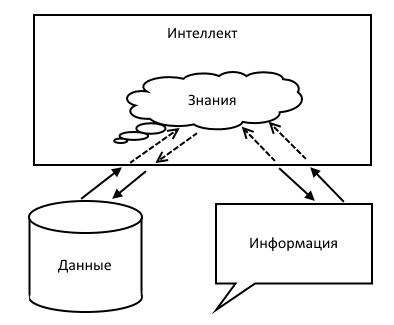


Рисунок . Формальное описание интеллектуальной деятельности

Вычислительная машина не различает данные и информацию, не содержит внутри себя знания [4]. Она способна лишь трансформировать данные из одного вида в другой (Рисунок 2). Поэтому главной задачей методов ИАД является совершение такого преобразования данных, при котором (с точки зрения человека) выходные данные алгоритмов ИАД содержали бы настолько очевидный практический смысл, что их без сомнения можно было бы назвать информацией. При этом данная информация должна соответствовать требованиям исследователя по ценности.

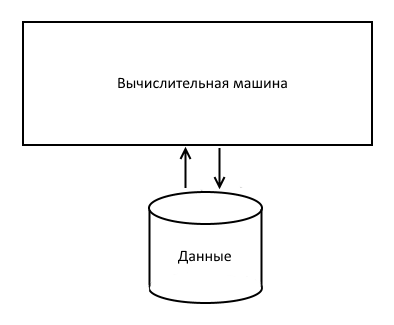


Рисунок . Формальное описание вычислительного процесса

Так как исторически ИИ создавался с целью создания искусственного разума, невозможность сохранения знаний в памяти ЭВМ побудила пересмотреть само понятие «знания». В узком смысле (применительно к методам ИИ), *знания* есть совокупность сведений, образующая целостное описание задачи, соответствующее некоторому уровню осведомленности об этой задаче. Т.е. применительно к определению в широком смысле, это есть *описание* *знаний*, информация, достаточная для принятия решений методами ИИ. Такое определение фактически позволяет признать, что ЭВМ, также как интеллект, может хранить в себе знания. Они представляются в памяти вычислительных устройств с помощью особых структур данных, таких, как *фреймы*, *семантические сети*, *продукционные правила*, *предикаты* логического языка 1-го порядка и т.д.

Так как большинство методов ИАД были разработаны в рамках области ИИ, в дальнейшем при упоминании термина «знания» будем иметь в виду понятие в узком смысле. Теперь рассмотрим предмет ИАД более детально.

## Понятие ИАД

*Интеллектуальный анализ данных* – область информационных технологий, решающая задачу извлечения полезной информации из больших объемов данных. Причем критерии полезности информации, которую должен извлечь метод ИАД, в большинстве случаев должны быть сформулированы исследователем заранее. То есть, большая часть методов ИАД ориентирована на поиск определенного вида *шаблонов* в данных. Эти шаблоны должны отражать характеристики информации, которую хочет извлечь исследователь [1].

Например, некоторый банк желает выявить среди пользователей своих карт лиц, осуществляющих сомнительные финансовые операции. В таком случае, изначально необходимо конкретизировать, по каким параметрам следует отслеживать пользователей – частота осуществления транзакций, объемы переводов и т.д. Вся эта информация должна быть легко извлекаема из базы данных клиентов банка (чтобы легко формировались выборки, необходимые для обучения, тестирования и работы методов ИАД). После этого необходимо также конкретизировать тип и структуру данных, которые должны получаться на выходе метода ИАД – в данном случае, это может быть описание всех сомнительных операций, в которых уличен конкретный пользователь, или, проще, вердикт – является данный пользователь мошенником, или нет. Данное описание выходных данных метода ИАД и является шаблоном, который должен быть выявлен из входных данных. Следует отметить, что при явном задании шаблона выходных данных пользователем, необходимо обладать статистикой зависимости между входными данными и значениями шаблона. Это необходимо для процесса обучения методов ИАД. В предыдущем примере данной статистикой могут быть данные о пользователях карт, дополненные фактом того, замечены они в сомнительных финансовых операциях или нет.

Так же существуют методы ИАД, способные обучаться и работать без шаблона выходных данных. Такие методы применяются в случаях, когда зависимости между входными данными и глубинная информация, содержащаяся в них, неочевидны самому исследователю, или информация, извлекаемая методом ИАД, должна основываться на некотором интуитивном критерии сходства (различия) входных данных. Например, задачи этого класса используются в маркетинге: среди всех клиентов, покупателей, товаров выделяются схожие друг с другом сущности, из них формируются группы. Для каждой группы схожих сущностей уже можно использовать методы ИАД, основанные на выявлении шаблонов. Так, каждая группа будет изучена по отдельности, после чего можно будет разрабатывать для взаимодействия с ней отдельную модель поведения.

Проблемы, в которых допускается задание пользователем шаблонов, принадлежат к одному классу задач, решаемых методами ИАД, а проблемы, при решении которых метод обязан самостоятельно сгенерировать шаблон – к другому. В следующем разделе приводится классификация задач, решаемых с помощью ИАД.

## Задачи, решаемые с помощью ИАД

Границы применимости методов ИАД довольно сильно размыты; в связи с этим могут появляться специфичные задачи, не подпадающие под классическую классификацию, приведенную ниже [5].

Задачи, решаемые методами ИАД, можно классифицировать по следующим признакам:

* *По типу извлекаемой информации*
* *По способу решения задачи (по способу обучения)*
* *По назначению задачи*

Рассмотрим каждую из приведенных классификаций.

### Классификация по типу извлекаемой информации

Как было отмечено выше, некоторые задачи позволяют пользователю проектировать шаблон, другие же устроены так, что шаблон может быть построен только методом ИАД. Очевидно, что сгенерированные шаблоны можно классифицировать вследствие детерминированности методов, генерирующих их, и конечности числа таких методов. Однако оказывается, что также можно выделить основные типы шаблонов, создаваемых исследователями вручную. В связи с этим, множество задач, решаемое методами ИАД, может быть классифицировано по типу извлекаемой информации.

Таким образом, выделяют следующие типы задач, решаемых методами ИАД [5]:

* *Классификация*. В ходе выполнения задачи классификации обнаруживаются признаки, характеризующие группы объектов исследуемого набора данных. Т.е. методы ИАД, решающие задачи классификации, разделяют все множество исследуемых ими данных на *классы*. Признаки классификации составляют шаблон, разрабатываемый исследователем вручную.
* *Кластеризация.* Результатом работы методов ИАД, решающих задачи кластеризации также, как и в случае классификации, является разбиение объектов на группы. Отличие заключается в том, что в задачах кластеризации классы изначально не предопределены; методы ИАД обязаны определить количество классов и их содержание автоматически.
* *Ассоциация*. При решении задачи данного типа отыскиваются закономерности (ассоциативные правила) между признаками данных. Классической задачей выявления ассоциаций является *анализ потребительской корзины*, где по некоторым данным о клиентах (социальный статус, возраст, и т.д.) устанавливается, какие типы товаров чаще всего приобретают клиенты, относящиеся к рассматриваемому классу. Шаблон генерируется автоматически, однако допускается возможность указания, по каким критериям следует проводить поиск (данные о клиенте), а по каким – искать ассоциации (приобретенные товары).
* *Анализ последовательностей.* Это задача, в которой необходимо найти закономерности между событиями, произошедшими в разные моменты времени. Отличается от задачи ассоциации тем, что в последней не фигурирует время событий, между которыми необходимо провести ассоциацию. Т.о., если правило ассоциации звучит как *«вместе с событием X происходит событие Y»*, то правило последовательности есть *«после наступления X через определенное время произойдет Y»*.
* *Восстановление зависимости (регрессии)*. В данной задаче значения шаблона принадлежат бесконечному (счетному или континуальному) множеству. В остальному задача схожа с задачей классификации.
* *Прогнозирование*. В задачах данного типа на основании исторической последовательности данных оцениваются значения целевых показателей, ожидаемых в будущем, или в пропущенных моментах времени данной последовательности.
* *Определение отклонений или выбросов*. Целью данной задачи является обнаружение и анализ данных, наиболее отличающихся (по одному или нескольким признакам) от общего множества данных, выявление нехарактерных шаблонов данных. Отклонения могут сигнализировать о необычном событии (неожиданный результат эксперимента, мошенническая операция по банковской карте) или об ошибке ввода данных в БД.

### Классификация по способу решения задачи

Так как большинство методов ИАД были заимствованы из области ИИ, они имеют довольно специфичный для обычных методов и алгоритмов процесс работы: перед первым использованием таких методов (для каждой отдельной задачи) их нужно *обучать*. В связи с этим, возникают дополнительные требования на данные, предназначенные для решения задач. Эти требования разделяют множество задач, решаемых с помощью ИАД, на следующие классы:

* *Использующие обучение с учителем*. В задачах данного класса должна иметься довольно большая статистика верных решений данной задачи. Т.е., формально, должен быть массив данных, каждый элемент которого представлен в виде , где – значения набора атрибутов, характеризующего предмет рассматриваемой задачи, а – значения шаблона, получаемого методами ИАД на выходе. Например, в задаче определения качества выпущенного предприятием изделия есть результаты испытаний, проводимых над изделием, а – вердикт, определяющий его качество. Данный массив разделяется на части, одни из которых задействованы в обучении метода ИАД, а другие – в проверке качества этого обучения. Первые называются *обучающими*, а вторые – *тестовыми* выборками. Существуют различные алгоритмы обучения с учителем, отличающиеся процессом разделения *генеральной выборки* на обучающие и тестовые, количеством данных выборок, последовательностью их применения и т.д. К классу использующих обучение с учителем относятся задачи *классификации*, *восстановления* *зависимости* и *прогнозирования*. Подробнее про данный процесс обучения и методы ИАД, обучающиеся с учителем, изложено далее (см. главу 1.3).
* *Использующие обучение без учителя*. Отличие данного класса задач от рассмотренного выше заключается в том, что при использовании методов, обучающихся без учителя, генеральная выборка не содержит статистику значений шаблона; она представляет собой лишь данные, представляющие текущую задачу. Таким образом, вся генеральная выборка является обучающей, тестовую выборку сформировать нельзя, т.к. результаты, выдаваемые обученным методов ИАД, не с чем сравнивать, нет известного правильного вердикта. Методы, обучающиеся без учителя, автоматически формируют шаблон, в отличие от методов, обучающихся с учителем, где создание шаблона возложено на исследователя. К данному классу относится задача *кластеризации*, так же сюда можно отнести (с некоторой натяжкой) задачу *определения отклонений*.
* *Не использующие обучение*. Сюда относятся задачи, процесс решения которых достаточно эффективен с использованием классических алгоритмов. К таким задачам относятся *ассоциация*, *анализ последовательностей*, *определение отклонений*. Для решения, например, первой задачи разработан т.н. *алгоритм Apriopi* (см. главу 1.3).

### Классификация по назначению

Более общей, чем рассмотренные выше, является классификация задач, решаемых с помощью ИАД, по назначению, конечной цели решения задачи. В соответствии с такой классификацией выделяют 2 типа задач:

* *Описательные*. Концепция данных задач подразумевает сравнение наборов данных, создание их характеристики. Исследователь получает шаблоны, описывающие данные, определяющие их отличительные особенности. Цель решения данных задач – нахождение нетривиальной, полезной информации *в текущем объеме* данных.
* *Предсказательные*. Этот класс задач основан на анализе текущих данных с целью создания модели, предсказания тенденций или свойств новых, или неизвестных событий. Т.о. в данных задачах цель есть нахождение важной информации в данных, *отнесенных к будущему*.

## Методы ИАД

В данном разделе рассматриваются основные и наиболее популярные методы ИАД. Все они были созданы в рамках областей ИИ и *машинного обучения*. В общем виде метод ИАД может быть представлен как функционал:

|  |
| --- |
|  |

*где – вектор атрибутов данных , из которых методом выявляется информация, ценная для исследователя; – параметры метода ИАД; – заполненный шаблон выходной информации. При этом каждая из компонент , и может иметь свой тип, определяющий область допустимых значений данной компоненты.*

При этом методы, не требующие обучения, не содержат параметров. Также можно определить *функцию качества* решения задачи методом ИАД:

|  |
| --- |
|  |

*где – значение шаблона, полученное рассматриваемым методом для данных ; - абсолютно верное значение шаблона для данных . Считается, что чем меньше значение для конкретных и , тем выше качество решения .*

Заметим, что применить функцию качества можно лишь в том случае, если *доступна генеральная выборка* для текущей задачи. Т.о. задача *обучения с учителем* есть процесс нахождения параметров метода , таких, что для всех из *обучающей выборки* выполняется равенство:

|  |
| --- |
|  |

То есть задача *обучения с учителем* – это задача оптимизации функции в многомерном пространстве параметров метода .

Данный процесс обучения является *внешним* по отношению к методу ИАД: ему не важна внутренняя структура метода, важен только результат его работы , по которому вычисляется качество обучения метода. Т.е. для алгоритма, осуществляющего обучение с учителем, обучаемый метод фактически является черным ящиком (Рисунок 3-а). Однако большинство таких методов обучения все же разработаны под конкретный метод ИАД, используют дополнительную информацию, специфичную для метода. Одними из самых популярных независимых от *решателя* (метода ИАД) методов обучения являются *генетические алгоритмы*.

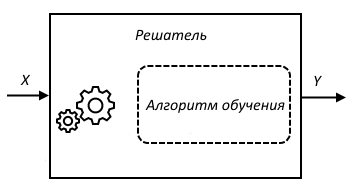
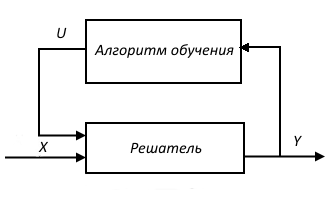
**Процесс *обучения без учителя*, напротив, является *внутренним* по отношению к методу (Рисунок 3-б). Каждый метод, обучающийся таким образом, реализует свой, предназначенный только для себя метод обучения. Как правило, такие *самоорганизующиеся* методы используют некоторую метрику в пространстве векторов для отыскания наилучшего вектора .

Рисунок 3-б. Схема обучения без учителя.

Рисунок 3-а. Схема обучения с учителем.

Ниже будут рассмотрены следующие методы ИАД:

* *Деревья решений*
* *Метод опорных векторов*
* *Метод ближайших соседей*
* *Нейронные сети*
* *Алгоритм Apriori*

Кроме этого будет рассмотрен универсальный алгоритм *обучения с учителем* – *генетический алгоритм*, формально не являющийся методом ИАД, но применяющийся в их обучении.

### Деревья решений

*Дерево решений* – способ представления набора *продукционных правил*[[2]](#footnote-2), извлеченных из данных, в последовательной, иерархической структуре [6]. Каждый узел такого дерева представляет собой вопрос, относящийся к рассматриваемой проблеме. Каждое ребро помечено возможным ответом на данный вопрос; утвердительный ответ на вопрос в узле переводит состояние рассматриваемой задачи по соответствующему ребру в следующий узел до тех пор, пока не будет достигнут один из листьев дерева. Каждый лист содержит в себе утверждение, представляющее собой решение рассматриваемой проблемы. Пример ДР для задачи, в которой банк определяет, выдавать ли клиенту кредит, приведен ниже (см. Рисунок 4).

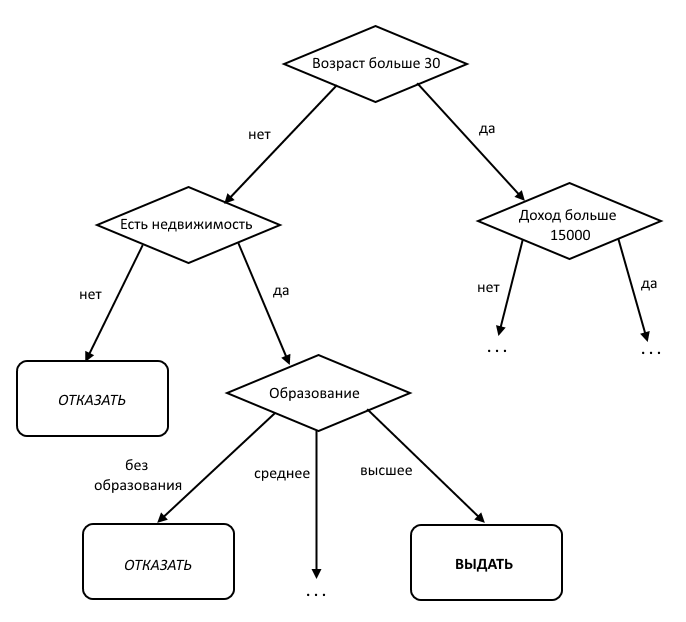


Рисунок . Пример дерева решений

ДР широко применяются при решении задач *классификации* и *восстановления зависимости*, могут также применяться в задачах *прогнозирования* и *определения отклонений*. Алгоритмы построения ДР относятся к методам *обучения с учителем*. В случае решения задачи классификации каждый лист дерева содержит идентификатор класса, к которому следует отнести рассматриваемый объект. В случае задачи *восстановления зависимости* – соответствующее ситуации непрерывное значение выходного параметра.

Так как при решении задачи с шаблоном из элементов с помощью ДР конкретное значение выходного вектора является согласованным по всем компонентам (конкретный узел задает значение сразу всего вектора ), без ограничения общности можно считать, что шаблон имеет всего 1 атрибут, и .

При использовании большинства методов ИАД на этапе формирования генеральной выборки (в общем случае из различных БД) главной задачей является определение состава атрибутов входных данных, значения которых должны составить входные векторы . Для результатов обучения многих методов является существенным нахождение или отсутствие -го атрибута. При использовании ДР можно не думать о этой проблеме и использовать в качестве входных атрибутов все доступные (кроме целевого). Алгоритмы построения ДР сами выберут наиболее значимые атрибуты, от которых действительно зависит результат.

Опишем *процесс построения дерева*. Пусть целевой атрибут во всей обучающей выборке принимает значения из множества . Причем хотя-бы в одной записи для всех . В задаче *классификации* есть множество всех классов, а в задаче восстановления зависимости – конкретные значения непрерывного целевого атрибута, т.е. Обозначим обучающую выборку через . Тогда для обучающей выборки существуют 2 ситуации:

1. *Все значения в принимают значение . Тогда ДР, построенное на этой выборке, состоит только из листа, помеченного значением .*
2. *Значения в принимают различные значения из . В таком случае необходимо разделить выборку . Для этого некоторым образом (см. ниже критерий расщепления) выбирается входной атрибут . Допустим, все значения этого атрибута в принадлежат множеству . Тогда выборка разделяется на подвыборки ; где Строится узел дерева, помеченный вопросом «каково значение атрибута ?». Далее для каждой подвыборки по данному правилу из п. 1-2 рекурсивно строится поддерево, связанное с данным узлом ребром, помеченным значением .*

Заметим, что алгоритм построения ДР является *нисходящим* – происходит от корня к листам. Четкое следование данному алгоритму влечет построение слишком *глубокого* ДР для достаточно представительных выборок. При этом эффективность решения задачи с помощью таких ДР может быть невысокой, поэтому существуют специальные *критерии останова* построения дерева, определяющие момент, когда дальнейшее построение дерева нецелесообразно. Рассмотрим самые популярные критерии останова [5]:

* *Ранняя остановка*. Определяет целесообразность разбиения узла. Данное разбиение может быть нецелесообразным, если в подвыборке, соответствующей данному узлу, число записей, представляющих класс существенно преобладает над числом записей, представляющих другие классы. Данный критерий позволяет существенно уменьшить время построения дерева, однако возникает риск снижения точности классификации.
* *Ограничение глубины дерева*. В этом случае построение заканчивается, если достигнута заданная глубина дерева.
* *Ограничение по размеру подвыборок*. При этом варианте ветвление продолжается до тех пор, пока текущему узлу сопоставлена выборка, содержащая не более чем заданное число элементов.

Процессу *построения* дерева предстоит выбор *критерия расщепления* дерева, а завершает процесс обучения *алгоритм отсечения ветвей* ДР. Таким образом, эти три последовательных действия составляют процесс обучения ДР.

Цель *критерия расщепления* состоит в нахождении такого условия (вопроса), которое разбивало бы множество , ассоциированное с текущим узлом, на подмножества так, чтобы они принадлежали одному классу (по значению атрибута ), или были максимально приближены к этому. В качестве такой проверки выбирается один из входных атрибутов , по которому осуществляется разбиение исходной выборки, после чего проверяется качество разбиения, по результату которого принимается решение, подходит данный атрибут для разбиения или нет. Рассмотрим 2 критерия качества разбиения:

* *Теоретико-информационный критерий.* Смысл данного критерия заключается в максимизации информации, полученной за счет разбиения множества . Для этого вводятся следующие понятия:

*Энтропией[[3]](#footnote-3)* множества называется величина:

*где – подмножество , во всех элементах которого ;*

*– число классов элементов обучающей выборки.*

*Условной энтропией* множествапо разбиению называется величина:

*где – подмножество , во всех элементах которого ;*

*– число всех возможных значений атрибута в выборке .*

*Информацией*, полученной из путем разбиения ее по называется:

Тогда данный критерий заключается в выборе такого атрибута , при разбиении по которому достигается максимум *информации*:

Часто вводится нормировочный коэффициент, определяющий количество информации, требуемое для разделения по текущему атрибуту:

В таком случае отыскание аргумента максимума от величины получается более эффективным.

* *Критерий индекса Гини.* В данном случае ищется как решение следующей задачи:

Величина называется *индексом Гини*.

Процесс *отсечения ветвей* происходит на этапе тестирования ДР. Вычисляется *ошибка распознавания -* отношение числа неправильно классифицированных объектов в *тестовой выборке* к общему ее объему. На основании этой величины происходит отсечение наиболее «ветвистых» областей ДР, но отсекаются только те ветви, удаление которых не влечет увеличение *ошибки распознавания*. Данный алгоритм чаще всего применяется как альтернатива использованию *критерия останова* при построении дерева.

Основными достоинствами ДР как метода ИАД являются простота в понимании и интерпретации структуры и алгоритма работы ДР, отсутствие требований на подготовку входных данных, явное содержание извлеченных из БД правил в своей структуре. К недостаткам относится то, что проблема построения оптимального дерева является NP-полной задачей, в связи с чем эффективность построенных деревьев может не соответствовать поставленным требованиям.

### Метод опорных векторов

*Метод опорных векторов* решает задачи *классификации, прогнозирования* и *восстановления зависимости*. МОВ является одним из методов *машинного обучения* [7]; изначально он был предназначен для решения задачи *бинарной классификации* (все рассматриваемые в рамках задачи объекты принадлежат одному из двух классов), однако были найдены способы использовать МОВ в классах задач, приведенных выше.

Итак, задача, решаемая МОВ, заключается в классификации множества объектов на 2 класса. Без ограничения общности будем полагать, что . Тогда графический смысл МОВ заключается в нахождении гиперплоскости[[4]](#footnote-4) в пространстве векторов , такой, что значения функции для из класса были отрицательны, а для – положительны (см. Рисунок 5). Здесь – вектор нормали,

В таких задачах существует 2 возможных случая: обучающая выборка либо является *линейно разделимой* (т.е. существует такая гиперплоскость), либо таковой не является. Сначала рассмотрим 1-й случай.

В силу несчетности пространства векторов существование хотя-бы одной разделяющей гиперплоскости влечет существование множества таких гиперплоскостей. В таком случае задача состоит в выборе оптимальной разделяющей гиперплоскости. Для этого рассматриваются т.н. *опорные вектора* – это те элементы множества из обучающей выборки, которые находятся на общей границе классов (на Рисунке 5 *опорные вектора* закрашены). Оптимальная разделяющая гиперплоскость максимизирует расстояние до всех опорных векторов. Таким образом, после нахождения и , соответствующих оптимальной гиперплоскости, МОВ считается обученным и построенная может быть применена для классификации новых объектов как часть решающей функции

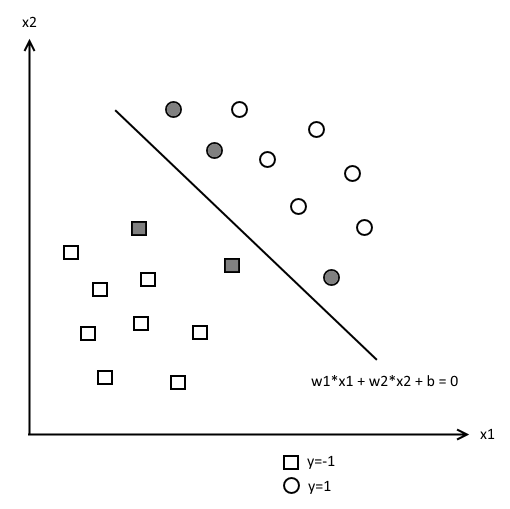


Рисунок . Пример классификации с помощью МОВ векторов из пространства

В случае *линейной неразделимости* обучающей выборки для нахождения оптимальной гиперплоскости применяются два различных подхода:

1. Вводят группу дополнительных переменных , характеризующих ошибку классификации векторов . Здесь – число элементов в обучающей выборке. Т.е. характеризует ошибку классификации -го вектора из обучающей выборки. В таком случае для нахождения оптимальной разделяющей гиперплоскости решается одновременно задача максимизации расстояния до опорных векторов и минимизации суммарной ошибки . Здесь – константа, определяющая «существенность» ошибки.
2. Исходная задача преобразуется в такую, где пространство векторов имеет большую размерность и возможно осуществление линейной разделимости. Такое преобразование осуществляется с помощью выбора и применения т.н. *оператора ядра* . Далее задача решается в пространстве , после чего с помощью обратного преобразования получается результат для исходной задачи.

Примерами *операторов ядра* могут служить:

* 1. Полиномиальное однородное:
  2. Полиномиальное неоднородное:
  3. Радиально-базисная функция:

В задачах *восстановления зависимости* в качестве решающей функции берут

в остальном алгоритм МОВ не меняется.

### Метод ближайших соседей

Данный метод решает задачи *классификации*, *восстановления зависимости* и *прогнозирования* и представляет собой один из методов *машинного обучения.* Для обучения метода кроме *обучающей выборки* необходимо задать *функцию расстояния* , определенную в пространстве векторов .

В случае решения задачи *классификации*, определение класса вектора согласно МБС осуществляется следующим образом [5]:

1. Для всех векторов из обучающей выборки определяется расстояние от данного вектора до классифицируемого:
2. Определяется число , соответствующее количеству векторов, расстояние от вектора до которых наименьшее среди всех . Из-за необходимости выбора данного параметра МБС также называют *метод ближайших соседей*.
3. Вектор относится МБС к тому классу, который чаще всего является классом векторов, соответствующих минимальным расстояниям до данного вектора.

Рассмотрим пример, представленный на Рисунке 6. Имеется обучающая выборка из 20 элементов, являющихся элементами 3-х классов («кружков», «квадратов» и «треугольников»). Необходимо классифицировать элемент, отмеченный «звездочкой». Результат работы МБС зависит от выбора значения параметра . Так, при рассматриваемый элемент будет классифицирован как «кружок», т.к. ближайшим его соседом является элемент из данного класса. При или классифицировать объект невозможно, т.к. в этих случаях ни один из классов не представлен большим числом элементов, чем другие. А при «звёздочке» будет присвоен класс «треугольник».

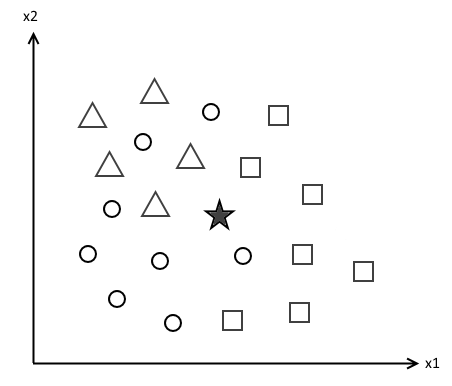


Рисунок . Пример множества для классификации с помощью МБС

Таким образом, выбор параметра существенно влияет на решение, выдаваемое МБС. Данный параметр вводился для сглаживания влияния выбросов и шумов в обучающей выборке на решение, ведь вектора, находящиеся ближе всего к выбросу, будут классифицироваться неправильно при . Однако значение , где число элементов в обучающей выборке, тоже неудачно, т.к. объект будет классифицирован тем значением, которое преобладает в обучающей выборке. Поэтому выбор оптимального значения является отдельной важной задачей. Это значение можно определить следующим образом: для каждого объекта из обучающей выборки и для каждого проверить, правильно ли данный объект классифицируется по своим ближайшим соседям. Далее следует выбрать с наименьшим числом ошибок.

В задачах *восстановления зависимости* значение параметра определяется как среднее арифметическое между значениями ближайших соседей. Таким образом, неопределенности, которая возникала в задачах *классификации*, здесь нет.

К достоинствам МБС прежде всего следует отнести его простоту, а главным недостатком метода является необходимость запоминания *обучающей выборки*. Для того, чтобы не использовать всю обучающую выборку, следует перед обучением метода провести её фильтрацию: оставить в ней только характерные элементы, действительно влияющие на работу метода.

### Нейронные сети

*Нейронные сети* являются одним из самых мощных методов ИИ, применяющихся в задачах ИАД. Как ясно из самого названия метода, преобразование информации в ИНС происходит по образу и подобию процессов, происходящих в мозге человека [8, 9, 10]. ИНС используются в задачах *распознавания*, *классификации*, *кластеризации, восстановления зависимости*, *прогнозирования,* отчасти в задачах *ассоциации* и *анализа последовательностей*. Таким образов ИНС может применяться при решении практически любой задачи ИАД. При этом каждый тип ИНС нацелен на решение задачи конкретного класса. В зависимости от класса поставленной задачи данный тип ИНС принадлежит к алгоритмам ИАД, *обучающимся с учителем*, или к методам, *обучающимся без учителя*. Одним из главных достоинств метода ИНС является особенность архитектуры, позволяющая выполнять вычисления, происходящие во время решения задачи данным методом, параллельно.

В общем случае ИНС представляет собой совокупность взаимосвязанных вычислительных единиц сети – *нейронов*. Для нейрона существуют 2 типа связи: *входная* и *выходная*. Каждый нейрон принимает сигналы от связанных с ним по входной связи нейронов, и на основании величины и характера данных сигналов формирует свой, который идет на вход тем нейронам, с которыми связан данный по выходной связи. Таким образом сигнал распространяется по сети. Группы нейронов, одновременно выполняющих вычисление своего сигнала, называются *слоями* ИНС. Каждая ИНС должна иметь *входной* и *выходной* слои нейронов. Каждый нейрон входного слоя должен иметь входную связь, по которой исследователь может передать требуемое значение. Соответственно, каждый выходной нейрон должен иметь выходную связь, с которой исследователь может считать значение сигнала данного нейрона. Таким образом, на входной слой ИНС передается вектор входных атрибутов задачи, после чего с выходного слоя считывается вектор. На рисунке 7 представлен пример ИНС, содержащей 8 нейронов. Входной слой содержит нейроны с номерами 1, 2 и 3. Выходной слой состоит из единственного нейрона №8. Нейроны с номерами 4-7 составляют единственный *скрытый* слой. Стрелками указаны входные связи нейронов. Так, нейрон №4 имеет входные связи с нейронами №1 и №3 и выходную связь с нейроном №8.

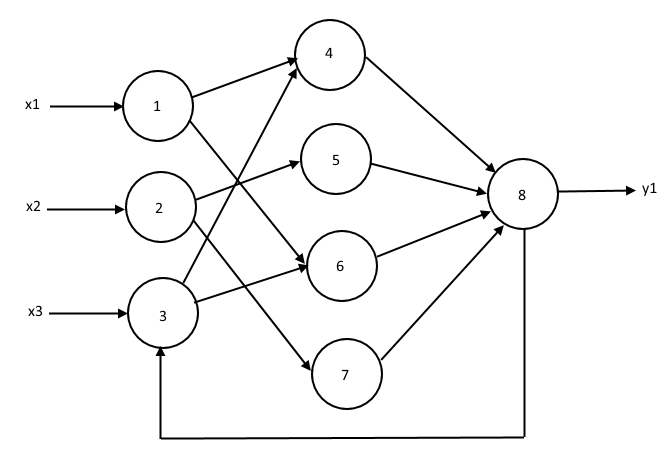


Рисунок . Пример ИНС

Заметим, что каждый нейрон входного и выходного слоя содержит единственную связь «с исследователем». В связи с этим число нейронов в данных слоях должно совпадать с размерностью векторов и соответственно для решения поставленной задачи. Поэтому ИНС строится и обучается только под решение одной, конкретной задачи. Для решения другой задачи в общем случае требуется построение и обучение новой ИНС.

Также рассматриваемая ИНС содержит т.н. *обратную связь* – нейрон №8 более позднего слоя является одним из входов нейрона №3 – более раннего слоя. ИНС, в которых присутствуют обратные связи, называются *рекуррентными*.

Теперь рассмотрим структуру вычислительной единицы ИНС – нейрона (см. Рисунок 8).

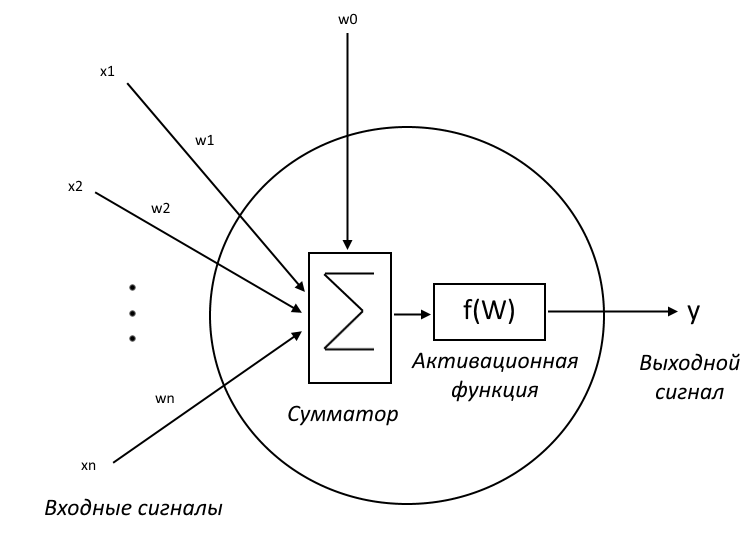


Рисунок . Структура искусственного нейрона

Каждый нейрон имеет входов, где – число нейронов, связанных с данным посредством выходной связи. По входам данному нейрону передаются входные сигналы . Эти не являются входными сигналами ИНС (т.е. это не компоненты вектора ). Они являются выходными сигналами нейронов, связанных с данным. Также каждая входная связь имеет вещественный *вес* , определяющий силу связи. Существует еще и вес , определяющий степень чувствительности нейрона. Веса нейронов ИНС являются скрытыми параметрами метода, они недоступны исследователю. Обучение ИНС заключается в подборе значений данных параметров. Работа нейрона заключается в формировании своего выходного сигнала и отправлении его на следующий слой ИНС. Этот процесс осуществляется в несколько стадий [8, 9, 10]:

1. Вычисление *взвешенной суммы* входных сигналов:
2. Формирование выходного сигнала с помощью применения *активационной функции* к *взвешенной сумме*:
3. Передача выходного сигнала на следующий слой ИНС.

*Активационная функция* отражает зависимость значения выходного сигнала – «степени возбуждения» нейрона – от суммарного воздействия нейронов предыдущего слоя, связанных с данным. Чаще всего используются следующие АФ [8, 9, 10]:

* Линейная АФ:
* Пороговая АФ:
* Сигмоидальная АФ:

*где – параметр, характеризующий крутизну наклона функции.*

* Гиперболический тангенс:
* Радиально-базисная функция:

*где – параметры.*

Вместо использования АФ нейрон может генерировать выходной сигнал с использованием генератора случайных чисел и плотности вероятности возникновения сигнала. В этом случае нейрон называется *стохастическим*, а при использовании АФ – *детерминированным* [9, 10].

Таким образом, ИНС можно классифицировать по следующим признакам:

* Характер распространения сигнала (прямого распространения или рекуррентные).
* Тип АФ, применяемых в нейронах.
* Тип входных и выходных сигналов ИНС: бинарные, действительные.
* Способ обучения сети: *с учителем* или *самоорганизующиеся.*
* Способ вычисления выходных сигналов: *детерминированные* или *стохастические*.
* Топология сети – закономерность структуры ИНС, проявляющаяся в связях сети, расположении нейронов, иногда их количестве, и т.д.

Рассмотрим самые популярные топологии ИНС.

#### Персептрон

Одной из первых изобретенных топологий является *персептрон Розенблатта* [8]. Он представляет собой *полносвязную* ИНС *прямого распространения* – т.е. каждый нейрон предыдущего слоя данной сети связан выходной связью с каждым нейроном последующего слоя (см. Рисунок 9).

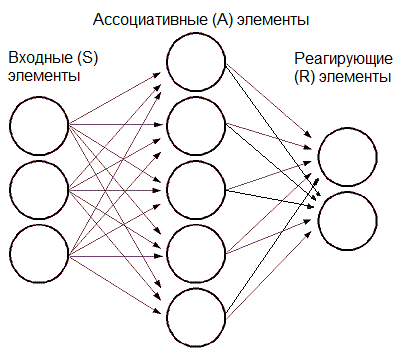


Рисунок . Двуслойный персептрон

Входной слой нейронов по историческим причинам называется S-слоем, скрытые слои называются A-слоями, а выходной – R-слоем. Число слоев, фигурирующее в названии персептрона, соответствует числу A и R слоев. Так, *однослойный персептрон* содержит по одному S и R слою, связанных друг с другом. Принцип работы однослойного персептрона аналогичен МОВ: данная ИНС также ищет гиперплоскость, разделяющую обучающую выборку на классы. Таким образом, однослойный персептрон не способен решить линейно неразделимые задачи – например, задачу XOR. Однако уже *двуслойный персептрон* в состоянии решать такие задачи.

*Персептрон* решает задачи *классификации*, *восстановления зависимости*, *прогнозирования*, обучается *с учителем*. Наиболее известным методом обучения персептрона является *метод обратного распространения ошибки*.

МОРО основан на минимизации *функции ошибки* на обучающих данных посредством *градиентного спуска*. Таким образом, для работы МОРО необходимо, чтобы АФ всех нейронов персептрона имели первую производную. Примером *функции ошибки* может служить сумма квадратов расстояний от выходных сигналов ИНС до их истинных значений:

*где – выходное значение -го нейрона выходного слоя; - истинное значение данного выхода*

МОРО вычисляет значение необходимой коррекции весов каждого нейрона начиная от выходного слоя ко входному. Опишем алгоритм МОРО.

Пусть обучается персептрон с слоями и дифференцируемой АФ нейронов. Обозначим через выходное значение -го нейрона -го слоя. Заметим, что . Пусть обучающая выборка состоит из записей . Обозначим через значение веса связи между -м нейроном слоя и -м нейроном -го слоя. Необходимо определить количество итераций метода – число «прогонов» обучающей выборки. Обозначим число итераций через . Тогда МОРО состоит в последовательном выполнении следующих действий [8]:

1. Инициализировать все веса случайными значениями, мало отклоняющимися от нуля.
2. Повторить раз:
   1. Для всех от до :
      1. Подать на вход сети вектор , подсчитать выходы каждого узла.
      2. Для последнего слоя подсчитать вклад каждого нейрона в общую ошибку:
      3. Для каждого слоя от до :
         1. Для каждого -го нейрона текущего слоя вычислить:
      4. Для каждого слоя от до , для каждой связи между соседними слоями вычислить:
3. Выдать значения .

Здесь есть параметр, влияющий на скорость обучения.

#### Сеть Кохонена

Рассмотрим *сеть Кохонена* – ИНС, решающую задачи *кластеризации*, и, соответственно, *обучающуюся без учител*я (самоорганизующуюся). Данная ИНС имеет 2 слоя: входной – размерности (число атрибутов в векторах ), и т.н. *слой Кохонена* из нейронов. Здесь – число кластеров, на которые необходимо разделить данные рассматриваемой задачи. Каждый нейрон входного слоя связан со всеми нейронами слоя Кохонена (см. Рисунок 10). Последний является также и выходным слоем. Т.о. *сеть Кохонена* является преобразователем векторов пространства размерности в вектора, как правило, меньшей размерности .

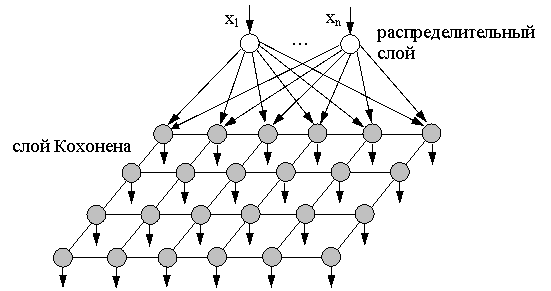


Рисунок . Самоорганизующаяся карта Кохонена

Приведем алгоритм обучения *сети Кохонена* [10]:

1. *Инициализация сети*. Выбираются первоначальные значения весов. Наиболее распространены способы инициализации случайными числами или присваивание вектору весов, соответствующего конкретному нейрону из слоя Кохонена, случайного вектора из обучающей выборки.
2. *Циклическое обучение***.**
   1. Пусть номер текущей итерации. Производится произвольный выбор обучающего вектора из выборки.
   2. Находится расстояние между этим вектором и каждым из векторов весов нейронов слоя Кохонена. Выбирается тот нейрон, расстояние между вектором весов которого и текущим обучающим вектором наименьшее. Если таких нейронов несколько, случайно выбрать из них один. Веса сети обозначим через где . Тогда на данном этапе выбирается . Метрика должна быть выбрана заранее. Нейрон с номером объявляется «победителем», дальнейшие шаги обучения зависят от выбора этого нейрона.
   3. Вокруг нейрона-победителя образуется *радиус обучения*. Он определяет сколько нейронов кроме нейрона-победителя участвуют в обучении (т.е. корректируют свои веса) на данной итерации. Под радиусом в данном случае подразумевается расстояние в пространстве векторов весов нейронов. Т. е. любой нейрон, расстояние от вектора весов которого до вектора весов нейрона-победителя менее радиуса обучения, участвует в коррекции весов на данной итерации. Радиус обучения максимален на первой итерации и уменьшается с увеличением числа итераций таким образом, что в конце обучения корректирует свои веса только нейрон победитель. Веса нейрона-победителя и всех нейронов, лежащих в пределах его радиуса обучения, подвергаются обучению по следующему правилу:

*где - коэффициент скорости обучения -го нейрона из радиуса обучения в -м цикле обучения.*

Веса нейронов, находящихся за пределами радиуса обучения, не изменяются. Коэффициент скорости обучения состоит из *функции соседства* и *функции скорости обучения* :

В качестве *функции соседства* применяется или пороговая

или функция Гаусса

При этом является убывающей от номера цикла обучения функцией.

В качестве *функции скорости обучения* также следует брать убывающую от номера цикла обучения. Чаще всего берут

*где и – константы.*

Применение такой функции приводит к тому, что все векторы из обучающей выборки вносят примерно равный вклад в результат обучения.

* 1. Рассчитывается ошибка карты:

*где – объем обучающей выборки.*

При достижении некоторого минимального значения цикл можно прекратить и считать сеть обученной.

#### Сеть Хопфилда

*Нейронная сеть Хопфилда* состоит из нейронов, каждый из которых имеет один вход. Нейроны соединены друг с другом рекуррентными связями по типу: 1-й нейрон соединен со 2-м, 3-м, … -м; 2-й соединен с 1-м, 3-м, …, -м и т.д. (см. Рисунок 11).

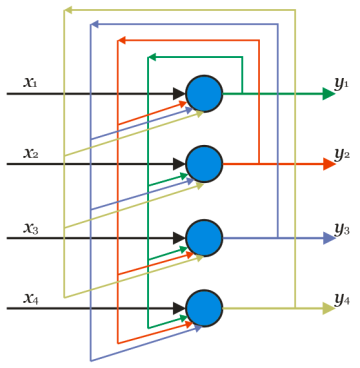


Рисунок . Сеть Хопфилда

Кроме этого каждый нейрон имеет один выход. Поэтому размерности входного и выходного пространства сети одинаковы. Каждый из нейронов имеет пороговую бинарную функцию активации. Матрица всех синаптических весов

в сети Хопфилда является симметричной, диагональные элементы равны нулю (т.е. нейрон не может воздействовать сам на себя).

Данная сеть применяется при решении задач *ассоциации* и *анализа последовательностей* [10]. Она запоминает определенные элементы входного множества при обучении, после чего пытается «приблизить», соотнести новые входные векторы к заученным, т.е. восстановить ранее запомненные знания новыми, но зашумленными. Данная сеть генерирует не один выходной импульс при своем возбуждении, а целый ряд. Это происходит за счет рекурсивности связей нейронов, когда выходное значение одной итерации идет не только на выход из сети, но и заново на входы нейронов. Таким образом, в процессе генерации ряда выходных импульсов сеть ищет ассоциации к полученному входному значению среди ранее запомненных. В итоге процесс сойдется, т.е. сеть с некоторого момента будет генерировать один и тот же сигнал, или 2 чередующихся. Эти сигналы, полученные после получения устойчивости сети, являются ее ответами на входной вектор.

Алгоритм обучения сетей Хопфилда отличается от всех методов обучения, расмотренных ранее, тем, что вместо последовательного приближения к нужному состоянию с вычислением ошибок, все коэффициенты матрицы рассчитываются по одной формуле, за один цикл, после чего сеть сразу готова к работе. Вычисление значений весов основано на том, что для всех запомненных образов матрица весов должна удовлетворять уравнению

,

т.к. именно при этом условии состояния сети будут устойчивы – попав в состояние , сеть в нем так и останется. Стоит отметить, что все входные векторы сети, как обучающие, так и рабочие, должны быть представлены в *бинарном виде*.

Расчет весов производится по следующей формуле:

*где – -я компонента -го обучающего вектора, – номер обучающего вектора в выборке.*

При этом для достижения во время работы сети т.н. *статического аттрактора* (полной устойчивости состояния сети, без колебаний) необходимо и достаточно, чтобы матрица была симметрична, по диагонали были нули, а также чтобы сеть работала в *асинхронном режиме* – импульсы между нейронами передавались с определенной для каждого нейрона (но неизменной) задержкой . При обычном, *синхронном режиме* (когда ), возможно появление *динамического аттрактора* – колебания сети между несколькими ответами.

Кроме этого сеть обладает достаточно небольшим объемом памяти [10]:

Попытка записи большего числа образов приводит к тому, что сеть перестает их распознавать.

Так же могут случиться ситуации, когда достижение устойчивого состояния сети не гарантирует правильности ответа. Т.е. когда сеть сходится к ложным аттракторам, «химерам». Как правило, «химеры» склеены из фрагментов различных образов.

### Алгоритм Apriopi

Данный алгоритм решает задачи *ассоциации*, не обучается. Идея алгоритма Apriori заключается в выявлении наиболее часто встречающихся признаков в данных и сопоставлении этих признаков друг другу [5]. Если отношение частоты встречаемости обоих этих признаков к частоте встречаемости одного из них превосходит заранее заданную константу, то эта группа признаков объявляется ассоциативным правилом. Стоит отметить, что для использования данного алгоритма выборка, сформированная на основе БД для решения задачи, должна быть преобразована в бинарный формат, т.е. все значения атрибутов векторов должны принадлежать множеству .

Приведем термины, необходимые для описания метода. Пусть имеется выборка, состоящая из векторов , где . При этом существует атрибутов, которые для каждого принимают конкретное значение. Введем дополнительное обозначение для множества атрибутов (признаков): , где . Каждому набору признаков поставим в соответствие предикат:

Если , значит признаки набора *совместно встречаются* у объекта . *Частотой встречаемости* набора в выборке из элементов называется функция

Набор называется *часто встречающимся*, если . Параметр называется *минимальной поддержкой*.

Пара непересекающихся наборов называется *ассоциативным правилом* если выполняются 2 условия:

Величина называется *значимостью правила*, параметр называется *минимальной значимостью*.

То есть, в ассоциативном правиле наборы совместно часто встречаются, и в значительной доле случаев (не менее ) если встречается набор , то встречается также и . Алгоритм Apriori ищет ассоциативные правила на основе свойства *монотонности* – для любого набора и его подмножества справедливо . Т.е. добавление признака в набор может привести только к снижению частоты его встречаемости. Следовательно, чтобы набор был часто встречающимся, необходимо (но не достаточно), чтобы все его подмножества также часто встречались. Данный алгоритм имеет следующую структуру:

1. Создание множества всех часто встречающихся признаков:
2. Для всех
   1. Создать множество всех часто встречающихся наборов мощности :
   2. Если , выйти из цикла.
3. Для всех выполнить .
4. Вернуть – список всех найденных ассоциативных правил.

1. Для всех
   1. .
   2. Если тогда добавить ассоциативное правило в список результатов .
   3. Если тогда выполнить

### Генетические алгоритмы

*Генетический алгоритм* представляет собой механизм поиска, построенный на эвристических знаниях о естественном отборе в природе [9]. Используется в основном в задачах оптимизации (нахождения минимального, максимального значения функции) и моделирования каких-либо процессов, машинном обучении. Применительно к методам ИАД используется при подборе параметров методов, т.е. в *обучении с учителем*. Основная идея данного алгоритма заключается в построении оптимальной *популяции* – такого множества объектов, называемых *хромосомами*, которые были бы наиболее приспособлены к окружающей среде. Это построение достигается за счет размножения, мутации и селекции *генов*. Наиболее приспособленная популяция, или самый приспособленный ее ген (в зависимости от постановки задачи) считается ответом ГА. Заметим, что ГА является полностью эвристическим алгоритмом, пока еще не доказано, что он будет сходиться для любой поставленной задачи.

Формально алгоритм ГА может быть представлен следующим образом [9, 11]:

1. *Сформировать начальную популяцию хромосом.*
2. *Вычислить приспособленности каждой хромосомы.*
3. *Пока приспособленность популяции не достигла определенного значения:*
   1. *Выбрать пару хромосом-родителей.*
   2. *Провести скрещивание родителей.*
   3. *Обеспечить мутацию некоторых из полученных хромосом.*
   4. *Провести отбор нового поколения.*
   5. *Вычислить приспособленность каждой хромосомы.*
4. *Вернуть оптимальную популяцию.*

Теперь рассмотрим этот алгоритм более подробно. Изначально необходимо выбрать структуру представления хромосом. Стандартными являются представления в виде векторов из действительных и бинарных чисел. Хромосома является важнейшей частью ГА, главным действующим элементом, поэтому выбор ее представления должен быть обоснован. Естественно, под понятием хромосомы скрывается главный структурный элемент задачи, которую необходимо решить ГА. Так, например, если задача заключается в нахождении экстремума некоторой функции, определенной на n-мерном пространстве, то под хромосомами понимают элементы данного пространства, а под приспособленностью хромосомы – значение данной функции на соответствующем элементе. Если требуется найти минимум функции, то хромосома считается тем более приспособленной, чем меньше число ее приспособленности, т.е. чем меньше значение функции на данном элементе.

Если же задача, поставленная перед ГА, заключается в нахождении оптимальных классифицирующих правил по типу «Если атрибут имеет значение , а атрибут имеет значение , то объект следует отнести к классу », то хромосомой является одно правило такого типа. В данном случае хромосома является вектором , а ее приспособленность можно определить как отношение количества объектов из обучающей выборки, для которых это правило истинно, к общему количеству объектов выборки. Тогда наилучшим классификатором будет самая приспособленная популяция.

Начальная популяция может не слишком соответствовать действительности, т.е. быть не особенно эффективной – на этот счет беспокоиться не стоит, т.к. ГА рано или поздно выведет оптимальную популяцию. От качества начальной популяции зависит только время работы ГА. Функция приспособленности хромосом разрабатывается исходя из особенностей решаемой задачи, а также исходя из требуемого результата.

Условие останова цикла ГА может быть также различным – если необходимо нахождение одного оптимального значения, то стоит под этим условием взять функцию максимума от приспособленности хромосом популяции, а если важна «общая картина» - то, например, средняя приспособленность хромосом текущей популяции. Также условием останова может служить определенное максимально допустимое количество пройденных алгоритмом итераций или отведенное на его выполнение время.

Существует несколько подходов к выбору *родительской пары*. Наиболее распространены следующие методы [9, 11]:

* *Панмиксия*. Каждому члену популяции ставится в соответствие случайное целое число из , где – количество особей в популяции. Данные числа интерпретируются как номера особей-партнеров. Если хромосоме поставлен в соответствие ее же номер, значит, она не участвует в скрещивании. Некоторые особи могут принимать участие в скрещивании несколько раз (Например, особи №1 поставлена в соответствие особь №2, а ей в свою очередь поставлена в соответствие особь №7).
* *Инбридинг***.** Первый родитель выбирается случайным образом, а вторым является особь, ближайшая к первому. Метрика зависит от представления хромосом и особенностей решаемой задачи (Например, для действительных векторов можно взять евклидовую метрику, для бинарных – хэммингову, и т.д.).
* *Аутбридинг***.** Брачные пары формируются из максимально далеких особей.
* *Селекция***.** Заключается в том, что родителями могут стать только те особи, значение приспособленности которых не меньше пороговой величины, например, среднего значения приспособленности по популяции. Наиболее известные способы селекции – турнирный и рулеточный отборы.
* *Турнирный отбор***.** Из популяции, содержащей особей, выбираются случайным образом особей, и лучшая из них записывается в промежуточный массив. Эта операция проводится раз, после чего особи из промежуточного массива используются для скрещивания, также случайно.
* *Рулеточный отбор***.** Каждой особи сопоставляется вероятность ее попадания в новую популяцию:

где – приспособленность -й особи.

После этого проводится отбор пар по данным вероятностям. Остальные методы отбора пар являются модификациями представленных выше.

Сразу после выполнения операции отбора родителей производится их *рекомбинация* с целью получения новых особей-потомков. Смысл рекомбинации заключается в том, что созданные потомки должны наследовать генную информацию от обоих родителей. Различают *дискретную рекомбинацию* и *кроссовер*.

*Дискретная рекомбинация* соответствует обмену генами между особями. Потомки наследуют гены либо одного, либо второго родителя с равной вероятностью. Дискретная рекомбинация применима для любого типа генов (двоичные, вещественные и символьные). Различают 2 типа дискретной рекомбинации:

* *Промежуточная рекомбинация* применима только к вещественным переменным. Гены потомков создаются по правилу

*где множитель -случайное число на отрезке параметр. Для каждого гена создаваемого потомка выбирается отдельный множитель .*

* *Линейная рекомбинация* отличается от промежуточной тем, что множитель выбирается для каждого потомка один раз.

Рекомбинацию бинарных строк принято называть *кроссовером* (кроссинговером, скрещиванием). *Одноточечный кроссовер* подразумевает выбор точки разреза в хромосомах и обмен данными частями. *Двуточечный кроссовер* – соответственно выбор 2-х точек, между которыми формируется разрез (см. Таблица 1).

Таблица .Пример применения операции кроссовер

|  |  |  |  |
| --- | --- | --- | --- |
| ***Одноточечный кроссовер*** | | | |
| Родитель 1 | Родитель 2 | Потомок 1 | Потомок 2 |
| *0110|01* | *1100|10* | *011010* | *110001* |
| ***Двуточечный кроссовер*** | | | |
| *1100[1001]01* | *0100[0001]11* | *1100000101* | *0100100111* |

Также существуют и другие методы скрещивания, являющиеся модификациями или следствиями данных методов.

После процесса рекомбинации происходят *мутации* случайно выбранных хромосом. Это необходимо для того, чтобы популяция не скапливалась в локальном экстремуме функции приспособленности. Мутации могут происходить как в одном отдельно взятом гене, так и в нескольких. Вероятность мутации (как правило, ) может являться или фиксированным случайным числом на отрезке , или функцией от какой-либо характеристики решаемой задачи.

Для создания новой популяции используются различные методы *отбора* особей. Так, в новую популяцию могут попасть не только выведенные на данном этапе скрещивания особи, но и их родители. Может производиться «элитарный» отбор, где в новую популяцию попадают только те особи, приспособленность которых лучше остальных. Или с помощью определенной функции, рассчитывающей вероятность попадания определенной особи в новое поколение.

## Этапы решения задач методами ИАД

Итак, выше были рассмотрены основные методы, с помощью которых решаются задачи ИАД. Однако сам процесс решения поставленной задачи в рамках области ИАД не ограничивается простым применением одного из этих методов. Выделяются следующие этапы в процессе ИАД [1, 5, 12]:

1. *Постановка задачи ИАД*. На данном этапе формулируется цель предстоящего анализа, оценивается целесообразность его проведения. Определяется тип задачи ИАД на основании сформулированной цели. Определяются структуры хранения данных, необходимых для проведения анализа. На данном этапе, как правило, исследователь работает совместно со специалистами в предметной области, которую необходимо проанализировать.
2. *Сбор данных*. Производится сбор информации по предметной области для последующего анализа.
3. *Подготовка данных*. Осуществляется предварительная обработка данных:
   1. *Очистка* – исключение противоречий, выбросов и случайных шумов из исходных данных
   2. *Интеграция* – объединение данных из нескольких источников в одно хранилище
   3. *Преобразование данных* – данные преобразуются в форму, подходящую для анализа. Применяется агрегация данных, дискретизация атрибутов, сжатие данных и сокращение размерности.
   4. *Формирование выборок данных* – из общего хранилища данных выбираются множества, составляющие генеральную, тестовые и обучающие выборки.
4. *Выбор модели*. На данном этапе производится выбор метода ИАД, который будет решать поставленную задачу. Также здесь выбираются методы обучения, подбираются их параметры. Выбирается модель обучения метода: на всей генеральной выборке, с использованием нескольких обучающих выборок, методом кросс-валидации и т.п.
5. *Подбор параметров модели.* Производится обучение выбранной модели на подготовленных данных.
6. *Анализ качества обучения*. Производится анализ качества решения задачи обученным методом посредством использования тестовых выборок и тестовых испытаний. В случае неудовлетворительных результатов осуществляется переход либо на п. 4, либо на п. 5.
7. *Анализ выявленных закономерностей*. Производится использование модели в решении поставленной задачи. При фиксации неудовлетворительных результатов работы метода осуществляется переход на п. 1, п. 4 или п. 5.

Процесс ИАД цикличен, причем период выполнения «итерации» может составлять от нескольких месяцев до нескольких лет. Самыми трудоемкими этапами ИАД являются *сбор*, *подготовка данных* и *подбор параметров модели*.

## Обзор программных систем в области ИАД

В данной главе проводится обзор некоторых систем из области ИАД. Стоит отметить, что большинство коммерческих программных комплексов имеют довольно большую цену, а системы с открытым кодом в большинстве своем представляют собой подключаемые библиотеки, требующие самостоятельного написания пользователем интерфейсной части. Условно все системы, работающие в области ИАД, можно разделить на два класса: использующие один метод решения задачи и многофункциональные. Первые отличаются более простым и удобным интерфейсом, спроектированным специально для использования включенного в систему метода и некоторыми дополнительными возможностями по анализу эффективности решения этим методом. Однако использование таких систем приходится комбинировать, чтобы найти наиболее эффективный метод ИАД для решения конкретной задачи. Системы второго класса позволяют решать задачу несколькими возможными способами, однако удобного интерфейса сравнения эффективности методов в них нет. Кроме этого, данные системы имеют довольно сложный интерфейс, для работы с которым необходима предварительная подготовка.

### Oracle Data Mining

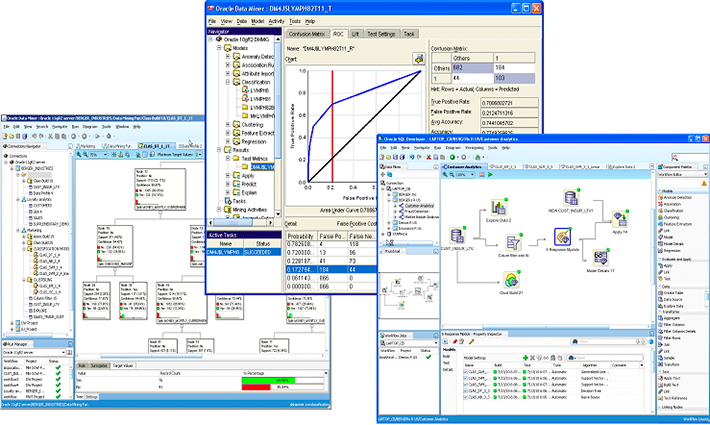


Рисунок 12. Интерфейс Oracle Data Mining

Данная система, разработанная компанией Oracle, позволяет использовать функциональности ИАД с помощью встроенных в Oracle Database – БД системы – SQL функций. Версия Data Mining 12с поддерживает следующие алгоритмы:

* Метод опорных векторов
* Деревья решений
* Метод ближайших соседей
* Алгоритм Apriopi

Кроме алгоритмов решения, в состав Oracle Data Mining входят средства подготовки данных, оценки результатов, применения моделей к новым наборам данных. Система распространяется как часть комплекса Oracle Advanced Analytics (€399.00).

### DTREG

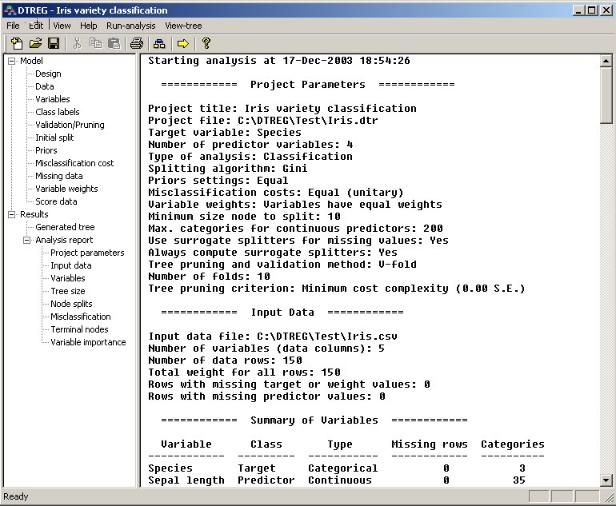


Рисунок . Интерфейс DTREG

Данная система предназначена для решения предсказательных задач. Среди используемых методов содержит:

* Многослойный персептрон
* ИНС с радиально-базисной АФ
* Метод опорных векторов
* Деревья и леса решений
* Метод ближайших соседей

Данная система является коммерческой, стоимость зависит от версии, количества включенных методов решения и т.д. (от $1000.00).

### STATISTICA Data Miner

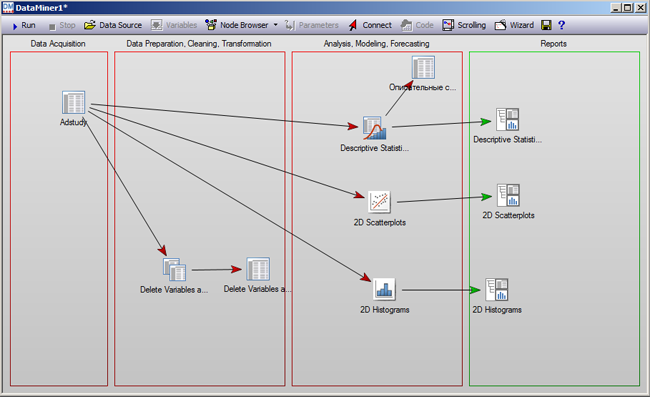


Рисунок 14. Интерфейс STATISTICA Data Miner

Данная система, разработанная компанией StatSoft Inc., позволяет провести полный цикл работ в области ИАД, от сбора и подготовки данных до поиска ценной информации.

STATISTICA Data Miner включает следующие методы решения задач:

* Многослойные персептроны
* ИНС с радиально-базисными АФ
* Самоорганизующиеся карты Кохонена
* Деревья решений
* Случайные леса
* Метод Байесовского вывода
* Метод опорных векторов
* Метод ближайших соседей

Кроме этого, система содержит возможности построения пользовательских топологий ИНС. Данная система является коммерческой.

### ORANGE DATA MINING

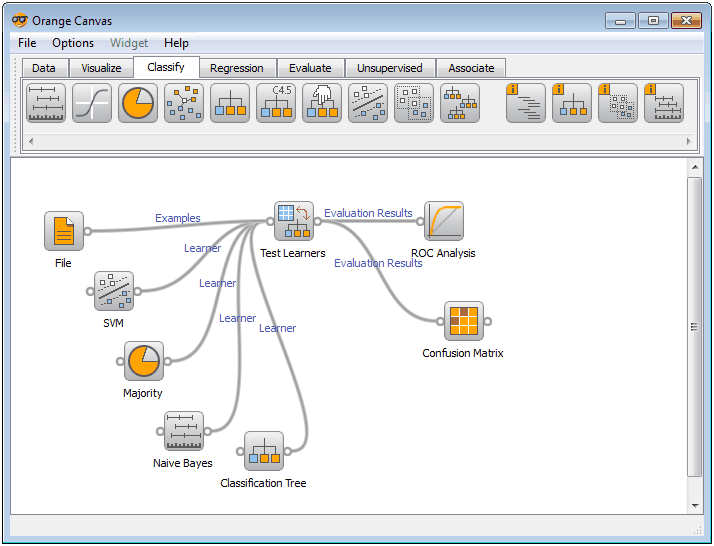


Рисунок 15. Интерфейс ORANGE DATA MINING

Данная система с открытым исходным кодом позволяет решать задачи классификации, восстановления регрессии, ассоциации, а также визуализировать полученные результаты благодаря мощным интерфейсам отображения информации. Содержит следующие методы решения задач:

* Метод опорных векторов
* Деревья решений
* Случайный лес
* Алгоритм Apriori

### NeuroShell 2

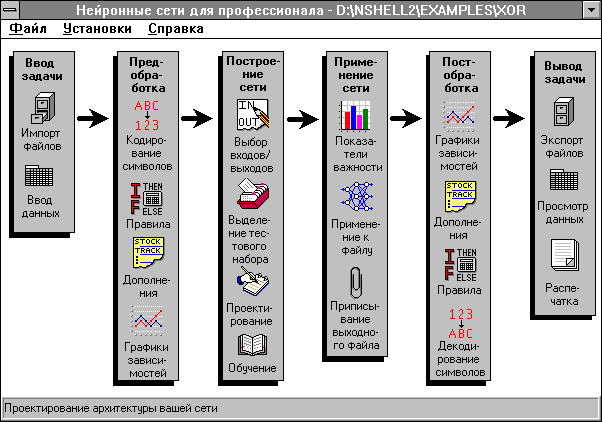


Рисунок . Интерфейс NeuroShell 2

Данный проект позволяет решать задачи ИАД с помощью ИНС. В нем реализованы следующие топологии:

* Персептрон
* Общая сеть «с обратным распространением ошибки» (связанными являются только соседние слои нейронов)
* Сети Кохонена
* ИНС с вероятностными нейронами

Также данный проект содержит некоторые дополнительные возможности по организации ИНС, такие как анализ важности входных переменных. Данная система является коммерческой (стоимость от $1500).

# Постановка задачи выпускной квалификационной работы

Целью данной работы является разработка *инструментальной системы интеллектуального анализа данных* (в команде), позволяющей осуществить этапы выбора модели, обучения, анализа качества и решения поставленной задачи ИАД (см. пункты 4-7 выше). Считается, что данные, используемые при работе ИСИАД над конкретной задачей, уже являются подготовленными (т.е. этапы 1-3 ИАД выполнены). ИСИАД должна использоваться при решении задач ИАД, которые позволяют использовать *обучение с учителем*; должна давать возможность решать задачи с помощью ИНС (в качестве топологии берется *персептрон*) и ДР. Должна быть предусмотрена возможность выбора различных методов обучения и методов ИАД для решения поставленной задачи. Предусматривается возможность сравнения результатов работы различных методов ИАД, а также использование различных методов обучения для одного метода ИАД. Известные системы ИАД имеют фиксированный набор методов, который менять нельзя. В связи с этим, ИСИАД должна позволять:

* провести вычислительный эксперимент по сравнению эффективности применения методов и алгоритмов, реализованных в библиотеке системы
* в случае неудовлетворительных результатов провести исследования по повышению эффективности методов
* добавить полученный в результате исследования метод в систему.

Индивидуальная задача автора данной работы заключалась в реализации нейронных сетей как метода ИАД, представленного в ИСИАД. Необходимо разработать и реализовать библиотеку классов ИНС и часть общей БД ИСИАД, хранящую информацию о ИНС.

# ИСИАД. Общая архитектура

Данный проект разрабатывался группой, в которую входил автор данной работы. Каждый разработчик создавал определенный модуль системы; поэтому для координации усилий и упрощения работы совместно были сформулированы общая архитектура проекта, термины и определения, применяемые при описании работы ИСИАД. В данной главе приводятся эти общие положения.

## Термины и определения

Введем следующее соглашение, связанное с обозначениями: для всех элементов будем указывать нижним индексом *принадлежность* данного элемента некоторому множеству, а верхним индексом – *связь* с некоторым множеством. Все имена индексов примем унифицированными таким образом, что по названию индекса можно будет понять, в каких пределах он изменяется и к какому множеству элементов относится.

Множество объектов, исследуемых с помощью ИСИАД, будем называть *множеством решаемых задач*:

*Задача* определяется как набор *входных параметров* и *единственного выходного параметра* :

Каждый входной и выходной параметр задачи содержит в себе следующие характеристики:

* *Название параметра*
* *Тип параметра:* целый или действительный
* *Область допустимых значений*

Заметим, что параметр задачи не содержит внутри себя конкретное значение. *Значения* выходных параметров и выходного параметра будем обозначать как и соответственно.

Каждой задаче соответствует одна или несколько *выборок* (генеральных, тестовых, обучающих):

*Выборка* является множеством записей – *значений выборки*:

Каждое *значение выборки* представляет собой совокупность из значений входных параметров и соответствующего им значения выходного параметра:

Множество всех *алгоритмов обучения*, используемых в ИСИАД, обозначим как :

Обученные для решения текущей задачи методы ИАД будем называть *решателями*. Каждой задаче соответствует множество решателей , при этом каждый решатель относится только к одной задаче.

Каждый решатель является функцией

есть значения параметров -того решателя -й задачи, полученные с использованием -того алгоритма обучения на -й выборке данной задачи.

Примем, что *решением задачи* системой ИСИАД является значение выходного параметра задачи при заданных значениях входных параметров на конкретном решателе. Одной задаче может соответствовать несколько *решений*, полученных на основании использования разных выборок задачи, разных решателей, обученных разными алгоритмами обучения.

Множество решений, соответствующее конкретной задаче, обозначим :

Конкретное *решение задачи* есть совокупность значений входных параметров и соответствующего им выходного решения:

Заметим, что значения входных параметров не зависят от выборки , т.к. это есть значения, отражающие реальную ситуацию, требующую решения. В то же время получаемое значение выходного параметра зависит от выборки, т.к. решатель, с помощью которого было получено это значение, обучался на конкретной -й выборке.

Обучение методов ИАД в системе может происходить 3-мя разными способами:

* *Вся генеральная выборка одновременно является обучающей и тестовой*. Качество обучения в данном случае (отношение числа совпадений тестирования к количеству всех элементов тестовой выборки) обозначим .
* *Генеральная выборка разделяется на обучающую и тестовую.* В данном случае качество обучения есть .
* *Метод кросс-валидации.* Аналогично двум предыдущим случаям, обозначим качество обучения как .

Каждый метод обучения может провести подбор параметров решателя всеми тремя способами независимо. Поэтому, в целях сравнения результатов обучения данными способами, под *общим качеством обучения* будем понимать тройку

При этом результатом обучения – значений параметров решателя – будем считать значения, полученные методом кросс-валидации, как наиболее надежной с точки зрения предоставления различных, несвязанных данных методу ИАД при обучении и тестировании.

Под *вариантом задачи* будем понимать конкретное сочетание задачи, выборки, решателя и алгоритма обучения:

Каждому варианту задачи соответствует:

* Значения параметров решателя
* Значения общего качества обучения
* Решения задачи

## Архитектура ИСИАД

Все необходимые для работы ИСИАД данные записываются в БД. Так, в ней хранятся описание задачи, обучающие и тестовые выборки, обученные параметры решателей. Сами решатели представляют собой библиотеки классов, входящие в состав ИСИАД. Исходный код системы представляет собой группу проектов, написанных на языке C# с помощью Microsoft Visual Studio 2010.

Опишем общую архитектуру ИСИАД на примере полного цикла его работы. Так, чтобы получить возможность решать поставленную задачу ИАД с помощью некоторого метода, включенного в состав ИСИАД, необходимо:

1. *Определить исследуемую задачу*. Исследователю предлагается выбрать название задачи, количество и типы входных параметров, тип выходного параметра. После этого в части БД, хранящей информацию о задачах, создаются новые записи, содержащие данные значения. БД системы содержит множество таблиц, которые условно можно поделить на:
   1. *Таблицы, хранящие информацию о задаче и ее выборках.*
   2. *Группы таблиц, содержащих информацию о методах ИАД: обученные параметры методов, дополнительная информация.*
   3. *Таблицы, включающие в себя данные о качестве обучения. Данные таблицы содержат значения общего качества обучения.*

Все эти группы являются взаимосвязанными (методы ИАД должны знать, к каким задачам они относятся, качество обучение должно также ссылаться на метод, задачу и выборку), вместе они составляют общую БД. Кроме этого, существует связь между данными в БД и библиотеками методов ИАД, методов обучения

(см. Рисунок 17).

*Задачи*

*Параметры ИНС*

*Качество обучения*

*Параметры ДР*

*Библиотека ИНС*

*Библиотека ДР*

*Библиотека алгоритмов обучения*

***База данных ИСИАД***

Рисунок . Структуры хранения данных ИСИАД в общем виде

Таким образом, БД содержит описания вариантов задач, созданных исследователем.

1. *Сформировать выборки данных*. После создания задачи предлагается загрузить предварительно очищенные данные из текстового файла в БД и присвоить таким группам данных названия, сформировав тем самым обучающие и тестовые выборки.
2. *Выбрать метод ИАД, с помощью которого необходимо решить задачу*. Сначала выбирается тип метода ИАД: нейронные сети или деревья решений. Далее следует указать дополнительные параметры метода. Например, в случае выбора ИНС, необходимо будет выбрать тип (на данный момент доступен только персептрон), количество слоев, активационную функцию и т.д. После выполнения этих действий в БД будут отмечены значения этих параметров, тем самым будет зарегистрировано, что для решения задачи подготовлен выбранный метод ИАД.
3. *Выбрать алгоритм обучения*. После выбора метода ИАД необходимо выбрать алгоритм обучения, способный подобрать параметры для данного решателя. Этот этап осуществляется всегда перед обучением, и только вместе с ним, т.к. БД не хранит информации о том, что для обучения текущего метода планируется использовать выбранный алгоритм обучения.
4. *Провести обучение выбранного метода ИАД выбранным алгоритмом обучения.* Для того, чтобы ИСИАД был способен решать поставленную задачу, необходимо выбрать генеральную выборку, которая будет использоваться при обучении – дробиться на обучающие и тестовые согласно способам обучения, рассмотренным выше. Далее необходимо выбрать метод, ранее описанный (проинициализированы все дополнительные параметры) для решения данной задачи. Далее необходимо выбрать алгоритм обучения, способный подобрать параметры данного алгоритма. После этого в автоматическом режиме начнется процесс обучения данного метода ИАД (см. Рисунок 18).

*нет*

Формирование обучающих и тестовых выборок

Обучение

Обучение не требуется

БД

Библиотека решателей

Библиотека АО

Есть

*да*

Рисунок . Общая схема обучения в ИСИАД

Результатом обучения является сохранение в БД значений параметров решателя, полученных методов кросс-валидации, и значений общего качества обучения. В дальнейшем есть возможность сравнения результатов определения качества работы различных методов и выбор наилучшего решателя для дальнейшей работы.

1. *Приступить к решению задачи*. Решение задачи возможно после выбора задачи и решателя, обученного на конкретной выборке конкретным алгоритмом обучения. После этого система загрузит из БД конфигурацию решателя и значения его параметров, полученные алгоритмом обучения. Решение заключается в передачи исследователем значений входных параметров задачи в решатель и формировании им значений выходного параметра.

# Разработка библиотеки нейросетей

Одним из методов ИАД, представленных в ИСИАД, являются нейронные сети. В рамках проекта были разработаны библиотека ИНС, хранящая сами алгоритмы и методы в виде машинных команд, и часть БД, содержащая обученные параметры ИНС и некоторую дополнительную информацию. В данной главе рассматривается архитектура как БД, так и библиотеки ИНС. БД ИСИАД представляет собой реляционную базу данных, спроектированную с использованием СУБД *SQLite* [13]. БД является локальной, используется только приложением ИСИАД, находящемся на локальном ПК.

## База данных ИНС

БД ИНС, как было отмечено в главе 2.2, является частью общей БД ИСИАД. Таблицы, содержащие данные о ИНС, связаны с таблицами, содержащими информацию о задачах, выборках, типах параметров (см. Рисунок 19).

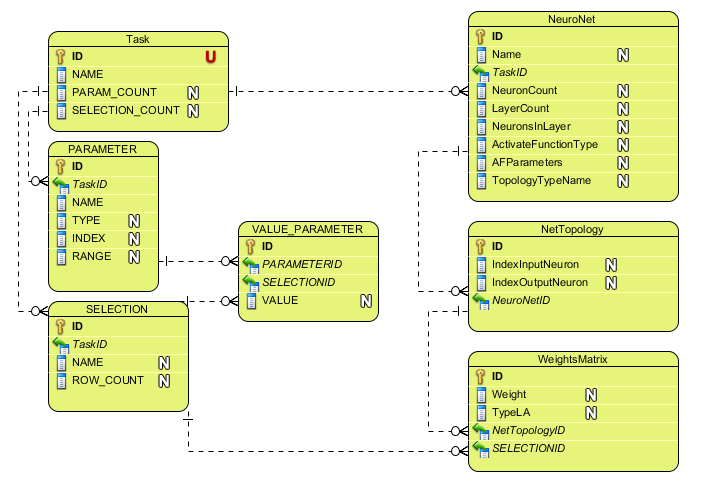


Рисунок . Таблицы БД ИСИАД, хранящие информацию о задачах и ИНС.

БД ИНС содержит в себе следующие таблицы:

* *Таблица ИНС (NeuroNet)*
* *Таблица топологий ИНС (NetTopology)*
* *Таблица значений параметров ИНС как метода ИАД (WeightsMatrix)*

Рассмотрим структуру данных таблиц подробнее. Условимся, что при рассмотрении конкретной таблицы не будем упоминать атрибут *ID*, который является универсальным ключом в таблице, т.к. другого, более важного смысла он в себе не несет.

### Таблица NeuroNet

Данная таблица содержит общую информацию обо всех ИНС, созданных исследователем для работы над всеми созданными задачами в рамках ИСИАД. Т.е. данная таблица содержит описание всех нейронных сетей в ИСИАД. Каждая запись в таблице представлена значениями следующих атрибутов:

* *Name –* имя ИНС, назначенное исследователем. Представляет собой строку.
* *TaskID –* идентификационный номер задачи, для решения которой создана данная ИНС. Задачи хранятся в таблице *Task*.
* *NeuronCount –* общее число нейронов в данной ИНС. Представляет собой целое число, используется при построении объекта ИНС в библиотеке нейронных сетей для обучения или решения задачи данной ИНС.
* *LayerCount –* число слоев данной ИНС.
* *NeuronsInLayer –* распределение нейронов в ИНС по слоям. Представляет собой строку вида где – число нейронов в -м слое. Первый слой в ИНС является входным, последний – выходным.
* *ActivateFunctionType –* название типа активационной функции, примененной в нейронах данной ИНС. Согласно текущей архитектуре БД, ИНС может иметь нейроны только с одним типом активационной функции. Данный тип совпадает с названием класса в библиотеке ИНС, представляющей объекты данной АФ.
* *AFParameters –* содержит в себе значения параметров активационной функции по аналогии с *NeuronsInLayer*. Параметры АФ задают, например, минимальное и максимальное значение, принимаемое АФ, в случае линейной АФ кроме этого еще и тангенс угла наклона.
* *TopologyTypeName –* название топологии, представляющей данную ИНС. Совпадает с названием класса топологии в библиотеке ИНС. На данный момент доступна только топология «персептрон» (*Perceptron*).

### Таблица NetTopology

Эта таблица содержит топологии всех ИНС, представленных в ИСИАД, т.е. представляет описание связей нейронов друг с другом. Таким образом, она состоит из следующих атрибутов:

* *IndexInputNeuron –* номер нейрона в ИНС, из которого выходит связь с другим нейроном. Все нейроны сети имеют номера от до включительно.
* *IndexOutputNeuron –* номер нейрона, в котором данная связь присутствует как входная.
* *NeuroNetID –* идентификатор, связывающий запись в данной таблицей с конкретной ИНС, представленной записью в таблице *NeuroNet.*

### Таблица WeightsMatrix

Данная таблица содержит значения параметров ИНС как метода ИАД­ – значений весов связей, полученных конкретным алгоритмом обучения. Каждая запись состоит из следующих атрибутов:

* *Weight –* действительное число, представляющее собой вес конкретной связи в определенной ИНС.
* *TypeLA –* название класса из библиотеки алгоритмов обучения, объектом которого был получен данный вес.
* *NetTopologyID –* идентификатор записи, соответствующей связи между нейронами, для которой получен данный вес. Напомним, что связь имеет ссылку на ИНС.
* *SELECTIONID –* Идентификатор генеральной выборки, по которой был получен данный вес. Выборки хранятся в таблице *SELECTION*.

Напомним, что данная таблица содержит только значения весов, полученные методом кросс-валидации.

## Библиотека ИНС

Библиотека ИНС является частью проекта ИСИАД, содержится в коде соответствующего решения и включена в исполняемый файл приложения. Условно данную библиотеку можно разделить на следующие составляющие:

* Классы, представляющие *нейрон*.
* Описания *активационных функций*, доступных в ИСИАД; соответствующие классы.
* *Топологии*, доступные в проекте.
* Классы для решения задач с помощью ИАД и интерфейсы для методов обучения.
* Вспомогательные классы – содержат в себе функционал загрузки из БД необходимой информации для работы библиотеки; контейнеры, используемые для передачи информации от одного модуля библиотеки к другому.

Рисунок 20 отражает *диаграмму классов* библиотеки ИНС. Заметим, что на данной диаграмме присутствуют описания не всех полей и методов в классах, выбраны только самые важные поля и методы. Стоит также отметить, что в данную диаграмму не вошли следующие классы:

* NeuroNetLearningInterface – класс, предоставляющий безопасный доступ к ИНС алгоритмов обучения, упрощающий подбор параметров сети данными алгоритмами.
* NeuroNetDefinition – контейнер с информацией об используемой ИНС. Используется для передачи данных между модулями библиотеки ИНС.
* DataBaseHandler – осуществляет извлечение данных из БД.

Данные классы являются вспомогательными или «межбиблиотечными», поэтому в дальнейшем не будем касаться деталей их архитектуры.

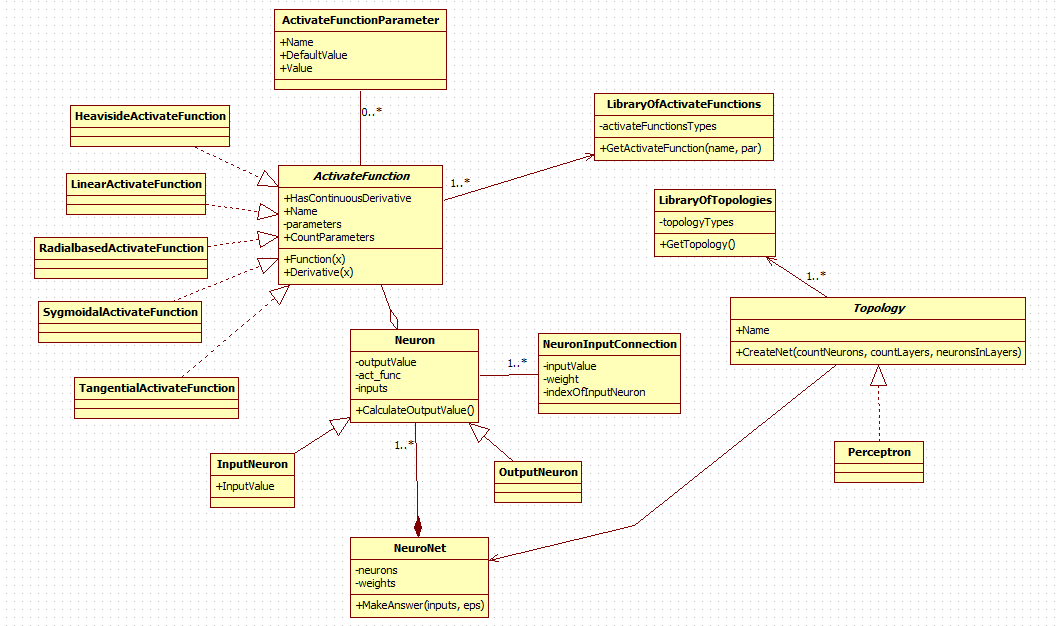


Рисунок . Диаграмма классов библиотеки ИНС

Рассмотрим подробнее архитектуру и функциональность классов, описанных на диаграмме классов библиотеки ИНС.

### Класс ActivateFunctionParameter

Данный класс представляет собой параметр активационной функции. Например, с помощью данного класса реализуется возможность дать АФ различные минимальные и максимальные значения, крутизну наклона. Класс содержит следующие поля:

* String name – название данного параметра.
* Double defaultValue – значение данного параметра по умолчанию.
* Double Value – текущее значение параметра.

Методов данный класс не имеет (кроме конструктора).

### Абстрактный класс ActivateFunction

Данный класс представляет АФ нейрона. Любая конкретная активационная функция должна быть наследована от данного класса.

*Поля:*

* bool HasContinuousDerivative – если АФ имеет непрерывную первую производную, данная переменная установлена в значение true.
* String Name – название данной АФ.
* List<ActivateFunctionParameter> parameters – список параметров, имеющихся в данной АФ.
* int CountParameters – количество параметров АФ. Необходимо сторонним классам, т.к. список параметров от них скрыт.

Методы:

* string GetNameOfParameter – возвращает имя параметра по его индексу в списке.
* double GetDefaultValueOfParameter – возвращает значение параметра по умолчанию по имени или индексу данного параметра.
* double GetValueOfParameter – аналогично, возвращает текущее значение параметра.
* void SetValueOfParameter – задает новое значение параметру.
* double Function – возвращает значение АФ в конкретной точке x.
* double Derivative – аналогично, возвращает значение производной АФ.

Данный класс имеет следующие реализации в ИСИАД:

* HeavisideActivateFunction – пороговая АФ.
* LinearActivateFunction – линейная АФ.
* RadialbasedActivateFunction – радиально-базисная АФ.
* SygmoidalActivateFunction – сигмоидальная АФ.
* TangentialActivateFunction – гиперболический тангенс.

### Статический класс LibraryOfActivateFunctions

Данный класс представляет собой *фабрику* АФ. Благодаря ему другие компоненты ИСИАД могут получить объект, представляющий активационную функцию требуемого типа. Также наличие данного класса предоставляет упрощение в расширении типов АФ. Так, для того, чтобы новая АФ стала доступна в системе, необходимо только лишь реализовать абстрактный класс ActivateFunction, наследовав от него новый класс. Данная фабрика сама находит всех наследников ActivateFunction, после чего они становятся доступны в системе через данный класс.

*Поля:*

* List<Type> activateFunctionsTypes – список названий всех классов-наследников ActivateFunction.

Методы:

* int GetCountActivateFunctions – возвращает количество типов АФ, доступных в ИСИАД
* string[] GetAllActivateFunctionNames – возвращает имена всех активационных функций разных типов, по одному на каждый тип.
* string GetActivateFunctionName – возвращает название АФ по названию класса, объектом которого она является.
* string GetActivateFunctionTypeName – обратная операция.
* ActivateFunction GetActivateFunction – возвращает объект АФ по названию АФ или по названию класса АФ.

### Абстрактный класс Topology

Данный класс содержит имя создаваемой топологии сети

string Name

и метод CreateNet, который должен быть реализован в наследниках данного класса. Данный метод возвращает матрицу bool[,], содержащую в себе информацию о связанности нейронов в сети – *матрицу инцидентности*. Так, если на пересечении -го столбца и -й строки данной матрицы стоит значение true, значит между нейронами с данными индексами в ИНС есть связь, выходящая из нейрона, представленного -й строкой.

На данный момент единственной реализованной топологией является персептрон, представленный одноименным классом Perceptron, являющимся реализацией данного абстрактного класса.

### Статический класс LibraryOfTopologies

Данный класс построен аналогично LibraryOfActivateFunctions, следовательно, создание новых топологий имеет те же преимущества с точки зрения простоты встраивания новых элементов в ИСИАД, что и новые АФ.

*Поля:*

* *List<Type> topologyTypes –* названия всех классов, наследованных от Topology.

*Методы:*

* *Topology GetTopology –* возвращает объект топологии по имени топологии или имени класса, реализующего ее.
* *List<string> GetAllTopologyTypeNames –* возвращает имена всех типов (названия классов) топологий, доступных в ИСИАД.
* *string GetTopologyName –* возвращает имя топологии по названию класса.
* *string GetTopologyTypeName –* производит обратное действие.

### Класс NeuronInputConnection

Данный класс представляет собой входную связь нейрона; методов не имеет, содержит следующие поля:

* *double inputValue –* выходное значение нейрона, который связан с текущим посредством выходной связи.
* *double weight –* значение веса данной связи
* *int indexOfInputNeuron –* индекс нейрона, с которым связан данный посредством этой связи.

### Класс Neuron

Данный класс представляет собой нейрон, структурную единицу ИНС.

*Поля:*

* *double outputValue –* выходное значение данного нейрона, полученное после применения АФ к взвешенной сумме входов.
* *ActivateFunction act\_func –* активационная функция данного нейрона.
* *NeuronInputConnection[] inputs –* входные связи данного нейрона.

*Методы:*

* *SetInputConnections –* устанавливает входные связи данного нейрона по массиву объектов NeuronInputConnection.
* *SetInputValue –* устанавливает выходное значение нейрона в связи по индексу этого нейрона, имеющего выходную связь с данным.
* *SetWeightValue –* устанавливает вес в связи по индексу нейрона.
* *CalculateOutputValue –* вычисляет выходное значение данного нейрона, подсчитав взвешенную сумму входов и применив АФ.

Существуют 2 класса-наследника: InputNeuron и OutputNeuron, немного отличающиеся от данного процессом вычисления выходного значения. Кроме этого, InputNeuron позволяет установить входное значение ИНС для данного нейрона.

### Класс NeuroNet

Данный класс представляет собой обертку для нейронной сети любой топологии. С помощью данного класса генерируются объекты, представляющие собой ИНС любого доступного в рамках ИСИАД типа (т.е. для которого реализованы наследники классов Topology и ActivateFunction). Таким образом, объекты данного класса могут представлять собой любую ИНС. Для создания конкретной ИНС необходимо передать в конструктор число нейронов в сети, количество входных и выходных нейронов, распределение нейронов по слоям, матрицу инцидентности, матрицу весов сети и АФ. Алгоритм работы ИНС, представленный в данном классе, способен выполнять вычисления как сети прямого распространения, так и рекуррентной.

*Поля:*

* *Neuron[] neurons –* массив всех нейронов сети.
* *InputNeuron[] input\_neurons –* массив входных нейронов. Они являются элементами также и массива *neurons.*
* *OutputNeuron[] output\_neurons –* массив выходных нейронов сети. Аналогично входным, являются элементами массива *neurons.*
* *bool isIterationsFinished –* переменная, определяющая, завершилась ли при решении задачи *итерация* – это ситуация, когда все нейроны ИНС отработали, нет больше нейронов, которым на вход пришло какое-либо значение. Итерация может завершиться только у сетей прямого распространения.
* *bool isWaveCameToOutputNeuron –* определяет ситуацию, когда становится активным один из выходных нейронов. Нейрон активен, если на все его входы пришли новые значения и, следовательно, требуется вычислить его выходное значение.
* *int[] neuronsInLayers –* распределение нейронов по слоям
* *bool[,] topology –* матрица инцидентности ИНС
* *double[,] weights –* матрица весов ИНС

*Методы:*

* *void ResetNeuroNet –* переводит ИНС в исходное начальное состояние – ожидания входного сигнала от исследователя.
* *Neuron GetNeuron –* возвращает нейрон по его индексу в сети.
* *int GetIndexNeuron –* совершает обратное действие.
* *void SetNewConnection –* создает новую связь между нейронами по их индексам и весу связи.
* *void DeleteConnection –* удаляет связь по индексам входного и выходного нейрона.
* *double[] MakeStep –* выполняет вычисление значений выходного вектора по значениям входного. Завершает свою работу, если ИНС завершила итерацию, либо стали активными рекуррентные связи. Опишем алгоритм работы данного метода:

1. *Входные значения входных нейронов инициализируются входным значением вектора ИНС.*
2. *Входные нейроны добавляются в пул[[5]](#footnote-5) активных нейронов.*
3. *Пока не завершена итерация или не активны рекуррентные связи:*
   1. *Создать новый пул нейронов – под нейроны следующего за данным слоя.*
   2. *Для каждого нейрона текущего пула выполнить:*
      1. *Вычислить его выходное значение*
      2. *Добавить в следующий пул все нейроны, с которыми связан данный посредством выходной связи (если таких еще нет в пуле). Если данный нейрон связан с входными нейронами, пометить флаг останова расчета значением true.*
      3. *Проверить, является ли текущий нейрон выходным.*
      4. *Если он выходной и активны рекуррентные связи – проинициализировать выходной вектор ИНС текущими выходными значениями выходных нейронов, пометить флаг останова расчета значением true.*
   3. *Для каждого нейрона текущего пула выполнить[[6]](#footnote-6):*
      1. *Установить входные значения связей нейронов, связанных с данным, равным выходному значению данного нейрона.*
   4. *Если следующий пул не пуст, сделать его текущим.*
   5. *Если флаг останова помечен значением true, вернуть выходной вектор.*
4. *Пометить флаг завершения итерации значением true.*
5. *Проинициализировать вектор выходных значений ИНС выходными значениями выходных нейронов.*
6. *Вернуть выходной вектор.*

* *double[] MakeIteration -* выполняет вычисление значений выходного вектора по значениям входного. Завершает свою работу, если стал активен хотя-бы один нейрон выходного слоя. Алгоритм работы основан на последовательном применении *MakeStep.*
* *double[] MakeAnswer –* самый общий метод решения. Выполняет вычисление значений выходного вектора по значениям входного и *ошибке*, которою необходимо преодолеть. Завершает свою работу при достижении этой заданной *точности*. Под *ошибкой* понимается евклидова норма разности выходных векторов, полученных на двух соседних итерациях работы ИНС. Заметим, что в сетях прямого распространения первые две итерации обращают ошибку в нуль.

## Работа с нейронными сетями

В данном пункте описывается процесс создания, обучения ИНС и решения с их помощью поставленных задач посредством интерфейсной части, обеспечивающей взаимодействия пользователя с БД и библиотекой ИНС. Данная интерфейсная часть состоит из нескольких окон. Главное окно выполняет справочную функцию – в нем можно получить информацию о созданных ИНС, а также выбрать действие над ИНС (добавление новой ИНС, обучение, решение задачи, и т.д.). Вспомогательные окна позволяют выполнить выбранное действие. Таким образом, интерфейсная часть позволяет:

1. Просмотреть информацию о созданных нейронных сетях
2. Описать новую ИНС
3. Выбрать ИНС для решения задачи или обучения
4. Решить задачу выбранной ИНС

Опишем подробнее процесс осуществления каждого из этих действий.

### Просмотр созданных ИНС

Главное окно интерфейса работы с ИНС содержит 2 вкладки: «Редактирование нейронных сетей» и «Использование нейронных сетей» (см. Рисунок 21, Рисунок 23). Справочная информация о структуре созданных ИНС содержится в первой вкладке. Она содержит 3 таблицы:

* *Описания созданных ИНС*. Содержит информацию о всех созданных в ИСИАД нейронных сетях. Здесь приводится название каждой ИНС, тип ее топологии, задача, для которой была создана данная ИНС, количество нейронов и слоев в сети, примененная АФ.
* *Распределение нейронов по слоям.* Содержит информацию о числе нейронов в каждом слое ИНС. Первый слой является входным, последний – выходным. Для того, чтобы в данной таблице появилась информация, нужно выбрать конкретную ИНС в предыдущей таблице, выделив её соответствующую строку.
* *Параметры активационной функции.* Содержит значения параметров АФ рассматриваемой ИНС.

Также в данной вкладке можно изменить или удалить выбранную ИНС, или перейти в окно добавления новой ИНС.

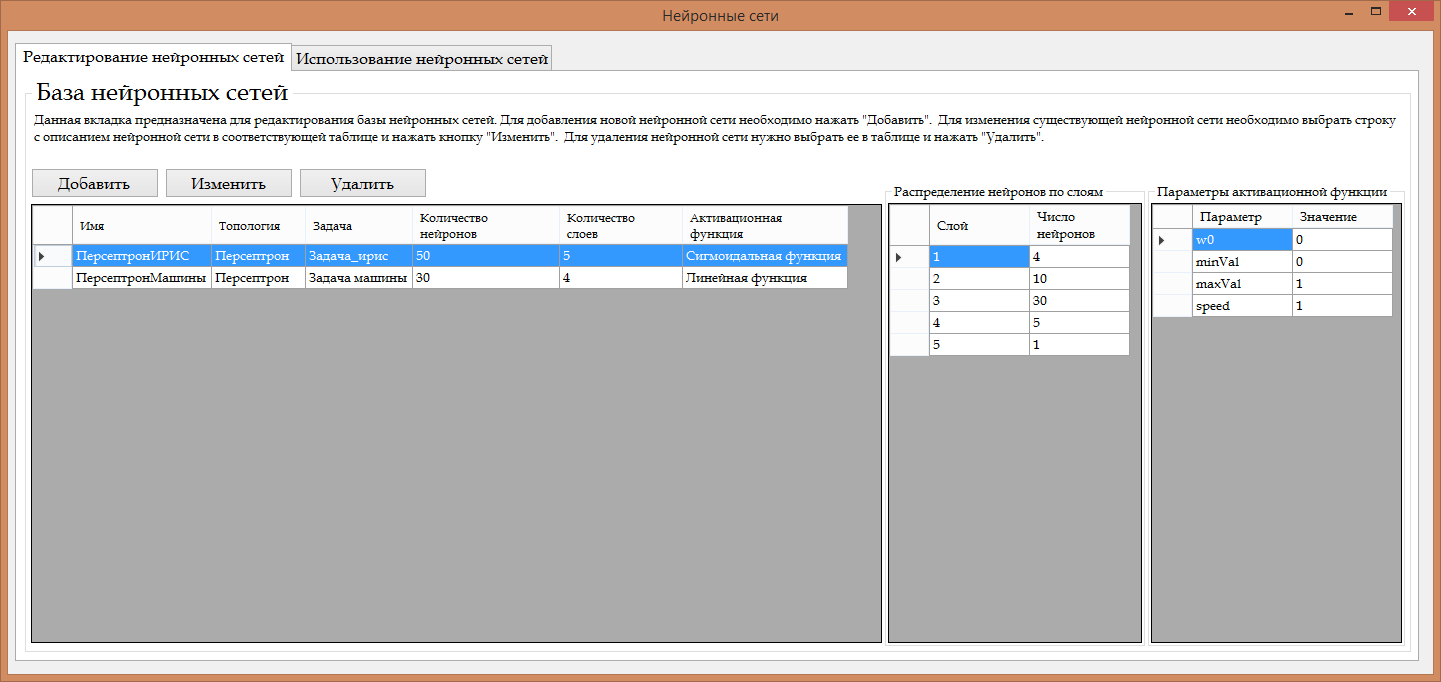


Рисунок . Вкладка "Редактирование нейронных сетей"

### Описание новой ИНС

После нажатия кнопки «Добавить» возникнет новое окно, в котором требуется указать параметры создаваемой ИНС (см. Рисунок 22). В том числе:

* *Название ИНС.* Пользователь не ограничен ни какими рамками при выборе названия создаваемой сети.
* *Задача.* Необходимо выбрать из списка имеющихся в ИСИАД задач (заранее созданных) ту, для решения которой создается данная ИНС.
* *Топология.* Из списка, содержащего названия всех имеющихся топологий библиотеки ИНС, необходимо выбрать ту, которая будет отражать внутреннюю структуру создаваемой ИНС.
* *Число нейронов в ИНС.* Указывается общее число нейронов в сети.
* *Число слоев в ИНС.*
* *Активационная функция.* Выбирается аналогично топологии.
* *Распределение нейронов по слоям.* В данной таблице требуется указать число нейронов в каждом слое. Заметим, что число нейронов во входном и выходном слоях ставится автоматически и является не редактируемым, т.к. данные параметры зависят от выбранной задачи, числа входных и выходных параметров в ней. Общее число нейронов должно совпадать с суммой числа нейронов по слоям, иначе создать ИНС не удастся, будет выведено информационное окно ошибки.
* *Параметры активационной функции.* В данной таблице необходимо указать значения параметров выбранной АФ. Эти значения могут быть только действительными числами.

После корректного заполнения данного окна и нажатия кнопки «Применить» создаваемая ИНС будет записана в БД и доступна для просмотра и применения в проекте ИСИАД.

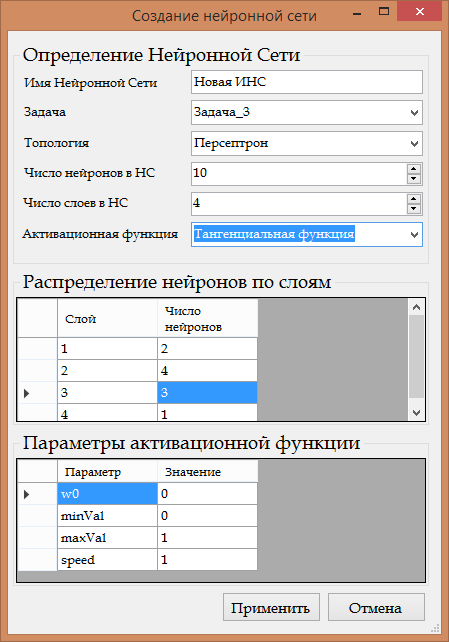


Рисунок . Описание новой ИНС

### Выбор ИНС для решения задачи или обучения

Вторая вкладка главного окна позволяет выбрать одну из созданных ИНС и перейти в режим решения задачи, если данная сеть обучена, либо обучить ее. Для этого необходимо выполнить следующие действия:

1. В дереве задач выбрать рассматриваемую задачу.
2. Открыть список выборок данной задачи (в этом же дереве) и выбрать обучающую выборку.
3. Выбрать ИНС из доступных для данной задачи в таблице нейронных сетей.
4. Выбрать алгоритм обучения.

Если выбранная ИНС обучена на выбранной выборке и текущем алгоритме обучения, станет доступны кнопки «Решить» и «Удалить», позволяющие решить задачу данной ИНС и удалить обученные веса соответственно. В противном случае станет доступна кнопка «Обучить», при нажатии которой произойдет переход в интерфейсную часть выбранного алгоритма обучения, где можно будет обучить выбранную ИНС на текущей выборке.

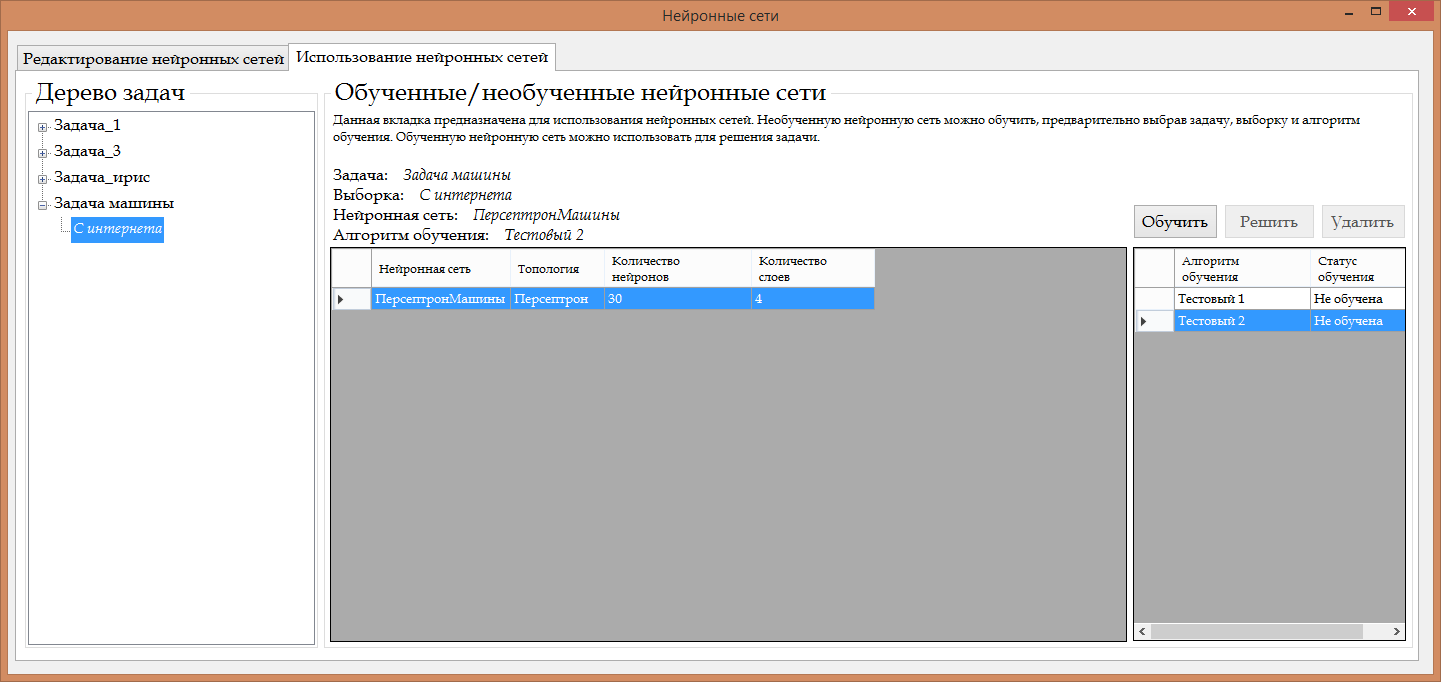


Рисунок . Вкладка "Использование нейронных сетей"

### Решение задачи выбранной ИНС

При нажатии кнопки «Решить» возникает окно, содержащее группу полей входных значений и группу полей выходных значений (число данных полей зависит от решаемой задачи). Для решения задачи необходимо записать значения входного вектора в первую группу полей и нажать кнопку «Решить» (см. Рисунок 24). После этого в группу полей выходных значений будут выведены значения выходного вектора, посчитанные выбранной нейронной сетью.

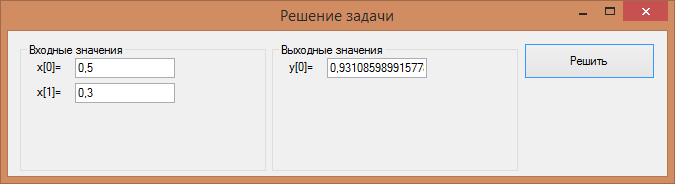


Рисунок . Окно решения задачи

# Заключение

В рамках данной работы был разработан проект ИСИАД (в команде), упрощающий проведение завершающих этапов ИАД. Была разработана часть БД, содержащая информацию о применяемых в проекте ИНС. Также была спроектирована и реализована расширяемая библиотека ИНС. Внутренне устройство данной библиотеки позволяет использовать в ИСИАД новые АФ нейронов и новые топологии сети (в том числе рекуррентные, в перспективе – со стохастическими нейронами, с задержкой передачи сигнала) без дополнительных усилий по интеграции новых частей в данную библиотеку. ИСИАД позволяет сравнить результаты работы различных методов решения задач. Библиотека классов ИНС была написана на языке C# с использованием интегрированной среды разработки Microsoft Visual Studio 2010. БД ИНС была спроектирована с помощью СУБД SQLite. В рамках библиотеки ИНС были спроектированы 16 классов; общий объем кода реализации библиотеки ИНС составил порядка 1500 строк. Кроме этого, были созданы 3 служебных класса, обеспечивающих извлечение данных из БД и обмен данными между классами библиотеки ИНС, 2 класса, обеспечивающих взаимодействие с алгоритмами обучения. Объем кода реализации данных классов составил примерно 1200 строк (большую часть данного объема заняли методы обмена информацией программы и БД ИСИАД). Также были созданы 4 формы интерфейсной части, обеспечивающей взаимодействия пользователя, БД и библиотеки ИНС. Код обработчиков событий данных форм составил около 700 строк.

# Список литературы

1. Степанов Р.Г. Технология Data Mining: Интеллектуальный Анализ Данных / Р.Г. Степанов – Казань: ГОУ ВПО КГУ, 2008. – 57 с.
2. Мичи Д. Компьютер-творец / Д. Мичи, Р. Джонсон – М.: Мир, 1987 – 254 с.
3. Deep Learning. – Режим доступа: <http://deeplearning.net>, свободный.
4. Пенроуз Р. Новый ум короля: О компьютерах, мышлении и законах физики. Пер. с англ. / Р. Пенроуз – М.: Изд. 4-е, УРСС: изд-во ЛКИ, 2011 – 400 с.
5. Чубукова И.А. Data Mining. – Режим доступа: http://www.intuit.ru/studies/courses/6/6/info, свободный.
6. Николенко С.И. Деревья принятия решений / MachineLearning – CS Club, весна 2008.
7. Воронцов К.В. Лекции по методу опорных векторов. – Режим доступа: <http://www.ccas.ru/voron/download/SVM.pdf>, свободный.
8. Нейронные сети и их устройство. – Режим доступа:

<http://ap-economics.narod.ru/library/NLP.html>, свободный.

1. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы.

Пер. с польск. / Д. Рутковская, М. Пилиньский, Л. Рутковский, – М.: Горячая линия-Телеком, 2006 – 452 с.

1. Хайкин С. Нейронные сети: полный курс, 2-е издание. Пер. с англ. / С. Хайкин – М.: Издательский дом «Вильямс», 2006 – 1104 с.
2. Николенко С.И. Генетические алгоритмы. – Режим доступа: logic.pdmi.ras.ru/~sergey/teaching/ml/04-genetic.pdf, свободный.
3. Основные понятия интеллектуального анализа данных. – Режим доступа: <https://msdn.microsoft.com/ru-ru/library/ms174949.aspx>, свободный.
4. Официальный сайт СУБД SQLite. – Режим доступа: <https://www.sqlite.org>, свободный.

# Приложения

## Приложение А. Заголовочные части классов библиотеки активационных функций

public class ActivateFunctionParameter

{

private string name;

private double defaultValue;

public double Value;

public ActivateFunctionParameter(string \_name, double \_defVal)

{

name = \_name;

defaultValue = \_defVal;

Value = defaultValue;

}

public ActivateFunctionParameter(ActivateFunctionParameter par)

{

name = par.name;

defaultValue = par.defaultValue;

Value = par.Value;

}

public string Name { get { return name; } }

public double DefaultValue { get { return defaultValue; } }

}

public abstract class ActivateFunction

{

public abstract bool HasContinuousDerivative { get; }

public abstract string Name { get; }

protected List<ActivateFunctionParameter> parameters;

public int CountParameters { get { return parameters.Count; } }

public string GetNameOfParameter(int index)

public double GetDefaultValueOfParameter(string name)

public double GetDefaultValueOfParameter(int index)

public double GetValueOfParameter(string name)

public double GetValueOfParameter(int index)

public void SetValueOfParameter(string name, double value)

public void SetValueOfParameter(int index, double value)

public ActivateFunction()

public ActivateFunction(ActivateFunction af)

public abstract double Function(double x);

public abstract double Derivative(double x);

}

public static class LibraryOfActivateFunctions

{

static private List<Type> activateFunctionsTypes;

public enum GetterParameter { ActivateFunctionName, TypeOfActivateFunctionName };

static LibraryOfActivateFunctions()

static public int GetCountActivateFunctions()

static public string[] GetAllActivateFunctionNames()

static public string GetActivateFunctionName(string typeName)

static public string GetActivateFunctionTypeName(string activateFunctionName)

static public int GetCountParametersOfAF(string name, GetterParameter par)

static public double GetDefaultValueOfParameterAF(string name, int indexPar, GetterParameter par)

static public double GetDefaultValueOfParameterAF(string name, string namePar, GetterParameter par)

static public ActivateFunction GetActivateFunction(string name, GetterParameter par)

}

## Приложение Б. Класс сигмоидальной активационной функции

class SygmoidalActivateFunction:ActivateFunction

{

public override bool HasContinuousDerivative

{

get { return true; }

}

public override string Name

{

get { return "Сигмоидальная функция"; }

}

public SygmoidalActivateFunction()

{

parameters = new List<ActivateFunctionParameter>();

parameters.Add(new ActivateFunctionParameter("w0", 0.0));

parameters.Add(new ActivateFunctionParameter("minVal", 0.0));

parameters.Add(new ActivateFunctionParameter("maxVal", 1.0));

parameters.Add(new ActivateFunctionParameter("speed", 1.0));

}

public override double Function(double x)

{

double w0 = parameters[0].Value;

double minVal = parameters[1].Value;

double maxVal = parameters[2].Value;

double speed = parameters[3].Value;

return minVal + (maxVal + 1) / (1 + Math.Exp(-speed\*(x - w0)));

}

public override double Derivative(double x)

{

double w0 = parameters[0].Value;

double maxVal = parameters[2].Value;

double speed = parameters[3].Value;

double exp = Math.Exp(speed\*(w0 - x));

return (maxVal + 1) \* speed \* exp / Math.Pow(1 + exp, 2);

}

}

## Приложение В. Класс топологии «Персептрон»

public class Perceptron : Topology

{

public override string Name

{

get { return "Персептрон"; }

}

public Perceptron()

{

}

public override bool[,] CreateNet(int countNeurons, int countLayers, int[] neuronsInLayers)

{

if (countNeurons <= 0)

throw new Exception("Invalid count of neurons");

if (countLayers < 2 || countLayers > countNeurons)

throw new Exception("Invalid count of layers");

if (neuronsInLayers.Length != countLayers)

throw new Exception("Invalid dimension of 'neurons in layers' array");

int sum = 0;

for (int i = 0; i < neuronsInLayers.Length; i++)

{

if (neuronsInLayers[i] <= 0)

throw new Exception(String.Format("Invalid count of neurons in {0} layer", i + 1));

sum += neuronsInLayers[i];

}

if (sum != countNeurons)

throw new Exception("Invalid sum of count of neurons in array");

bool[,] connections = new bool[countNeurons, countNeurons];

for (int i = 0; i < countNeurons; i++)

{

for (int j = 0; j < countNeurons; j++)

{

connections[i, j] = false;

}

}

int out\_neuron\_index = 0;

int in\_neuron\_index = neuronsInLayers[0];

for (int i = 0; i < countLayers - 1; i++)

{

for (int j = 0; j < neuronsInLayers[i]; j++)

{

for (int k = 0; k < neuronsInLayers[i + 1]; k++)

{

connections[out\_neuron\_index, in\_neuron\_index] = true;

in\_neuron\_index++;

}

out\_neuron\_index++;

in\_neuron\_index -= neuronsInLayers[i + 1];

}

in\_neuron\_index += neuronsInLayers[i + 1];

}

return connections;

}

}

## Приложение Г. Заголовочные части классов NeuronInputConnection, Neuron и NeuroNet

public class NeuronInputConnection

{

public double inputValue;

public double weigth;

public int indexOfInputNeuron;

public NeuronInputConnection(double \_weight, int \_indexOfInputNeuron)

public NeuronInputConnection(double \_weight, double \_inputValue, int \_indexOfInputNeuron)

public NeuronInputConnection(NeuronInputConnection nic)

}

public class Neuron

{

protected double outputValue;

protected ActivateFunction act\_func;

protected NeuronInputConnection[] inputs;

public double OutputValue { get { return outputValue; } }

public int InputsCount

public ActivateFunction ActivateFunctionOfNeuron { get { return act\_func; } }

protected Neuron()

public Neuron(ActivateFunction af)

public Neuron(ActivateFunction af, NeuronInputConnection[] \_inputs)

public Neuron(Neuron neu)

public void SetInputConnections(NeuronInputConnection[] \_inputs)

public void SetInputValue(double inputValue, int indexOfInputConnection)

public void SetWeightValue(double weightValue, int indexOfInputConnection)

public virtual void CalculateOutputValue()

{

double sum = 0;

foreach (NeuronInputConnection item in inputs)

{

sum += item.weigth \* item.inputValue;

item.inputValue = 0.0;

}

outputValue = act\_func.Function(sum);

}

}

public class NeuroNet

{

private Neuron[] neurons;

private InputNeuron[] input\_neurons;

private OutputNeuron[] output\_neurons;

private Queue<Queue<Neuron>> evaluation\_machine;

private bool isIterationsFinished;

private bool isWaveCameToOutputNeuron;

private int[] neuronsInLayers;

private bool[,] topology;

private double[,] weights;

public int NeuronsCount { get { return neurons.Length; } }

public int InputNeuronsCount { get { return input\_neurons.Length; } }

public int OutputNeuronsCount { get { return output\_neurons.Length; } }

public bool IsIterationsFinished { get { return isIterationsFinished; } }

public bool IsWaveCameToOutputNeuron { get { return isWaveCameToOutputNeuron; }

public int[] NeuronsInLayers

public bool[,] ConnectionsOfNeurons

public double[,] WeightsOfConnections

private void addInputNeuronsToFirstPool()

private void setInputConnections()

public NeuroNet(int numb\_input\_neurons, int numb\_output\_neurons, int[] numbNeuronsInLayers,

bool[,] \_connections, double[,] \_weights, ActivateFunction af)

public NeuroNet(NeuroNet net)

public void ResetNeuroNet()

public Neuron GetNeuron(int index)

public int GetIndexNeuron(Neuron neuron)

public void SetNewConnection(int indexInput, int indexOutput, double weight)

public void DeleteConnection(int indexInput, int indexOutput)

public double[] MakeStep(double[] inputs)

public double[] MakeIteration(double[] inputs)

public double[] MakeAnswer(double[] inputs, double eps = 1E-16)

}

1. Mining (с англ.) – горное дело, горная промышленность. [↑](#footnote-ref-1)
2. Продукционное правило – знание, представленное в виде логической конструкции «Если …, то …». Например, «Если *идет дождь*, следует *взять зонтик*»*.* [↑](#footnote-ref-2)
3. Энтропия множества есть мера неопределенности, разнородности элементов данного множества. Чем выше неопределенность, тем больше значение энтропии. [↑](#footnote-ref-3)
4. Скалярное произведение векторов и записывается как . [↑](#footnote-ref-4)
5. Пул нейронов – контейнер, содержащий те нейроны, которые должны сформировать свое выходное значение одновременно. В сетях прямого распространения один пул всегда содержит один слой нейронов. В рекуррентных ИНС в одном пуле могут оказаться нейроны из разных слоев. [↑](#footnote-ref-5)
6. Условие цикла то же, но выполнять в цикле, предыдущем данному, эти действия нельзя, т.к. необходимо, чтобы сначала все нейроны из текущего пула выполнили расчет своих выходных значений. [↑](#footnote-ref-6)