Министерство образования и науки Российской Федерации

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

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**КУРСОВАЯ РАБОТА**

**Инструментальная СИАД. Разработка библиотеки нейросетей**

**Выполнил**:студент группы М0816-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Смирнов М.А.

Подпись

**Научный руководитель**:

к.т.н., доц.,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Карпенко С.Н.

Подпись

Нижний Новгород  
2016

Оглавление

[Введение 4](#_Toc451976784)

[1. Постановка задачи 6](#_Toc451976785)

[2. Используемые методы ИАД 7](#_Toc451976786)

[2.1. Нейронные сети 7](#_Toc451976787)

[2.2. Деревья решений 10](#_Toc451976788)

[2.3. Генетический алгоритм 12](#_Toc451976789)

[3. Общая архитектура ИСИАД 14](#_Toc451976790)

[4. Архитектура библиотеки нейронных сетей 19](#_Toc451976791)

[4.1. Класс ActivateFunctionParameter 20](#_Toc451976792)

[4.2. Абстрактный класс ActivateFunction 20](#_Toc451976793)

[4.3. Статический класс LibraryOfActivateFunctions 21](#_Toc451976794)

[4.4. Абстрактный класс Topology 22](#_Toc451976795)

[4.5. Статический класс LibraryOfTopologies 22](#_Toc451976796)

[4.6. Класс NeuronInputConnection 23](#_Toc451976797)

[4.7. Класс Neuron 23](#_Toc451976798)

[4.8. Класс NeuroNet 24](#_Toc451976799)

[5. Вычислительный эксперимент 27](#_Toc451976800)

[5.1. Описание задач 27](#_Toc451976801)

[5.1.1. Ирис 28](#_Toc451976802)

[5.1.2. Морское ушко 28](#_Toc451976803)

[5.1.3. Пожары в лесах 30](#_Toc451976804)

[5.1.4. Комната 31](#_Toc451976805)

[5.1.5. Кожа лица 32](#_Toc451976806)

[5.2. Условия проведения эксперимента 32](#_Toc451976807)

[5.3. Результаты 33](#_Toc451976808)

[5.3.1. Ирис 34](#_Toc451976809)

[5.3.2. Морское ушко 35](#_Toc451976810)

[5.3.3. Пожары в лесах 36](#_Toc451976811)

[5.3.4. Комната 38](#_Toc451976812)

[5.3.5. Кожа лица 38](#_Toc451976813)

[5.3.6. Сравнение результатов топологии по всем задачам 40](#_Toc451976814)

[Заключение 42](#_Toc451976815)

[Список литературы 43](#_Toc451976816)

# Введение

Повсеместное использование электронных вычислительных средств во многих сферах деятельности человека привело к стремительному росту объемов хранимой и обрабатываемой информации. Общество генерирует гигабайты текстовых файлов, видео, фото и аудио-контента ежедневно благодаря тесной интеграции сети Интернет в его жизнь. Однако большая часть производимой обществом информации находится в «сыром», не стандартизованном, сложном для анализа виде. Такую информацию принято называть данными[1]. Их анализ представляет собой довольно трудоемкую задачу: данные разнородны, содержат большое количество внутренних зависимостей, ошибок ишума (информации, не имеющей ценности для получения результата анализа).

Интеллектуальный анализ данных (англ. Data Mining) занимается извлечением неявной, ранее неизвестной и потенциально полезной информации из больших наборов данных [1]. Для этого применяются различные алгоритмы искусственного интеллектаи машинного обучения: нейронные сети [2], деревья решений [3], метод опорных векторов, случайный лес [4], байесовские сети и т.д. Эти методы хорошо зарекомендовали себя при решении трудно формализуемых задач [5], к каким относится и задача выявления полезной информации. Интеллектуальный анализ данных успешно применяется, например, при диагностике и предсказании сердечно-сосудистых заболеваний [6, 7], увеличении продаж на рынке мобильных телекоммуникаций [8], предсказании мест возникновения аварий [9], оценки индивидуальных лекарственных доз (Ефимов, А.С. Метод оценки индивидуальных лекарственных доз на основе адаптивных нечётких моделей; диссертация на соискание ученой степени к.т.н., Н.Новгород, ННГУ, 2011 г.), анализе следственных версий (Изутов, Е.О. Анализ следственных версий на основе искусственной нейронной сети; выпускная квалификационная работа бакалавра, Н.Новгород, ННГУ, 2013 г.).

Решению задачи анализа данных предшествуют стадии сбора и предобработки данных, выбора модели решения задачи и подбора ее параметров – обучения модели. Большинство современных авторов работ в области интеллектуального анализа данных исследуют возможность использования того или иного метода решения задачи применительно к имеющимся данным; основой их работ является описание совершённых над исходными данными преобразований для улучшения предсказательных способностей решающей модели. Однако данные преобразования являются, в большинстве своем, эффективными только в рассматриваемой автором предметной области. Правда, существуют исследования, выявляющие правила предобработки данных, способствующие повышению эффективности моделей [10, 11]. Но попыток формулирования фундаментальных правил выбора наилучшей модели для решения задач конкретного класса не наблюдается. Большинство работ описывают достижения в решении задач классификации, кластеризации и восстановления зависимости. Некоторые работы предлагают эффективные решения задач анализа временных последовательностей [10]. Каждая из этих работ предлагает способы предобработки данных и обучения конкретной модели; но не делается попытки сравнения и анализа эффективности использования различных методов для решения конкретной задачи.

Эффективность применения методов интеллектуального анализа данных в самых разных научных и прикладных областях способствовало созданию множества различных программных средств и инструментов, позволяющих упростить проведение экспериментов и решение задач [12, 13, 20, 21]. Были сформулированы новые техники обработки больших массивов данных (например, использование распределенной структуры проведения вычислений в облачных серверах для обработки больших объемов данных [14]), новые методы обучения моделей и решения задач (глубокое обучение [15, 16]). Однако современные программные средства интеллектуального анализа данных нацелены на получение максимального результата для выбранного пользователем метода решения задачи.

Цель данной работы заключается в создании библиотеки нейронных сетей в рамках разработки инструментальной системы интеллектуального анализа данных (ИСИАД). Данная система разработана в команде (в которую также входит автор данной работы). ИСИАД включает в себя несколько различных методов решения задач (библиотека нейронных сетей, деревья решений) и алгоритмов обучения. Эта система автоматизирует процесс выбора наилучшей модели для решения задач классификации и восстановления зависимости. Таким образом, актуальность данной работы состоит, во-первых, в упрощении использования методов интеллектуального анализа данных для решения поставленных задач. Пользователь данной системы не обязан быть экспертом в области интеллектуального анализа данных, т.к. он может не выбирать метод решения задачи вручную – ему необходимо лишь выбрать наилучшую обученную модель для использования среди тех, что предложила ему система. Во-вторых, система актуальна с исследовательской стороны, т.к. для решения одной задачи будут применяться различные модели, методы обучения задачи и алгоритмы предобработки данных. Будут доступны для анализа параметры этих методов и мера эффективности каждой обученной модели. Автор данной работы надеется, что это поспособствует исследованию эффективности методов ИАД при решении целых классов задач.

# Постановка задачи

Интеллектуальный анализ данных представляет собой итеративный процесс сбора данных, их предобработки и трансформации, построения решающей модели и интерпретации результатов, полученных при использовании этой модели (см. рис. 1).

Рисунок . Схема интеллектуального анализа данных

*Необработанные данные*

*Предобработанные данные*

*Трансформированные данные*

*Решающая модель*

*Собранные данные*

*Знания*

Сбор данных

ПРЕДОБРАБОТКА ДАННЫХ

ОБУЧЕНИЕ / РЕШЕНИЕ

ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТА

Удаление шума

Извлечение признаков

Нормализация

Уменьшение размерности

У

Классификация

Кластеризация

Визуализация

Несмотря на важность предобработки данных, которая безусловно влияет на эффективность методов решения задач ИАД, самым сложным как в вычислительном, так и в исследовательском плане является этап выбора наилучшей решающей модели. Существует множество решателей, и каждый из них «заточен» под решение конкретных типов задач. Так, например, деревья решений показывают отличные результаты на задачах классификации, но проигрывают в эффективности нейронным сетям на задачах восстановления регрессии.

Целью данной работы является:

1. Создание Инструментальной системы интеллектуального анализа данных (ИСИАД), содержащей расширяемые библиотеки методов анализа данных и позволяющей производить сравнительный анализ применения различных методов решения одной задачи.
2. Разработка библиотеки нейронных сетей как части ИСИАД, интеграции данной библиотеки с алгоритмами обучения.
3. Проведение вычислительного эксперимента по сравнению результатов, полученных на разных топологиях нейронных сетей.

Данный проект выполнялся в команде, поэтому 1-й пункт представленных выше целей является общим для каждого члена команды. Остальные цели относятся только к данной работе.

# Используемые методы ИАД

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*; –* вектор выходной информации. При этом каждая из компонент *,* иможет 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 значениедля конкретныхи *,* 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:

|  |
| --- |
|  |

Рассмотрим методы ИАД, реализованные в ИСИАД.

## 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. ИНС используются в задачах распознавания, классификации, кластеризации, восстановления зависимости, прогнозирования*,* отчасти в задачах ассоциациии 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. На рис. 2 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.

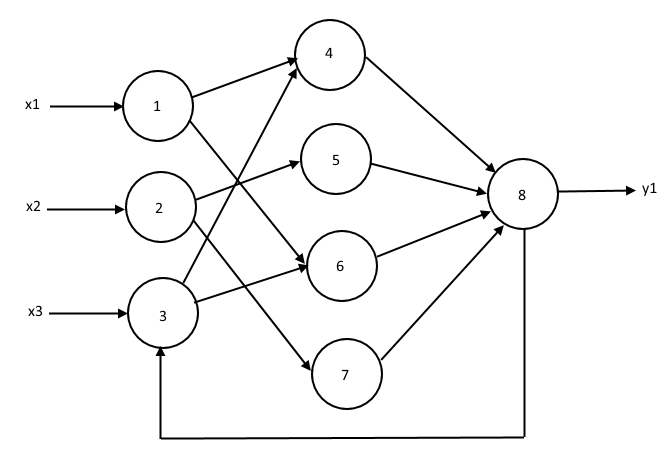


Рисунок . Пример нейронной сети

Теперь рассмотрим структуру вычислительной единицы ИНС – нейрона (см. рис. 3).

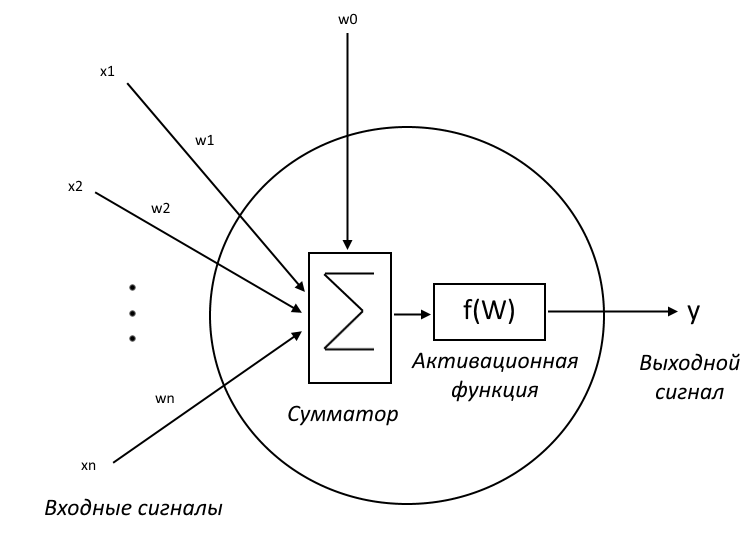


Рисунок . Искусственный нейрон

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:

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:

* 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.

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: сучителем 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.

Самыми популярными топологиями ИНС являются:

* Персептрон – для задач классификации и восстановления зависимости
* Сеть Кохонена – для задач классификации
* Сеть Хопфилда – для задач ассоциации
* Ограниченная машина Больцмана – применяется в глубоком обучении
* Сети адаптивного резонанса – для распознавания воздействия внешних факторов среды
* Сверточные нейронные сети – для распознавания образов

## Деревья решений

Дерево решений – способ представления набора продукционных правил[[1]](#footnote-1), извлеченных из данных, в последовательной, иерархической структуре. 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 определение кредитоспособности клиента банка приведен ниже (см. рис. 2).

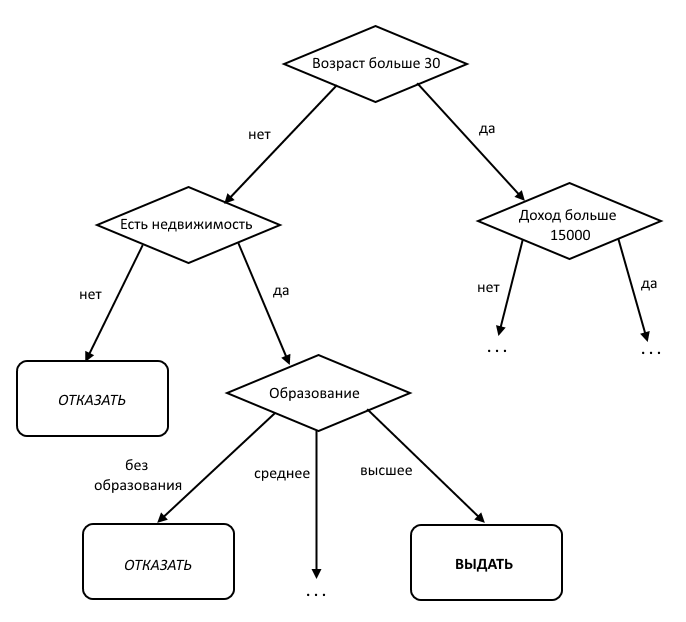


Рисунок .Пример дерева решений

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.

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 значениявпринимают значение *.* Тогда ДР, построенное на этой выборке, состоит только из листа, помеченного значением *.*
2. Значениявпринимаютразличныезначенияиз *.* В таком случае необходимо разделить выборку *.* Для этого некоторым образом (см. ниже критерий расщепления) выбирается входной атрибут *.* Допустим, все значения этого атрибута впринадлежат множеству *.* Тогда выборка разделяется на подвыборки *;* гдеСтроится узел дерева, помеченный вопросом «каково значение атрибута *?*»*.* Далее для каждой подвыборкипо 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 критерииостанова построения дерева, определяющие момент, когда дальнейшее построение дерева нецелесообразно. Рассмотрим самые популярные критерии останова [17]:

* 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.

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.

## Генетический алгоритм

Генетический алгоритм представляет собой механизм поиска, построенный на эвристических знаниях о естественном отборе в природе [19]. 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 [19]:

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.

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 [19]:

* 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.

# Общая 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 2013.

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:
   * Таблицы, хранящие информацию о задаче и ее выборках.
   * Группы таблиц, содержащих информацию о методах ИАД: обученные параметры методов, дополнительная информация.
   * 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. рис. 8).

*Задачи*

*Параметры ИНС*

*Качество обучения*

*Параметры ДР*

*Библиотека ИНС*

*Библиотека ДР*

*Библиотека алгоритмов обучения*

***База данных ИСИАД***

Рисунок . Схема структур хранения данных в ИСИАД

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. Выбрать тип нормализации значений атрибутов. Перед обучением значения всех атрибутов преобразовываются в числовой формат. На данный момент для каждого типа атрибута (перечисленный, целый, действительный) доступны 3 вида нормализации значений:
   * Простое числовое – перечисленное значение преобразовывается в номер этого значения, целое и действительное не меняются
   * Действительное нормализованное – все значения переводятся в интервал [0, 1]. Целые и перечисленные значения переводятся в значения, лежащие в центре отрезка с номером, равным естественному значению атрибута.
   * Целое нормализованное – действительное нормализованное, умноженное на , где – число значимых цифр, и приведенное к целому числу. Для целого и перечисленного атрибутов , где – количество различных значений атрибута во всей выборке. Для действительного атрибута где – минимальное расстояние между двумя соседними значениями атрибута во всей выборке.
5. Провести обучение выбранного методаИАД 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. рис. 9).

*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.

# Архитектура библиотеки нейронных сетей

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.

Рис. 10 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.

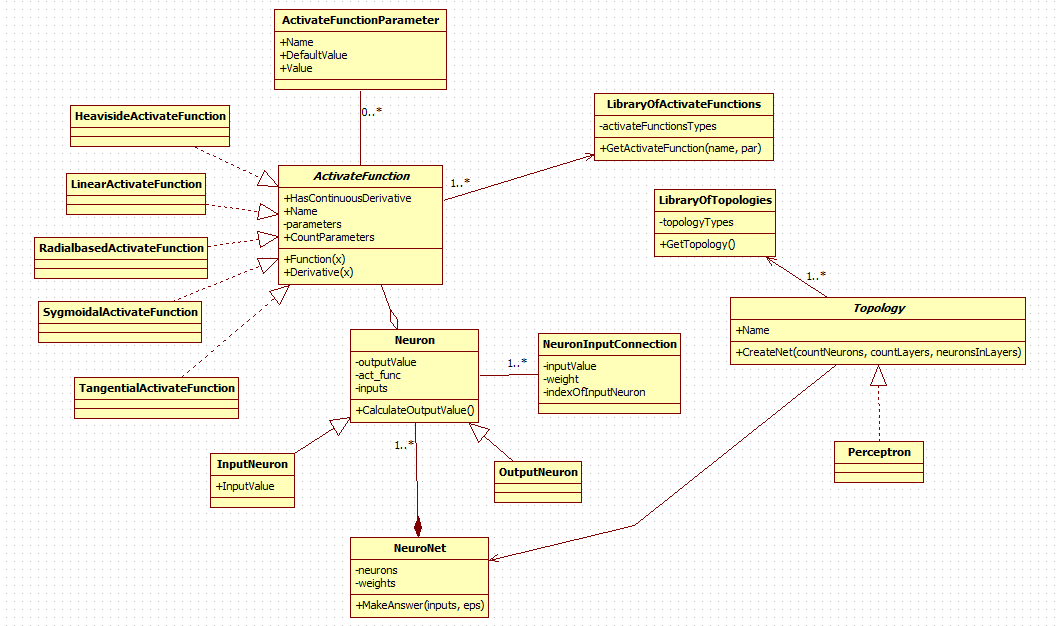


Рисунок . Диаграмма классов библиотеки ИНС

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[[2]](#footnote-2) 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[[3]](#footnote-3):
      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.

# Вычислительный эксперимент

В целях определения эффективности работы реализованных решателей и алгоритмов обучения были подобраны 5 задач для проведения вычислительного эксперимента. Вычислительный эксперимент заключается в обучении каждого решателя методом обратного распространения ошибки и в получении значения ошибки на обучающей и тестовой выборке. Такие же данные собираются для сторонних разработок в области ИАД – Matlab и scikit-learn [20, 21]. Результаты, полученные на ИСИАД, сравниваются с результатами применения сторонних систем. Параметры всех систем подбирались таким образом, чтобы условия их работы были максимально похожими друг на друга.

## Описание задач

При выборе задач для эксперимента автор руководствовался следующими принципами:

* Смысл задачи должен быть понятен человеку, не являющемуся специалистом в предметной области задачи. Для этого собранные данные должны содержать небольшое число атрибутов (меньше 15); атрибутов с узкоспециальным смыслом должно быть мало.
* Эксперимент должен содержать как классические, хорошо известные задачи, так и новые, актуальные в момент написания работы.
* Эксперимент должен содержать как задачи классификации, так и восстановления зависимости.
* Эксперимент должен содержать задачи с разным объемом выборок и разным числом атрибутов. Т.е. должны быть «большие», «средние» и «малые» по объему выборок и по числу атрибутов задачи.

Поиск задач осуществлялся в Интернет-ресурсе UC Irvine Machine Learning Repository. Для эксперимента были выбраны следующие задачи:

* Ирис
* Морское ушко
* Пожары в лесах
* Комната
* Кожа лица

Рассмотрим каждую задачу.

### Ирис

Ирис — род многолетних корневищных растений семейства. Ирисы встречаются на всех континентах. Род насчитывает около 800 видов с богатейшим разнообразием форм и оттенков*.* Ирис является одной из самых известных задач в области интеллектуального анализа данных. Задача основана на исследовании Р.А. Фишера [22], используемые для эксперимента данные были собраны в 1988 году.

Смысл задачи: Необходимо определить тип ириса по значениям входных атрибутов.

Входные атрибуты:

1. Длина чашелистика в сантиметрах (минимальное значение: 4.3, максимальное: 7.9, среднее: 5.84).
2. Ширина чашелистика в сантиметрах (минимальное значение: 2.0, максимальное: 4.4, среднее: 3.05).
3. Длина лепестка в сантиметрах (минимальное значение: 1.0, максимальное: 6.9, среднее: 3.76).
4. Ширина лепестка в сантиметрах (минимальное значение: 0.1, максимальное: 2.5, среднее: 0.76).

Все входные атрибуты имеют действительный тип данных.

Выходной атрибут: Класс ириса. Тип перечисленный, может принимать 3 значения – Iris setosa, Iris versicolour, Iris virginica.

Тип задачи: Классификация.

Размер выборки: 150 записей, по 50 записей для каждого класса.

Особенность задачи заключается в том, что данные для класса Iris setosa линейно отделимы от данных для двух других классов. Однако данные для Iris versicolour и Iris virginica линейно неотделимы друг от друга. Следовательно, для правильного распознавания первого типа ириса (Iris setosa) достаточно однослойного персептрона, но для распознавания двух остальных классов необходимо наличие скрытых слоев в обучаемой НС.

### Морское ушко

Морские ушки, или галиотисы — род брюхоногих моллюсков*.* Некоторые виды употребляются в пищу, раковины идут на украшения*.* Данные по этой задаче были получены С. Ваухом для Морской исследовательской лаборатории Австралии в 1995 году [23].

Смысл задачи: Необходимо определить возраст морского ушка по набору входных атрибутов.

Входные атрибуты:

1. Пол морского ушка (мужской, женский, младенец).
2. Максимальная длина раковины в миллиметрах (минимальное значение: 0.075, максимальное: 0.815, среднее: 0.524).
3. Диаметр в миллиметрах – длина раковины по оси, перпендикулярной той, по которой измеряется максимальная длина раковины (минимальное значение: 0.055, максимальное: 0.650, среднее: 0.408).
4. Высота в миллиметрах (минимальное значение: 0, максимальное: 1.130, среднее: 0.140).
5. Полный вес в граммах (минимальное значение: 0.002, максимальное: 2.826, среднее: 0.829).
6. Вес мяса в граммах (минимальное значение: 0.001, максимальное: 1.488, среднее: 0.359).
7. Вес внутренностей в граммах (минимальное значение: 0.001, максимальное: 0.760, среднее: 0.181).
8. Вес раковины в граммах (минимальное значение: 0.002, максимальное: 1.005, среднее: 0.239).

Первый атрибут имеет перечисленный тип, остальные – действительный.

Выходной атрибут: Число колец морского ушка. Имеет целочисленный тип данных. Минимальное значение: 1, максимальное: 29, среднее: 9.934. По количеству колец можно однозначно установить возраст морского ушка в годах:

Тип задачи: Восстановление зависимости.

Размер выборки: 4177 записей. Значения выходного атрибута распределены по выборке неравномерно. На рис. 11 изображена зависимость количества записей в обучающей выборке от значения выходного атрибута.

Рисунок . Распределение значения выходного атрибута

По оси ординат в логарифмической шкале содержится число:

т.к. значение 28 колец не представлено во всей выборке. По оси абсцисс содержатся значения выходного атрибута.

### Пожары в лесах

Задача «Пожары в лесах» была создана П. Кортезом и А. Мораисом в 2007 году в целях изучения эффективности различных методов машинного обучения [24].

Смысл задачи: Необходимо определить площадь возгорания в парке Монтесино (Испания, провинция Аликанте, г. Лос-Монтесинос) по метрологическим данным.

Входные атрибуты:

1. Х-координата в парке (от 1 до 9)
2. Y-координата в парке (от 2 до 9)
3. Месяц
4. День
5. FFMC индекс системы FWI (от 18.7 до 96.2)
6. DMC индекс системы FWI (от 1.1 до 291.3)
7. DC индекс системы FWI (от 7.9 до 860.6)
8. ISI индекс системы FWI (от 0.0 до 56.10)
9. Температура в Цельсиях (от 2.2 до 33.3)
10. Относительная влажность в процентах (от 15 до 100)
11. Скорость ветра в км/ч (от 0.4 до 9.4)
12. Осадки в мм/м2 (от 0.0 до 6.4)

Первые два атрибута целочисленные, следующие два – перечисленные, остальные – действительные.

Выходной атрибут: площадь горящего леса в гектарах (от 0 до 1090.84)

Тип задачи: Восстановление зависимости.

Размер выборки: 517 записей. Значения выходного атрибута распределены по выборке очень неравномерно (см. рис 12) – большинство записей соответствует нулевой площади возгорания. При этом разброс значений этого атрибута также велик.

Рисунок .Распределение значений выходного атрибута

### Комната

Данные для этой задачи были собраны в 2015 году [25].

Смысл задачи: необходимо определить, находятся ли в комнате люди на основании данных от тепловых, световых и других датчиков, установленных внутри комнаты.

Входные атрибуты:

1. Дата, время (от 11.02.15 14:48 до 18.02.15 9:19)
2. Температура в градусах Цельсия
3. Влажность
4. Количество света
5. Концентрация CO2
6. Относительная влажность

Все атрибуты кроме первого имеют действительный тип данных.

Выходной атрибут: Наличие людей в комнате – бинарный тип (0 – нет, 1 - да).

Тип задачи: Бинарная классификация

Размер выборки: 9752 записи.

### Кожа лица

Данные для этой задачи были собраны в 2012 году с помощью случайной выборки точек на снимках лиц людей разных возрастов, рас и полов.

Смысл задачи: по цвету точки определить, является ли он цветом кожи лица человека.

Входные атрибуты: Красная, Синяя и Зеленая составляющие цвета точки (RGB-спектр). Все атрибуты имеют целочисленный тип данных, их значения заключены в отрезке [0, 255].

Выходной атрибут: Является ли цвет заданной точки цветом кожи лица – бинарный тип (0 – нет, 1 - да).

Тип задачи: Бинарная классификация.

Размер выборки: 101718 значений. Половина записей соответствуют точкам кожи лица, а другая половина – остальным точкам на фотографии.

## Условия проведения эксперимента

Автор данной работы проводил эксперимент на нейронных сетях с использованием метода обратного распространения ошибки. Для того, чтобы осуществить сравнение результатов сторонних систем с результатами ИСИАД необходимо обеспечить работу всех этих систем в максимально похожих условиях. Поэтому было решено ограничиться 2-мя топологиями персептрона для решения каждой задачи и зафиксировать ряд параметров применяемого метода обучения (см. таблицу 2).

Таблица . Параметры проведения эксперимента

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Задача** | **Персептрон 1** | | **Персептрон 2** | | **Скорость обучения** |
| ***Топология*** | ***Количество итераций*** | ***Топология*** | ***Количество итераций*** |
| Ирис | 4-6-6-1 | 4000 | 4-6-7-6-1 | 6000 | 0,5 |
| Морское ушко | 8-11-11-1 | 8000 | 8-11-13-11-1 | 12000 | 0,5 |
| Пожары в лесах | 12-14-14-1 | 12000 | 12-14-16-14-1 | 18000 | 0,5 |
| Комната | 9-11-11-1 | 9000 | 9-11-13-11-1 | 13500 | 0,5 |
| Кожа лица | 3-5-5-1 | 3000 | 3-5-6-5-1 | 4500 | 0,5 |

Все персептроны, применяемые в экспериментах, имеют сигмоидальную функцию активации. Число нейронов в слоях и максимальное количество итераций выбирались по эмпирическим формулам:

1. Число нейронов по слоям выбиралось как:
   1. Входной слой – количество входных параметров (это значение фиксировано и зависит только от задачи)
   2. Скрытый слой 1 – , где – число нейронов в предыдущем слое, – коэффициент, принимающий значения в отрезке [1.2, 1.5]. Значение данного коэффициента подбиралось для каждой задачи в зависимости от числа входных атрибутов – чем их больше, тем меньше коэффициент. Такое правило было выбрано для ограничения роста нейронной сети «вширь». Таким образом для каждой из задач данный коэффициент принимает следующие значения:
      1. Ирис -
      2. Морское ушко -
      3. Пожары -
      4. Комната -
      5. Кожа лица -
   3. Скрытый слой 2 (для 2-й топологии) – , где – число нейронов в предыдущем слое
   4. Скрытый слой 3 (для 2-й топологии), скрытый слой 2 (для 1-й топологии) – число нейронов такое же, как в скрытом слое 1.
   5. Выходной слой – 1 нейрон (фиксированное значение).
2. Число шагов метода обратного распространения ошибки выбиралось в зависимости от сложности задачи и топологии следующим образом:

где -число входных атрибутов, – число слоев в нейронной сети.

Данные для всех задач перед обучением были преобразованы: значения каждого атрибута были приведены к нормализованному вещественному представлению.

## Результаты

Для получения результатов эксперимента выборки всех задач были разделены на обучающую и тестовую выборки случайным образом в пропорции 4:1. После этого все системы были обучены на первой выборке. На каждой итерации вычислялась ошибка решателя на тестовой и обучающей выборке. За результат бралась пара ошибок, чья сумма была минимальна за все итерации обучения решателя.

Значение ошибки есть процент неправильных ответов решателя. Для сравнения результата, полученного решателем, с правильным ответом осуществлялись следующие действия:

1. Преобразование полученного и правильного ответов из нормализованного действительного формата в тот формат, в котором выходной атрибут хранится в БД (действительный, целый или перечисленный).
2. Производится сравнение полученных значений:
   1. Если тип атрибута действительный, значения считаются равными, если выполняется неравенство где – правильный ответ, – полученный ответ, – минимальное расстояние между двумя соседними значениями выходного атрибута во всей выборке (в объединении обучающей и тестовой выборок).
   2. Целые и перечисленные значения считаются равными, если .

Опишем результаты для каждой задачи по отдельности.

### Ирис

Ошибки, полученные для всех систем на задаче «Ирис», представлены в таблице 3.

Таблица . Результаты на задаче Ирис

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Персептрон 1** | | | **Персептрон 2** | |
| ***ИСИАД*** | ***Matlab*** | ***ИСИАД*** | | ***Matlab*** |
| Обучающая выборка | 5,83 | 2,5 | 3,34 | | 3,33 |
| Тестовая выборка | 6,67 | 3,33 | 3,34 | | 6,67 |
| Количество итераций (тыс.) | 3,947 |  | 1,554 | |  |

Из графического представления этих данных (см. рис. 13) видно, что реализация ИСИАД выдает результат в 2 раза хуже, чем в системе Matlab на персептроне 1 и, наоборот, в 2 раза лучший результат (если судить по тестовой выборке) на персептроне 2. При этом персептрон 2 затратил в 2 раза меньше итераций для достижения наилучшего результата, чем персептрон 1.

Рисунок . Результаты на задаче Ирис

### Морское ушко

Значения ошибки на выборе «Морское ушко» представлены на рис. 14. Несмотря на то, что достигнутые результаты лучше результатов Matlab на обеих топологиях, ошибка получилась большой.

Рисунок . Результаты на задаче Морское ушко

Это получилось из-за того, что выходное значение атрибута имеет 29 различных значений и распределено по гауссовому закону (см. рис. 11). Таким образом, выборка не является достаточно репрезентативной для слишком больших и слишком малых значений выходного атрибута – нейронные сети просто «не запоминают» эти значения.

С другой стороны, на рис. 15 видно, что для большинства записей отклонение полученного в результате решения ответа от правильного значения выходного атрибута лежит в полосе от -2 до 2.

Рисунок . Различие правильного и полученного значения атрибута на тестовой выборке

Таким образом, если заменить критерий сравнения результатов с на , где – параметр, ошибка резко уменьшается уже при (см. таблицу 4), что говорит о том, что нейронная сеть все-таки нашла зависимость значения выходного атрибута от данных.

Таблица . Зависимость ошибки на обучающей выборке от допуска при сравнении

|  |  |
| --- | --- |
| Допуск | Ошибка |
| 0 | 72,53 |
| 1 | 32,97 |
| 2 | 16,09 |
| 3 | 8,52 |
| 4 | 4,84 |

### Пожары в лесах

Пожары в лесах – самая «трудная» из всех представленных задач – при ее решении ни одна нейронная сеть как в ИСИАД, так и в Matlab, не смогла найти зависимости данных от значений выходного атрибута (см. таблицу 5).

Таблица . Результаты на задаче Пожары в лесах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Персептрон 1** | | **Персептрон 2** | |
| ***ИСИАД*** | ***Matlab*** | ***ИСИАД*** | ***Matlab*** |
| Обучающая выборка | 100 | 100 | 99,75 | 100 |
| Тестовая выборка | 99,02 | 100 | 100 | 100 |
| Количество итераций (тыс.) | 4,849 |  | 3,001 |  |

Это произошло из-за следующих причин:

* Сложной зависимости в данных – наблюдается «паралич» нейронной сети – алгоритм обучения не находит значений весов, которые давали бы меньше, чем 100% ошибки (см. рис. 16).
* Большого разброса значений выходного атрибута
* Большинство записей в обучающей выборке соответствуют одному значению выходного атрибута (нулевая площадь возгорания) – выборка не репрезентативна.
* Неэффективности приведения значений выходного атрибута к вещественному нормализованному формату – расстояние между нормализованными значениями выходного атрибута слишком малы. При конвертировании теряется большое различие между некоторыми значениями (см. таблицу 6).

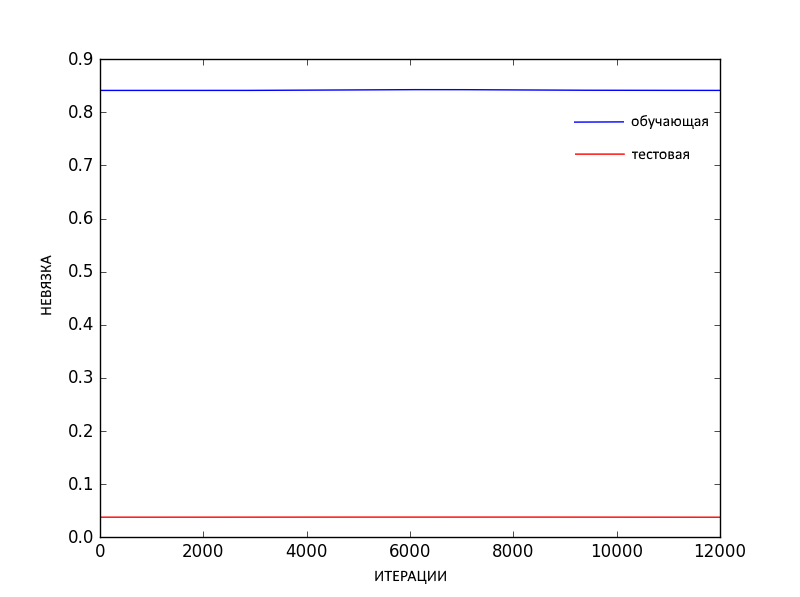


Рисунок . Невязка на задаче пожары в лесах, Персептрон 1

Таблица . Значения выходного атрибута

|  |  |  |  |
| --- | --- | --- | --- |
| Правильное | Полученное | Правильное нормализованное | Полученное нормализованное |
| 0 | 14,12857053 | 0 | 0,01295201 |
| 8,85 | 13,54917231 | 0,008113014 | 0,012420861 |
| 0,79 | 13,29557196 | 0,000724213 | 0,01218838 |
| 8,98 | 15,18384499 | 0,008232188 | 0,013919406 |
| 17,85 | 14,80637571 | 0,016363536 | 0,013573371 |
| 22,03 | 14,80637571 | 0,020195446 | 0,013573371 |
| 1,1 | 16,13902416 | 0,001008397 | 0,014795042 |
| 24,24 | 14,97067696 | 0,022221407 | 0,01372399 |
| 0 | 15,8068499 | 0 | 0,01449053 |
| 0 | 13,99629225 | 0 | 0,012830747 |
| 0 | 14,66900437 | 0 | 0,013447439 |
| 0 | 15,66064458 | 0 | 0,0143565 |
| 0 | 15,60233306 | 0 | 0,014303044 |
| 0 | 15,01704967 | 0 | 0,013766501 |
| 3,52 | 15,18061966 | 0,003226871 | 0,013916449 |
| 0 | 13,24433695 | 0 | 0,012141411 |
| 0 | 13,67579241 | 0 | 0,012536937 |
| 4,41 | 13,94038473 | 0,004042756 | 0,012779495 |
| 34,36 | 14,02234891 | 0,031498662 | 0,012854634 |
| 2,18 | 14,15565931 | 0,00199846 | 0,012976843 |
| 4,42 | 15,15584726 | 0,004051923 | 0,01389374 |

### Комната

Результаты для данной задачи представлены в таблице 7. Matlab и ИСИАД показывают похожие, очень низкие значения ошибки. Можно считать, что зависимость нейронными сетями найдена полностью.

Таблица . Результаты на задаче Комната

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Персептрон 1** | | **Персептрон 2** | |
| ***ИСИАД*** | ***Matlab*** | ***ИСИАД*** | ***Matlab*** |
| Обучающая выборка | 0,6152 | 0,56 | 0,5767 | 0,54 |
| Тестовая выборка | 0,6153 | 0,55 | 0,51282 | 0,56 |
| Количество итераций (тыс.) | 0,962 |  | 1,183 |  |

### Кожа лица

Результаты для этой задачи представлены в таблице [*номер*].

Таблица . Результаты на задаче Кожа лица

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Персептрон 1** | | **Персептрон 2** | |
| ***ИСИАД*** | ***Matlab*** | ***ИСИАД*** | ***Matlab*** |
| Обучающая выборка | 0,1425 | 0,39 | 0,1302 | 0,43 |
| Тестовая выборка | 0,1523 | 0,4 | 0,113 | 0,42 |
| Количество итераций (тыс.) | 0,103 |  | 0,141 |  |

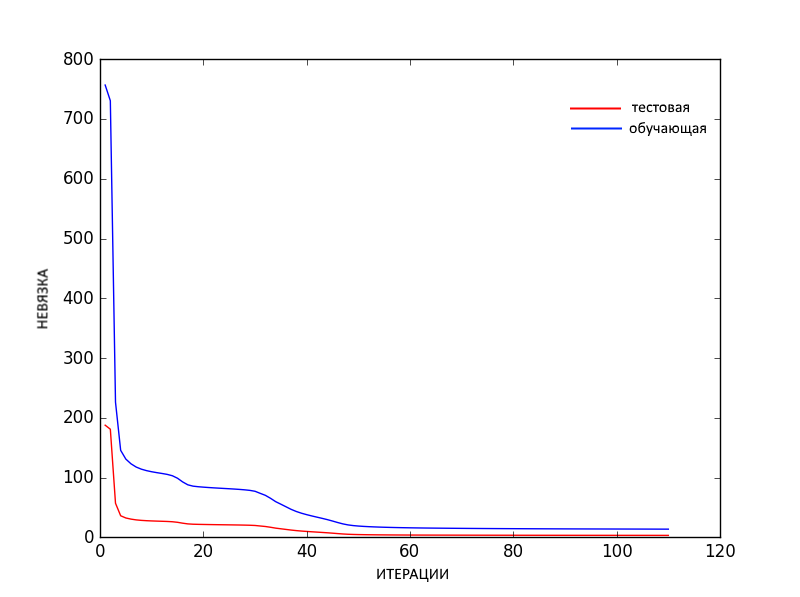
На рис. 17 видно, что реализация ИСИАД показала намного лучший результат, чем Matlab.

Рисунок . Результаты на задаче Кожа лица

Исходя из графиков невязки и ошибки (рис. 18 и рис. 19), можно утверждать, что:

1. Начиная с 50-й итерации ИНС полностью обучилась – не происходит заметного уменьшения невязки.
2. Минимизируемый методом обучения функционал и ошибка, в общем говоря, различные критерии, и минимизация функционала не всегда приводит к уменьшению ошибки (что подтверждают значения на графиках в первые 5 итераций обучения).

Рисунок . Невязка на задаче Кожа лица



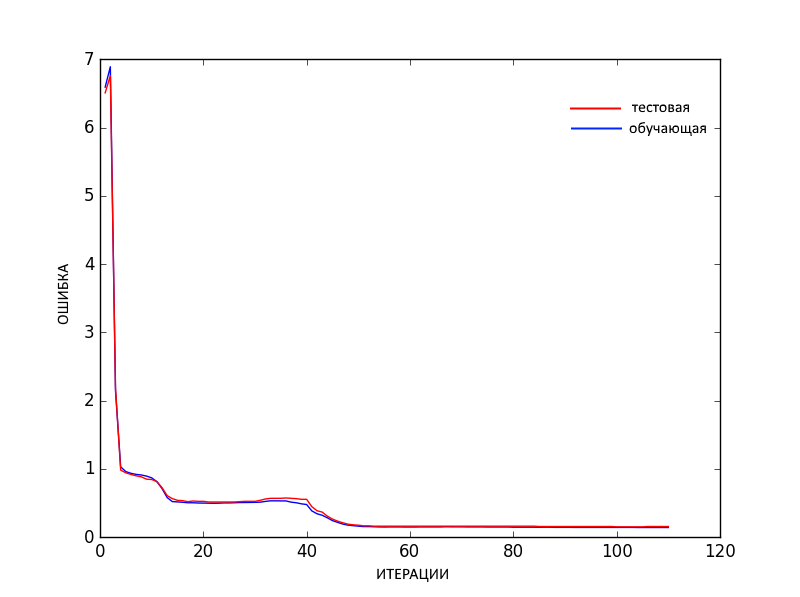


Рисунок . Ошибка на задаче Кожа лиц

### Сравнение результатов топологии по всем задачам

Данные, представленные выше, можно также интерпретировать другим образом: сравнить результаты конкретных топологий по всем задачам (см. рис. 20 и рис. 21).

Рисунок . Топология Персептрон 1

Рисунок . Топология Персептрон 2.

Из приведенных графиков видно, что результаты Matlab и ИСИАД сопоставимы. При этом на персептроне 1 результаты ИСИАД чуть лучше для 2-х задач, а на персептроне 1 – для 3-х задач.

# Заключение

В рамках данной работы был создан проект ИСИАД, позволяющий решать задачи ИАД, упростить последние этапы ИАД, провести сравнение различных методов решения задач и различных алгоритмов обучения применительно к конкретной задаче. Была разработана библиотека нейронных сетей как часть ИСИАД. Была создана топология «Персептрон» и произведена ее интеграция с решателями - генетическим алгоритмом и методом обратного распространения ошибки. Был осуществлен вычислительный эксперимент: были подобраны исследовательские задачи, на которых осуществлялось обучение алгоритмов ИСИАД и сторонних разработок (Matlab – нейронные сети, scikit-learn – деревья решений). Было произведено сравнение результатов разработанной системы с результатами сторонних разработок.

Полученные результаты показали, что разработанная библиотека нейронных сетей не уступает по качеству решения задач системе Matlab, а на некоторых задачах даже превосходит ее. Также из результатов видно, что система легко справляется с задачами классификации (Ирис, Комната, Кожа лица), но не справляется с задачами восстановления регрессии (Морское ушко, Пожары в лесах).

По результатам данной работы были сформированы следующие планы по дальнейшему совершенствованию системы:

* Добавление новых топологий в библиотеку нейронных сетей.
* Дальнейшая разработка системы предобработки данных.
* Создание дополнительных способов разделения данных на тестовые и обучающие выборки (например, кросс-валидация).
* Модернизация метода обратного распространения ошибки (например, применение адаптивного алгоритма выбора скорости обучения), добавление новых алгоритмов обучения.
* Улучшение результатов на задачах восстановления регрессии
* Оптимизация скорости обучения нейронных сетей.

# Список литературы

1. Witten, I.H. Data Mining: Practical machine learning tools and techniques / I.H. Witten, E. Frank, M.A. Hall – Burlington: Elsevier, 2011.
2. Haykin, S. Neural Networks. A comprehensive foundation / 2nd edition – New Jersey: Prentice Hall, 1999.
3. Breiman, L. Classification and regression trees / L. Breiman, J.H. Friedman, R.A. Olshen // Wadsworth & Brooks/Cole Advanced Books & Software – 1984.
4. Breiman, L. Random forests // Machine Learning – 2001. – 45.
5. Russel, S.J. Artificial Intelligence: A Modern Approach / S.J. Russel, P. Norvig – 2nd edition – New Jersey: Prentice Hall, 2003.
6. Ansari, U. Predictive Data Mining for Medical Diagnosis: An Overview of Heart Disease Prediction / U. Ansari, S. Soni // International Journal of Computer Applications – 2011.
7. Srinivas, K. Applications of Data Mining Techniques in Healthcare and Prediction of Heart Attacks / K. Srinivas, B. Kavihta Rani, A. Govrdhan // International Journal on Computer Science and Engineering – 2010. – 2.
8. Ahn, H. Facilitating cross-selling in a mobile telecom market to develop customer classification model based on hybrid data mining techniques / H. Ahn, J.J. Ahn, K.J. Oh, D.H. Kim // Expert Systems with Applications – 2011. – 38(5).
9. Rivas, T. Explaining and predicting workplace accidents using data-mining techniques / T. Rivas, M. Paz, J.E. Martin, J.M. Matias // Reliability Engineering and System Safety – 2011. – 96(7).
10. Fu, T.C. A review on time series data mining // Engeenering applications of artificial intelligence – 2011. – 24.
11. Liu, H. Feature Selection: An ever evolving frontier in Data Mining / H. Liu, H. Motoda, R. Setiono, Z. Zhao // JMLR: Workshop and Conference Proceedings – 2010. – 10.
12. Alcala-fdez, J. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework / J. Alcala-fdez, A. Fernandez, J.Luengo, J.Derrac, S.Garcia, L. Sanchez, F.Herrera // Old City Publishing, Inc. – 2011. – 17.
13. Cortez, P. Data Mining with Neural Networks and Support Vector Machines using the R/rminer Tool // University of Minho – 2010.
14. Low, Y. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud // Carnegie Mellon University – 2012.
15. Dahl, G.E. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition // IEEE transactions on audio, speech, and language processing – 2012.
16. Krizhevsky, A. ImageNet classiﬁcation with deep convolutional neural networks // A. Krizhevsky, I. Sutskever, G.E. Hinton // University of Toronto – 2013.
17. Чубукова И.А. Data Mining. – Режим доступа: http://www.intuit.ru/studies/courses/6/6/info, свободный.
18. Нейронные сети и их устройство. – Режим доступа: <http://ap-economics.narod.ru/library/NLP.html>, свободный.
19. 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. Neural Networrk Toolbox. Matlab. – Режим доступа:

http://www.mathworks.com/products/neural-network/?requestedDomain=www.mathworks.com, свободный.

1. Scikit-learn – Режим доступа: http://scikit-learn.org/stable/, свободный.
2. Fisher, R.A. The use of multiple measurements in taxonomic problems // Annual Eugenics – 1936. – 7, Part II, 179-188.
3. Waugh, S. Extending and benchmarking Cascade-Correlation // PhD thesis, Computer Science Department, University of Tasmania – 1995.
4. Cortez, P. A Data Mining Approach to Predict Forest Fires using Meteorological Data / Cortez P., Morais A.// In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, Guimaraes, Portugal – 2007 – pp. 512-523.
5. Candanedo, L. M. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models // Energy and Buildings – 2016 – Volume 112, Pages 28-39.

1. Продукционное правило – знание, представленное в виде логической конструкции «Если …, то …». Например, «Если идет дождь, следует взять зонтик». [↑](#footnote-ref-1)
2. 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-2)
3. 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-3)