МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Национальный исследовательский**

**Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра информатики и автоматизации научных исследований**

Направление подготовки: «Прикладная информатика»

Профиль подготовки: «Прикладная информатика в области принятия решений»

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

на тему:

**«Нейросетевой модуль гибридной интеллектуальной системы»**

**Выполнил:**

студент группы 381607м

Алявдин Александр Владимирович

Подпись\_\_\_\_\_\_\_\_\_\_

**Научный руководитель:**

к.т.н., доцент Басалин П.Д.

Подпись\_\_\_\_\_\_\_\_\_\_

Нижний Новгород  
2018

Оглавление

[Введение 3](#_Toc515530878)

[1. Постановка задачи 5](#_Toc515530879)

[2. Представление знаний о предметной области на инфологическом уровне 6](#_Toc515530880)

[2.1. Дерево решений как формализм описания знаний на инфологическом уровне 7](#_Toc515530881)

[2.2. Граф решений 8](#_Toc515530882)

[3. Гибридная система интеллектуальной поддержки процессов принятия решений 10](#_Toc515530883)

[3.1. Концепция системы, основанной на знаниях 10](#_Toc515530884)

[3.2. Нейросетевой подход к построению интеллектуальной системы 11](#_Toc515530885)

[3.3. Концепция гибридной системы интеллектуальной поддержки и её основные компоненты 13](#_Toc515530886)

[4. Нейронная сеть как основа механизма вывода гибридной системы интеллектуальной поддержки 17](#_Toc515530888)

[4.1. Многослойный персептрон 17](#_Toc515530889)

[4.2. Активационные функции нейронов 19](#_Toc515530890)

[5. Нейросетевой модуль гибридной системы интеллектуальной поддержки 23](#_Toc515530891)

[5.1. Основные компоненты нейросетевого модуля 23](#_Toc515530892)

[5.2. Структура и топология нейронной сети 24](#_Toc515530893)

[5.2.1. Сенсорный слой 25](#_Toc515530894)

[5.2.2. Скрытые слои 25](#_Toc515530895)

[5.2.3. Моторный слой 26](#_Toc515530896)

[5.3. Алгоритм построения нейронной сети по графу решений 26](#_Toc515530897)

[5.4. Пример построения нейронной сети по графу решений из области диагностики неполадок сетевого соединения 27](#_Toc515530898)

[6. Программная реализация 29](#_Toc515530899)

[6.1. Общее описание программной реализации 29](#_Toc515530900)

[6.2. Архитектура программной системы 29](#_Toc515530901)

[6.2.1. Диаграмма классов компонента нейронной сети 29](#_Toc515530902)

[6.2.2. Диаграмма классов компонента построения нейронной сети 30](#_Toc515530903)

[6.2.3. Диаграмма классов компонента дообучения нейронной сети 30](#_Toc515530904)

[6.3. Интеллектуальный интерфейс 31](#_Toc515530905)

[6.4. Компонент построения нейронной сети по графу решений 31](#_Toc515530906)

[6.5. Компонент нейросетевого вывода 32](#_Toc515530907)

[7. Полученные результаты 33](#_Toc515530908)

[7.1. Тестовый базис 33](#_Toc515530909)

[7.1.1. Фрагмент графа решений для области диагностики неполадок сетевого подключения 33](#_Toc515530910)

[7.1.2. Результаты работы нейросетевого модуля 34](#_Toc515530912)

[8. Сравнение результатов работы нейросетевого и аналитического вывода 35](#_Toc515530913)

[Заключение 36](#_Toc515530914)

[Список литературы 37](#_Toc515530915)

[Приложение 39](#_Toc515530916)

Введение

Во многих областях своей деятельности человеку приходится сталкиваться с задачами трудноформализуемого и неформального плана, алгоритмы (сценарии) решения которых заранее не известны.

К таким задачам относятся задачи, математически формулируемые, но не имеющие априори известного алгоритма решения, гарантирующего получение приемлемого результата в реальном масштабе времени, особенно в случае их высокой размерности (NP-трудные проблемы). К таким задачам относятся, в частности, задачи принятия решений в предметных областях, связанных с диагностикой, прогнозированием процессов, управлением сложными системами.

Помочь человеку в решении этих задач способна *система интеллектуальной поддержки*, объединяющая возможности компьютера со знаниями и опытом эксперта в такой форме, что она способна предложить «разумный» совет или осуществить «разумное» решение поставленной задачи. Такая система должна уметь рассуждать при сомнительных, неполных данных, объяснять ход своих рассуждений понятным пользователю способом, самообучаться и адаптироваться к конкретным условиям применения.

Концепция системы, основанной на знаниях (СОЗ), позволяет добиться этой цели, обеспечивает открытость и масштабируемость системы, однако аналитический механизм получения решения в таких системах достаточно громоздкий и не всегда срабатывает в реальном масштабе времени.

Для решения данной проблемы предлагается использовать нейронную сеть, что значительно повышает производительность системы, кроме того, появляется возможность интегрировать аналитический модуль системы с нейросетевым, используя результаты работы аналитического модуля для дообучения нейронной сети.

*В первом разделе* данной работы поставлены основные цели и задачи.

*Во втором разделе* рассмотрены концепции представления знаний на инфологическом уровне, их преимущества и недостатки.

*В третьем разделе* описана концепция гибридной системы интеллектуальной поддержки, концепция системы, основанной на знаниях, которая является базовой для гибридной системы, а также основные компоненты гибридной системы.

*В четвертом разделе* рассмотрен многослойный персептрон - структура нейронной сети, на базе которой предлагается создать нейронную сеть для нейросетевого модуля гибридной системы. Также в этом разделе рассмотрены основные типы активационных функций нейронов.

*В пятом разделе* описан подход создания нейросетевого модуля гибридной системы интеллектуальной поддержки, его основные компоненты, а также механизм построения и структура нейронной сети, которые лежат в его основе. Также в этом разделе рассмотрены концепции дообучения сети.

*В шестом разделе* описана программная реализация модуля, его архитектура, диаграммы классов и детали реализации.

*В седьмом разделе* описан тестовый базис в виде графа решений для предметной области диагностики неполадок сетевого подключения ПК и описан тестовый базис для ручного тестирования корректности результатов нейронной сети.

*В восьмом разделе* представлено сравнение результатов работы нейросетевого и аналитического модулей гибридной системы, запущенных на тестовом графе решений из предметной области диагностики неполадок сетевого подключения.

1. Постановка задачи

Основная цель данной работы - создать модуль, позволяющий принимать решения в различных предметных областях на основании нейросетевого механизма вывода и, при необходимости, использовать результаты работы аналитического модуля для дообучения нейронной сети.

Для достижения данной цели поставлены следующие задачи:

1. **Разработать подход к созданию нейросетевого модуля гибридной системы интеллектуальной поддержки.** В рамках данной задачи, необходимо рассмотреть концепцию создания системы, основанной на знаниях и гибридной интеллектуальной системы. Разработать алгоритм преобразования знаний в эквивалентную по «рассуждениям» нейронную сеть, а также сформировать структуру нейронной сети и способ дообучения нейронной сети
2. **Разработать архитектуру нейросетевого модуля.** В рамках данной задачи необходимо описать основные компоненты нейросетевого модуля гибридной системы интеллектуальной поддержки, разработать программную архитектуру, диаграммы классов.
3. **Создать программную реализацию.** В рамках данной задачи необходимо определиться со средой разработки, языком, используемыми форматами файлов и библиотеками, написать и протестировать программу, создать тестовую инфраструктуру.
4. **Сравнить результаты работы нейросетевого и аналитического модулей гибридной системы.** В рамках данной задачи нужно сравнить полученные результаты двух модулей и проанализировать их.

2. Представление знаний о предметной области на инфологическом уровне

*Данные* и *знания* – это две категории информационного обеспечения задач, принципиально отличающиеся друг от друга ролью, выполняемой ими в вычислительном процессе.

*Данные* можно определить как информацию, полученную в результате наблюдений или измерений отдельных свойств объектов предметной области. Они используются при интерпретации *процедурных знаний* (знаний, заложенных в программах) и имеют удобную для этого структуру (простую или сложную).

*Знания* – это информация, отражающая полученные эмпирическим путем связи и закономерности предметной области, обеспечивающие возможность решения возникающих в ней задач.

Знания можно рассматривать как *данные о данных*, несущие информацию о том, каким образом последние следует интерпретировать.

Можно выделить ряд базовых свойств, отличающих знания от обычных данных [3, 7].

*Внутренняя интерпретируемость*. Это свойство знания отображать в своей структуре всю информацию, необходимую для его содержательной интерпретации, т.е. семантику знания.

*Структурированность и связность*. Данное свойство предполагает возможность представления информационных единиц знания в виде иерархических (рекурсивно вложенных) структур и установления между ними внешних связей, выражающих различные типы отношений (отношения структуризации, каузальные отношения, пространственные и временные отношения, отношения ситуационной близости и другие). Обозначенные связи позволяют эффективно решать проблемы наследования информации, использования обобщенных процедур, сужения пространства поиска знаний, релевантных рассматриваемой ситуации, и т.д.

*Шкальная и ассоциативная упорядоченность*. Для взаимного упорядочения информационных единиц могут использоваться метрические (абсолютные и относительные) шкалы, порядковые шкалы (например, шкала оценок успеваемости студента), нечеткие порядковые шкалы (например, шкала «никогда – всегда» с рядом промежуточных значений: «почти никогда», «редко», «часто», «почти всегда»), оппозиционные шкалы (например, шкала «добрый – злой» с третьим, нейтральным, значением «не добрый — не злой»).

*Активность*. Это свойство определяет способность знания активизировать систему на выполнение тех или иных действий. Знания предписывающего типа активизируют систему посредством программ, в которых они заложены. Декларативные знания активизируют СОЗ через программное ядро, основой которого являются механизм вывода и подсистема приобретения знаний, инициализация которых может быть вызвана появлением в базе знаний (ее оперативной части) исходных фактов для рассуждений или описаний новых фрагментов знаний.

Знания системы могут формироваться на основе:

* рекомендаций эксперта;
* аналогий;
* примеров;
* наблюдений, открытий и экспериментов;
* умозаключений на базе глубинных представлений.

Неавтоматизированный сбор знаний экспертов – это трудоемкий процесс, плохо структурированный и носящий циклический характер. Одна его итерация состоит в получении от эксперта максимального количества знаний о каком-либо фрагменте предметной области и анализ их на непротиворечивость.

Для ускорения процесса накопления знаний и придания ему комфортного характера необходима разработка специального программного интерфейса приобретения знаний от эксперта. При создании такого интерфейса важно правильно определиться с языком описания знаний на инфологическом (содержательном) уровне. С одной стороны, этот язык должен быть достаточно формален, нагляден и удобен для эксперта; он должен позволять эксперту использовать свою терминологию на естественном языке. С другой стороны, этот язык не должен создавать проблем компьютеру с его распознаванием и интерпретацией, как, например, естественный язык.

2.1. Дерево решений как формализм описания знаний на инфологическом уровне

Анализ действий экспертов, связанных с диагностикой состояний сложных объектов и систем, показывает, что профессионал проводит экспертизу вполне целенаправленно, придерживаясь определенной стратегии. При этом он руководствуется множеством правил (эвристик), которые могут быть представлены в форме

ЕСЛИ *<посылка>* ТО  *<заключение>,*

называемой *продукционным правилом*.

Правило может быть задано с *коэффициентом уверенности*, определяющим положительным значением меньше или равном единице.

Указанным выше требованиям при описании знаний продукционного типа на инфологическом уровне в достаточной степени удовлетворяет традиционное *дерево решений* [8, 9]. Это бинарная древовидная структура с вершинами двух типов:

* *вершины условий* (*типа «овал»*), из которых могут исходить только две ветви (одна – по «Да», другая – по «Нет»);
* *вершины выводов* (*типа «прямоугольник»*), имеющие единственный выход (обязательно на вершину условия) для промежуточных выводов и не имеющие выхода (не продолжающие логику рассуждений) в случае окончательного вывода.

Каждая вершина условия содержит некоторое высказывание, которое может принимать значения «Истина» или «Ложь» (соответствует выходам по «Да» или «Нет»). Вершина вывода содержит одно или несколько предложений, описывающих некоторое промежуточное или окончательное заключение в виде набора фактов или ссылок на определенные действия. Корень дерева обозначается как вершина условия (овал), содержащая высказывание, с которого начинается процесс логических рассуждений.

Таким образом, дерево решений представляет собой структуру, описывающую логику рассуждений в предметной области. Для описания структурированных (с целью сужения пространства поиска) знаний может использоваться не одно, а несколько деревьев решений.

Данная структура дерева решений обладает следующими недостатками. Во-первых, она не ориентирована на описание нечеткой логики рассуждений эксперта. Во-вторых, заставляет дублировать фрагменты дерева решений в случаях прихода к ним с разных направлений.

2.2. Граф решений

*Граф решений* [1] представляет собой ориентированный помеченный граф с вершинами двух типов:

*вершины условий* (*типа «овал»*), из которых могут исходить несколько ребер (по «Да» или «Нет» со своими коэффициентами уверенности, помечающими эти ребра);

*вершины выводов* (*типа «прямоугольник»*), имеющие один вход и один выход (обязательно на вершину условия) для промежуточных выводов и не имеющие выхода (не продолжающие логику рассуждений) в случае окончательного вывода.

Каждая вершина условия (вершина-овал) содержит некоторое высказывание, которое может принимать значения «Истина» или «Ложь» с различными коэффициентами уверенности (каждому коэффициенту уверенности соответствует помеченный им выход).

Вершина вывода содержит одно или несколько предложений, описывающих некоторое промежуточное или окончательное заключение в виде набора фактов или ссылок на определенные действия. Вершина вывода помечается (например, в нижнем правом углу) *коэффициентом уверенности[4,5,6]*, определяющим положительным значением меньше единицы степень адекватности (правдоподобия) вывода условиям его активации.

Граф имеет одну корневую вершину-овал, содержащую высказывание, с которого начинается описание логики рассуждений. Некоторые из вершин-овалов могут иметь несколько входных ребер с разных направлений, не приводящих к появлению обратных связей (исключением является корневая вершина-овал, не имеющая ни одного входного ребра).

Таким образом, граф решений позволяет описать более глубокую, нечеткую логику рассуждений эксперта в предметной области. Для описания структурированных (с целью сужения пространства поиска) знаний может использоваться не один, а несколько графов решений.

3. Гибридная система интеллектуальной поддержки процессов принятия решений

3.1. Концепция системы, основанной на знаниях

Под *интеллектуальной системой* (ИС, или IS — Intellectual System)принято понимать программно-техническую систему, способную решать задачи, традиционно считающиеся творческими, принадлежащие конкретной предметной области, знания о которой хранятся в памяти интеллектуальной системы.

*Экспертная система* (ЭС, или ES — Expert System) – система, объединяющая возможности компьютера со знаниями и опытом эксперта в такой форме, что система может предложить «разумный совет» или осуществить «разумное решение» поставленной задачи. При этом система способна пояснить «ход своих рассуждений» в понятной для пользователя форме.[2]

По определению Эдварда Фейгенбаума (Станфордский университет), *система, основанная на знаниях* (СОЗ, или на английском knowledge-based system), – это интеллектуальная компьютерная программа, использующая знания и процедуру вывода для решения проблем, которые настолько сложны, что требуют привлечения эксперта.

С точки зрения функционального назначения, понятия СОЗ, ИС и ЭС можно рассматривать как синонимичные. Однако в определении СОЗ явно прослеживается базовый принцип организации системы *декларативного* (не предписывающего) типа. Именно этот принцип, предполагающий четкое отделение друг от друга базы знаний и механизма вывода, максимально обеспечивает модульный принцип построения, открытость системы, возможность создания оболочек экспертных систем (empty expert systems), настраиваемых через формализм базы знаний на различные предметные области. В остальном СОЗ присущи все особенности, характерные для ИС (ЭС):

* ограниченность определенной областью экспертизы;
* качественный характер выходных результатов;
* способность рассуждать при сомнительных, неполных данных;
* способность объяснять ход своих рассуждений понятным пользователю способом;
* способность самообучаться и адаптироваться к конкретным условиям применения и многое другое.

Целесообразность создания и использования СОЗ в конкретной предметной (проблемной) области определяется следующими критериями:

* отсутствие строгих алгоритмов и существование эвристических методов (приемов) решения задач;
* наличие эксперта, способного решать задачи и объяснять ход их решения;
* надежность и статичность имеющихся знаний (статичность понимается в том плане, что есть сложившийся, устоявшийся костяк знаний);
* сомнительный характер доступных данных;
* в основе решения задач лежит метод логических рассуждений.

*Автономная* СОЗ традиционно призвана оказывать консультации и использует в основном эвристические приемы решения задач, не привлекая формальные методы моделирования, анализа и синтеза.

*Неавтономная* СОЗ выступает как *средство интеллектуальной поддержки процессов принятия решений* в различных предметных (проблемных) областях. Она может быть организована в виде интеллектуальной надстройки над прикладными программами, а может и интегрироваться с последними, придавая им интеллектуальность.

Основой для создания как автономных, так и неавтономных экспертных систем может служить *оболочка СОЗ* (empty knowledge-based system), реализующая определенные формы представления знаний, механизмы их приобретения и интерпретации с использованием процедурного анализа и метапроцедур, лежащих в основе интеллектуальной деятельности человека.

3.2. Нейросетевой подход к построению интеллектуальной системы

*Искусственная нейронная сеть* (ИНС) – это существенно параллельно распределенный процессор, обладающий способностью к приобретению, сохранению и репрезентации опытного знания [10] и сходный с мозгом в двух аспектах:

* знание приобретается сетью в процессе обучения;
* для сохранения знания используются силы межнейронных связей (*синаптические веса*).
* Составными элементами ИНС являются спецпроцессоры двух видов:
* *искусственные нейроны* (или просто *нейроны*), суммирующие поступающие на их входы сигналы и преобразующие смещенную на величину порога сумму в соответствии с заданной активационной функцией нейрона;
* *связи между нейронами*, реализующие межнейронные взаимодействия в виде сигналов, умножаемых на синаптические веса связей.

Схематическое представление модели нейрона с входными связями изображено на рис. 1, где использованы следующие обозначения:

**** — сигналы, поступающие на входные связи нейрона;

**** — синаптические веса входных связей;

**** — порог срабатывания нейрона;

****;

**** — активационная функция нейрона, реализующая тот или иной вид преобразования суммы взвешенных входных сигналов, смещенной на величину порога активационной функции ****.

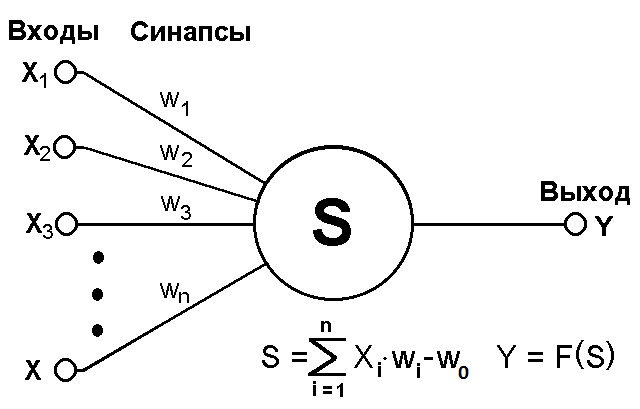


Рисунок 1. Схематическое представление нейрона с входными связями

В нейронной сети между нейронами **** и **** могут быть установлены направленные связи **** и ****, каждой из которых присваиваются, соответственно, веса **** и **** (*синаптические веса*) (рис. 2).

...

...

(i, j)

*wji*

*wij*

(j, i)

Рисунок 2.Межнейронные связи

Таким образом, объединяя нейроны между собой направленными связями, можно строить различные топологии (структуры) нейронных сетей и обучать их решению конкретных задач. *Обучить ИНС* – это значит подобрать такие значения синаптических весов ее межнейронных связей и, возможно, порогов и параметров активационных функций нейронов, при которых сеть приобретает способность выполнять желаемый вид преобразования.

3.3. Концепция гибридной системы интеллектуальной поддержки и её основные компоненты

Традиционные системы, основанные на знаниях продукционного типа, используемые в качестве средств интеллектуальной поддержки процессов принятия решений, обладают рядом достоинств:

* способны осуществлять как четкий, так и нечеткий логический вывод.
* обеспечивают возможность отслеживания логики рассуждений для обоснования принимаемых решений.
* используют сравнительно несложную технологию получения знаний от эксперта в виде графа решений и трансформации их в набор продукционных правил.
* обладают свойством открытости в плане внесения изменений в базу правил.

Вместе с тем эти системы обладают и недостатками, среди которых можно выделить следующие:

* громоздкость процесса вывода, связанная с проверкой условий применимости правил;
* сложность управления процессом вывода, сопряженная с риском обойти вниманием важные решения.

Отмеченные недостатки обусловлены лежащим в основе систем базовым принципом аналитического (символьного) распознавания ситуаций.

Избавиться от указанных недостатков позволяет синтетический (образный) способ восприятия и распознавания ситуаций, свойственный искусственным нейронным сетям.

Образное восприятие ситуации в целом и практически мгновенная реакция обученной нейронной сети значительно ускоряют процесс принятия решений. Однако такому подходу свойственны свои недостатки:

* сложно отследить логику «рассуждений» сети для обоснования принимаемых ею решений;
* сложно сформировать множество шаблонов для обучения (дообучения) нейронной сети.

Все вышесказанное может служить основанием для введения концепции *гибридной* системы интеллектуальной поддержки [1], сочетающей в себе оба рассмотренных подхода, каждый из которых способен компенсировать недостатки другого.

В составе оболочки *гибридной* системы интеллектуальной поддержки процессов принятия решений предусмотрены следующие основные компоненты (рис. 3):

* исходно пустая база знаний;
* механизм вывода;
* подсистема приобретения знаний;
* механизм обучения нейронной сети;
* подсистема объяснения;
* интеллектуальный интерфейс.

*База знаний* включает две традиционные составляющие. Одна из них (*база правил*) объединяет в себе долговременные знания о предметной области в виде набора продукционных правил, получаемых от подсистемы приобретения знаний на этапе настройки оболочки системы на определенную предметную область. Другая составляющая базы знаний (*рабочая память*) представляет ее динамическую часть, в которой хранятся факты (оперативные данные), описывающие текущую ситуацию (состояние процесса принятия решений).

*Механизм вывода* представляет собой аналитическое ядро системы, выполняющее функцию построения прямой логической цепочки рассуждений с использованием стратегий поиска в глубину и в ширину.

*Подсистема приобретения знаний* в качестве главного механизма получения знаний от эксперта использует граф решений. Принимая на вход описание графа решений, она формирует в требуемом формате набор продукционных правил, который передает в базу правил.

Кроме этого подсистема приобретения знаний должна генерировать нейронную сеть, эквивалентную (по логике «рассуждений») исходному графу решений.

Достаточно обученная нейронная сеть при необходимости способна заменить механизм вывода и осуществлять существенно параллельный *нейровывод*. Широтой охвата пространства поиска нейровыводом можно управлять путем селективной активизации входных сигналов сети.

*Подсистема объяснения* призвана представить по требованию пользователя в понятном для него виде цепочку рассуждений, приведшую к тому или иному заключению (заключениям). При работе системы в аналитическом режиме рассуждений обеспечение такой функции не вызывает проблем. Сложнее дело обстоит с нейровыводом, хотя какие-то механизмы раскрытия логики «рассуждений» сети, очевидно, могут быть реализованы (об этом говорит пунктирная связь нейронной сети с подсистемой объяснения, отмеченная на рис. 3).

*Интеллектуальный интерфейс* объединяет в себе лингвистические, информационные и программные средства взаимодействия пользователя, инженера по знаниям и эксперта с соответствующими компонентами системы. Они должны быть ориентированы на неподготовленного пользователя, обладать способностью настраиваться на его терминологию и создавать по возможности комфортные условия для работы в системе.

Рабочая память

База правил

Подсистема приобретения знаний

Интеллектуальный интерфейс

Пользователь

Инженер по знаниям

Эксперт

Подсистема объяснения

Механизм вывода

База знаний

Алгоритм формирования продукционных правил

Алгоритм формирования нейронной сети

Граф решений

Решения нейронной сети

Решения механизма вывода

Механизм обучения нейронной сети

Ошибка

Нейронная сеть

Набор продукционных правил

Рисунок 3. Схема взаимодействия компонент оболочки гибридной системы интеллектуальной поддержки

4. Нейронная сеть как основа механизма вывода гибридной системы интеллектуальной поддержки

4.1. Многослойный персептрон

*Многослойный персептрон* представляет собой сеть прямого распространения с одним *входным* (*сенсорным*)слоем, одним *выходным* (*моторным*) слоем и одним или несколькими *внутренними* (*скрытыми, связующими*) слоями нейронов. Характерной его чертой является *прямонаправленность*: информация, преобразуясь, передается от входного слоя через скрытых слоев к выходному слою.

В стандартной (регулярной) топологии персептрона каждый нейрон   слоя   непосредственно воздействует с синаптическими весами  на все нейроны   следующего слоя  и ни на какие другие (– соответственно, число нейронов в слое  и в слое ;  и  обозначают, соответственно, входной и выходной слои). Схема многослойного персептрона стандартной топологии представлена на рис. 4.

**..................**

**.......................**

**................**

**.......................**

*х1*

*х2*

*хN*

*y1*

*y2*

*yM*

Входной

слой

Скрытые слои

Выходной слой

**.................................**

Рисунок 4. Многослойный персептрон стандартной топологии

Модифицированные версии многослойного персептрона могут иметь нерегулярные связи между слоями (какие-то связи могут отсутствовать) и прямые (непосредственные) связи между нейронами несмежных слоев.

Входной слой персептрона выполняет функции приема и ретрансляции входных сигналов  на нейроны первого скрытого слоя. Основное нелинейное преобразование сигналов происходит в скрытых слоях. Нейроны выходного слоя осуществляют суперпозицию взвешенных сигналов последнего скрытого слоя (имеют линейную активационную функцию) или выполняют нелинейное преобразование, как и нейроны скрытых слоев.

*Обучение персептрона* – это итеративный целенаправленный процесс изменения значений весов синаптических связей (и, возможно, порогов активационных функций нейронов), реализуемый «внешней» процедурой (*алгоритмом обучения*) с использованием тренировочных шаблонов (*обучение с супервизором*) до тех пор, пока сеть не приобретет желаемые свойства. [12]

*Тренировочный шаблон* представляет собой пару векторов , один из которых  – вектор известных входных сигналов, другой  – вектор желаемых выходных сигналов. В процессе обучения векторы  из тренировочного набора  последовательно подаются на вход ИНС и для каждого из них оценивается ошибка между фактическим  и желаемым  откликом (выходом) сети . Затем определяется общая ошибка , на основании которой (а возможно, и с использованием данных о предыдущих итерациях) алгоритм обучения осуществляет модификацию значений настроечных параметров сети, направленную на уменьшение ошибки. Как вариант, модификация значений варьируемых параметров сети может осуществляться после оценки действия каждого очередного шаблона, т.е. по «локальной» ошибке . Процесс обучения повторяется до тех пор, пока сеть не приобретет способность выполнять желаемый тип преобразования, заданный тренировочным набором шаблонов .

В результате обучения сеть достигает способности правильно реагировать не только на шаблоны множества , но и на другие допустимые возмущения, с которыми она никогда ранее не имела дела. В этом состоит уже упоминавшееся *свойство ИНС обобщать*. Ошибка в обобщении, которая может иметь место, определяется двумя причинами:

* ограниченностью размеров сети, ведущей к недостаточному качеству аппроксимации;
* ограниченностью объема обучающей выборки.
  1. Активационные функции нейронов

Работа и обучение нейронной сети зависят не только от её топологии, но и от конфигурации каждого отдельного нейрона, в частности, от активационных функций нейронов и их параметров.

**** — активационная функция нейрона, реализующая тот или иной вид преобразования суммы взвешенных входных сигналов, смещенной на величину порога активационной функции **** [2].

Графические представления некоторых из возможных видов активационных функций приведены на рис. 5.

Рис. 5а представляет *линейную* активационную функцию

.

На рис. 5б изображена *линейная с насыщением* активационная функция

,

имеющая настроечный параметр .

На рис. 5в представлена *ступенчатая* активационная функция (ее часто обозначают )

.

*a*

*s*

*f*(*s*)

- *a*

- *a*

*a*

*б)*  Линейная с насыщением активационная функция

Рис. 12. Некоторые из возможных видов активационных функций

0

*s*

*f*(*s*)

0

1

*s*

*f*(*s*)

*в)*  Ступенчатая активационная функция

*а)*  Линейная активационная функция

0

*f*(*s*)

*s*

1

*г)*  Сигмоидальная активационная функция

0,5

*-σ*

1

*f*(*s*)

*s*

*a*

*д)*  Радиально-симметричная активационная функция

0

0

*K*

*s*

*f*(*s*)

*е)*  Пороговая активационная функция

*σ*

Рис. 12 (продолжение). Некоторые из возможных видов активационных функций

*Сигмоидальная* активационная функция, описываемая выражением ,

изображена на рис. 5*г*. При малых значениях настроечного параметра  функция является более пологой, нежели при больших, приближающих ее к единичной ступеньке.

Рис. 5*д* представляет *радиально-симметричную* активационную функцию

,

с настроечным параметром **.**

Один из самых распространенных видов активационной функции, используемый для многослойных персептронов – это сигмоидальная функция. Её преимущество в том, что сигмоидальная функция дифференцируема на всей оси абсцисс и диапазон. Непрерывность первой производной позволяет обучать сеть градиентными методами (например, методом обратного распространения ошибки).

Сигмоид сужает диапазон изменения так, что значение выхода нейрона лежит между нулем и единицей. Его дополнительное преимущество состоит в контроле усиления. Для слабых сигналов выходная кривая имеет наклон, дающий большее усиление. Когда величина сигнала становится больше, усиление падает. Так большие сигналы воспринимаются сетью без насыщения, а слабые сигналы проходят по сети без сильного ослабления. [15]

1. Нейросетевой модуль гибридной системы интеллектуальной поддержки

5.1. Основные компоненты нейросетевого модуля

Нейросетевой модуль гибридной системы должен содержать [2]:

* Интеллектуальный интерфейс для эксперта или иной компонент для получения знаний в том виде, который пригоден для построения нейронной сети
* Интерфейс для взаимодействия с пользователем, получения от него данных о состоянии системы и вывода решения
* Компонент построения нейронной сети по входному графу решений в заданном формате
* Нейронная сеть, построенная по графу решений, сохраняемая на носителе
* Компонент дообучения нейронной сети, который способен использовать результаты, генерируемые аналитическим модулем системы

Высокоуровневая архитектура основных компонентов и их связь и взаимодейтсвие с компонентами аналитического модуля отражена на рис. 6.

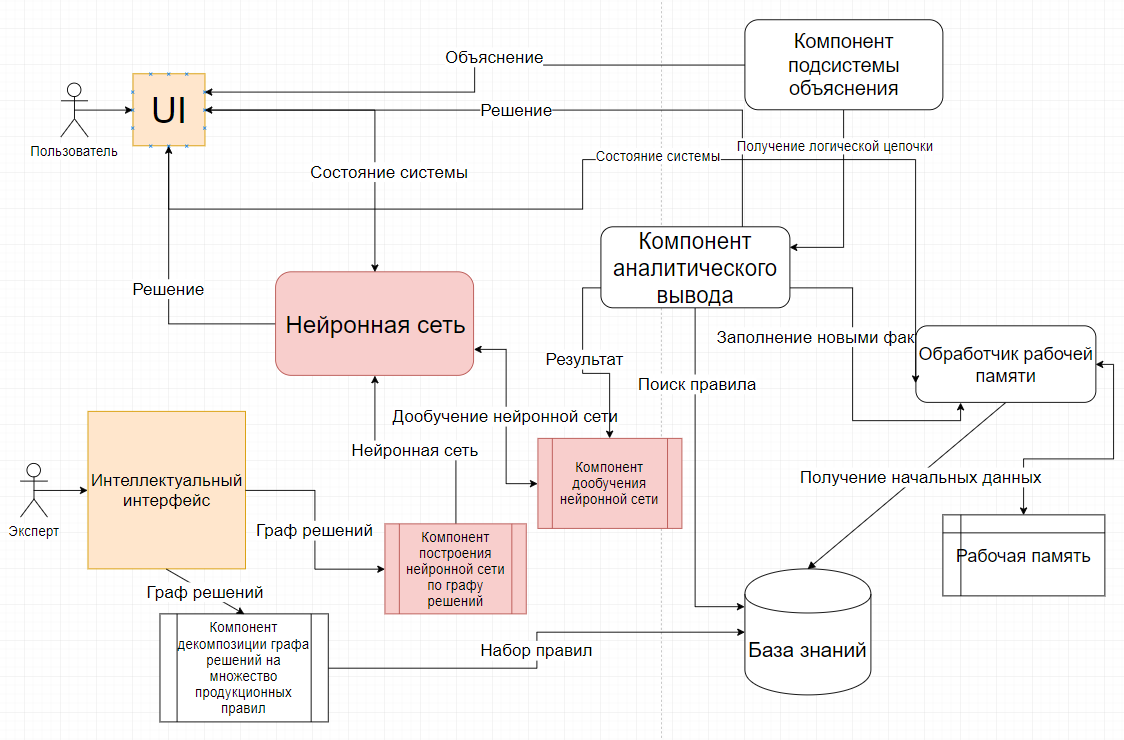


Рисунок 6. Высокоуровневая архитектура компонентов гибридной системы интеллектуальной поддержки. Выделенные компоненты – основные в нейросетевом модуле

* 1. Структура и топология нейронной сети

Нейронная сеть, которая строится по графу решений, должна быть эквивалентна по логике «рассуждений» исходному графу решений [2]. Значит необходимо сопоставить какие-то элементы топологии графа, или его структурные элементы, некоторым элементам нейронной сети.

Нейронная сеть, которая строится по графу решений, представляет собой многослойный персептрон нерегулярной структуры. Это значит, что каждый нейрон каждого слоя может быть связан только с последующими слоями, но не каждый нейрон очередного слоя связан с каждым нейроном следующего слоя. Данная топология нейронной сети выбрана как наиболее отражающая логику «рассуждений» исходного графа.

* + 1. Сенсорный слой

Абсолютно логично поставить в соответствие вершинам-условиям (овалам) графа решений нейроны сенсорного слоя нейронной сети, т.к. вершины-условия используются для получения информации о текущем состоянии системы.

Каждая исходящая связь из вершины-условия должна породить собственный нейрон для реализации нечеткой логики, чтобы у пользователя была возможность получить рассуждения сразу по нескольким фактам, в случае сомнений.

Логично использовать коэффициенты уверенности каждой связи, исходящей из вершины-условия как синаптический вес соответствующей исходящей связи из нейрона. Однако просто использовать коэффициенты уверенности как синаптические веса нельзя, т.к. в таком случае рассуждения, связанные с коэффициентами уверенности близкими к нулю проводиться не будут, а это делает итоговое решение неполным и исключает из него часть рассуждений (например факт «Нет искры» – это «Есть искра» с коэффициентом уверенности 0)

В связи с этим предлагается задавать синаптические веса исходящих связей нейронов сенсорного слоя равными 1. А для решения проблемы, связанной с просчетом нужных направлений рассуждений, использовать специальную активационную функцию у нейронов сенсорного слоя:

*,* где *–* порог активационной функции, равный коэффициенту уверенности эксперта на соответствующей исходящей связи вершины-условия*, s –* коэффициент уверенности факта, который задает пользователь.

* + 1. Скрытые слои

Скрытые слои нейронной сети – промежуточное звено, предназначенное для «рассуждений». Логично поставить в соответствие нейронам скрытых слоёв сети вершины промежуточных выводов, а также вершины окончательных выводов, сделав нейроны моторного слоя ретранслирующими.

Связи между нейронами распределяются соотвественно связям в графе.

Активационная функция нейронов скрытых слоёв – сигмоидальная функция c фиксированным порогом. Такой фиксированный порог необходим для того, чтобы функция выдавала корректные результаты в диапазоне [0,1]. С помощью настроечного параметра a можно управлять крутизной сигмоиды, настраивая работу сети.

*,* где *a* – настроечный параметр функции.

* + 1. Моторный слой

Выходной слой сети является ретранслирующим, то есть активационная функция каждого нейрона – линейная с параметром 1. Каждый нейрон моторного слоя должен соответствовать каждой вершине окончательного вывода в исходном графе решений. Значения, полученные в результате работы сети – это и есть решение, проанализировав его, пользователь может сделать выбор.

* 1. Алгоритм построения нейронной сети по графу решений

Предлагается следующий алгоритм построения нейронной сети по графу решений:

**ШАГ 1** Пройти по всем вершинам графа, если это эллипс - найти все связи, выходящие из него, добавить нейрон с порогом активационной функции, равным коэффициенту уверенности связи, на сенсорный слой. Записать id узла равным id вершины в графе. **ПЕРЕЙТИ НА ШАГ2**

**ШАГ 2** Для каждого нейрона сенсорного слоя выполнить:

**ШАГ 2.0** Найти вершину графа с id равным id текущего нейрона. Найти все вершины, связанные с этой вершиной. Для каждой из них **ВЫПОЛНИТЬ РЕКУРСИВНУЮ ФУНКЦИЮ TRAVERSE** с параметрами index = 0, weigth = 1, sender = текущий нейрон сенсорного слоя, current = текущая вершина.

**Функция traverse**:

Функция принимает на вход *index* – номер текущего скрытого слоя нейронной сети, *sender* – нейрон, для которого новая связь будет исходящей. *weight* - вес связи между *sender* и очередным нейроном, *current – вершина, для которой выполняется обход*

**ШАГ 2.1** Найти вершину графа с id равным id нейрона sender. Если тип этой вершины – эллипс, то **ПЕРЕЙТИ НА ШАГ 2.2**, иначе проверить количество скрытых слоев сети, если оно меньше index + 1, то добавить новый скрытый слой. **ПЕРЕЙТИ НА ШАГ 2.1.0**

**ШАГ 2.1.0** Определить количество связей найденной вершины. Если количество связей равно нулю, то **ПЕРЕЙТИ НА ШАГ 2.1.1**, иначе **ПЕРЕЙТИ НА ШАГ 2.1.2**

**ШАГ 2.1.1** Проверить, нет ли на скрытых слоях нейрона с id равным (id вершины + 'motor') если есть, то **ПЕРЕЙТИ НА ШАГ 2.1.1.1**, иначе **ПЕРЕЙТИ НА ШАГ 2.1.1.2**

**ШАГ 2.1.1.1** Создать нейрон на моторном слое, создать нейрон на скрытом слое с номером index. Создать связь между нейроном sender и нейроном скрытого слоя, присвоить связи вес weight. Создать связь между нейроном скрытого слоя и нейроном моторного слоя. Присвоить связи вес, который соответствует коэффициенту найденной на предыдущих шагах вершины.

**ШАГ 2.1.1.2** Найти нейрон скрытого слоя с id равным id вершины. Создать связь между нейроном, который вызвал функцию traverse и нейроном скрытого слоя, присвоить связи вес weight.

**ШАГ 2.1.2** Проверить, нет ли на скрытых слоях нейрона с id равным id вершины. Если нет, то **ПЕРЕЙТИ НА ШАГ 2.1.2.1**, иначе **ПЕРЕЙТИ НА ШАГ 2.1.2.2**

**ШАГ 2.1.2.1** Создать нейрон на скрытом слое с индексом index, создать связь между нейроном sender и нейроном скрытого слоя, присвоить связи вес weight. Для всех вершин, связанных с текущим, вызвать функцию traverse, передав в неё параметр index + 1 и weight = коэффициент узла-прямоугольника, sender = нейрон скрытого слоя.

**ШАГ 2.1.2.2** создать связь между нейроном sender и нейрономом скрытого слоя, присвоить связи вес weight

**ШАГ 2.2** Для всех вершин, связанных с найденной вершиной, вызвать функцию traverse, передав в неё параметр index, weight, sender равные текущим.

* 1. Пример построения нейронной сети по графу решений из области диагностики неполадок сетевого соединения

Для примера используется фрагемент графа решений из области диагностики неполадок сетевого соединения. Фрагмент графа изображен на Рис. 7, а полученная нейронная сеть на Рис. 8

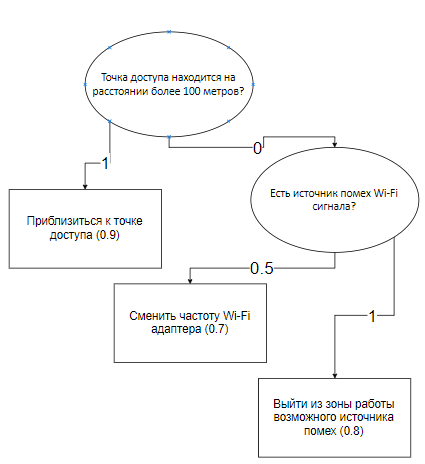


Рисунок 7. Пример исходного фрагмента графа решений

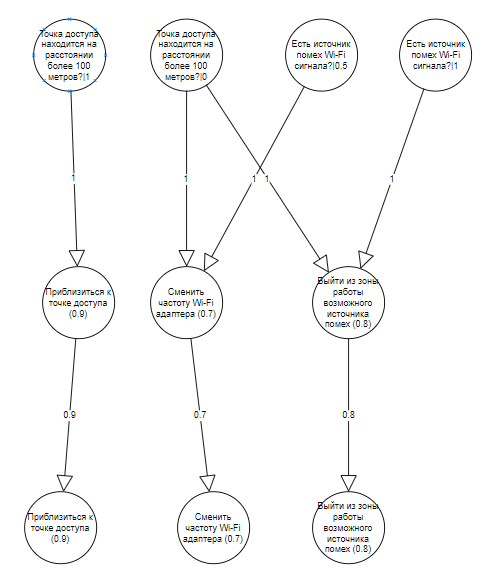


Рисунок 8. Полученная в результате работы алгоритма нейронная сеть

1. Программная реализация

6.1. Общее описание программной реализации

Программная реализация нейросетевого модуля гибридной интеллектуальной системы написана на языке Python 3.5.6 [14], использована среда разработки PyCharm Community Edition, использованы только стандартные модули-библиотеки языка, такие как “ElementTree”, “json”.

Разработка велась с использованием системы контроля версий Git, использовался репозиторий GitHub, средство для управления репозиторием – SourceTree.

Исходные коды основных модулей системы доступны в приложении и в репозитории, ссылка так же в приложении.

* 1. Архитектура программной системы

Основные архитектурные решения и компоненты системы описаны с помощью диаграмм классов языка UML [11]

* + 1. Диаграмма классов компонента нейронной сети

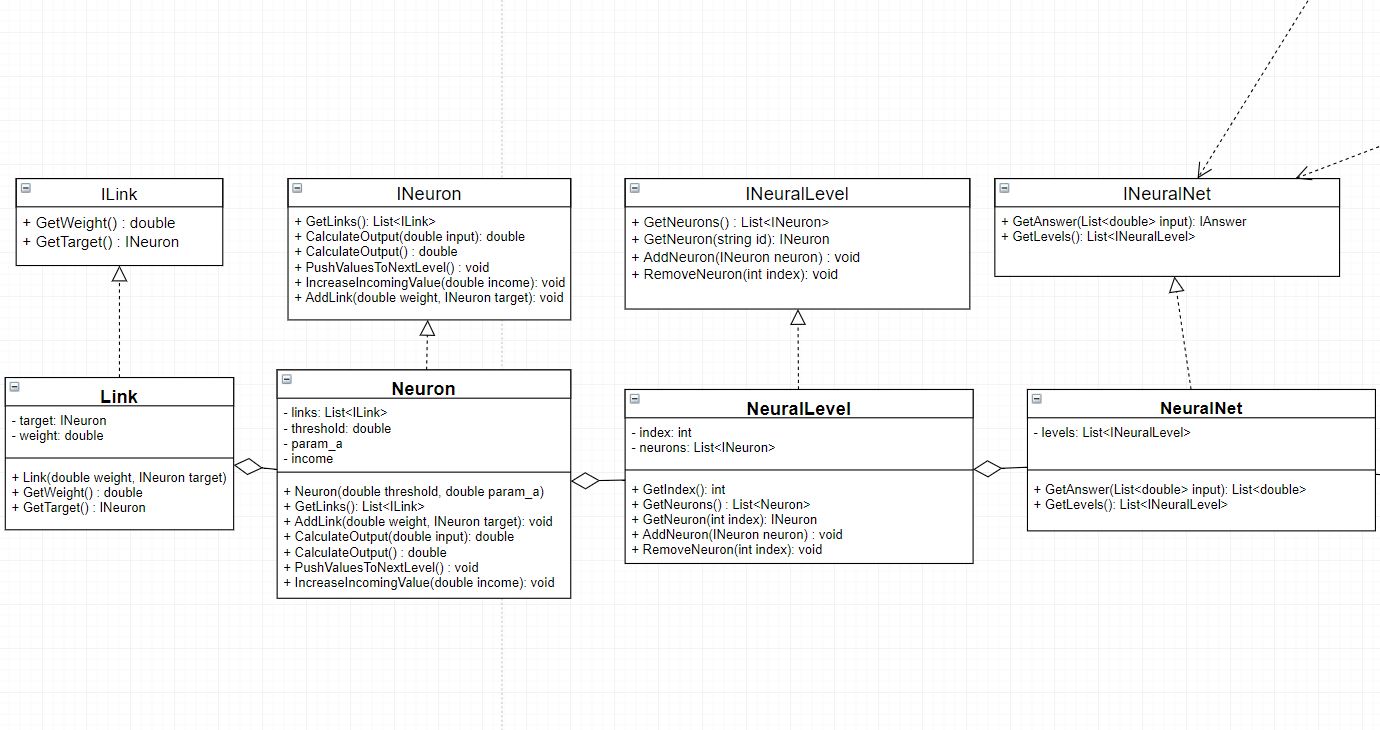


Рис. 9. Диаграмма классов компонента нейронной сети

*Основные компоненты*: NeuralNet – класс нейронной сети, агрегирует класс NeuralLevel – класс слоя нейронной сети - агрегирует Neuron – класс нейрона – агрегирует Link – класс связи нейронной сети

* + 1. Диаграмма классов компонента построения нейронной сети

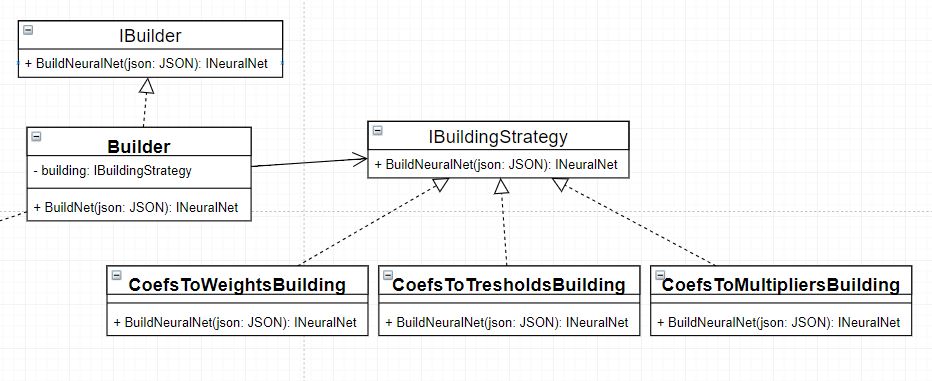


Рис. 10. Диаграмма классов компонента построения нейронной сети

Основные компоненты: Builder – модуль обучения нейронной сети, с помощью паттерна стратегия позволяет настраивать стратегию построения нейронной сети(IBuildingStrategy) – CoefsToThreshholdsBuilding– класс построения с помощью перевода коэффициентов уверенности в пороги активационных функций, CoefsToValuesBuilding – класс построения нейронной сети с помощью перевода коэффициентов уверенности в синаптические веса нейронов, CoefsToWeightsBuilding – класс построения нейронной сети с помощью перевода коэффициентов уверенности в пороги активационных функций.

* + 1. Диаграмма классов компонента дообучения нейронной сети

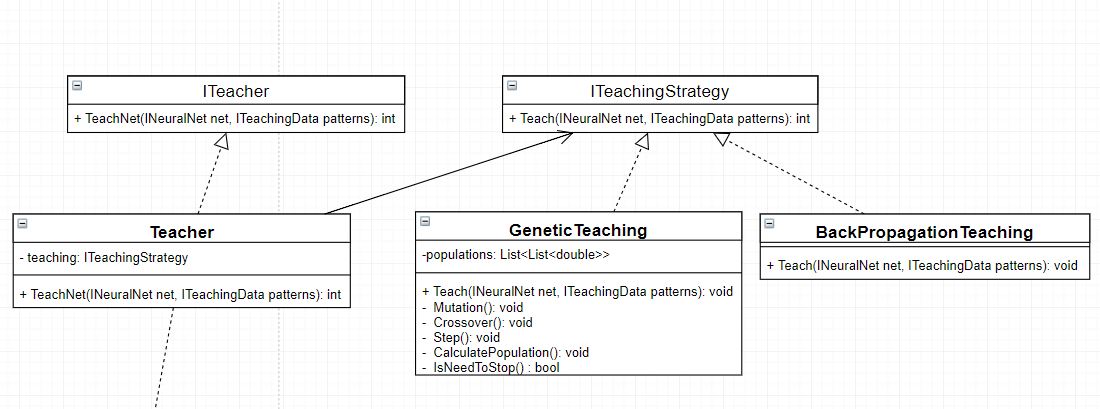


Рис. 11. Диаграмма классов компонента построения нейронной сети

Основные компоненты: Teacher – модуль обучения нейронной сети, с помощью паттерна стратегия позволяет настраивать стратегию обучения(ITeachingStrategy) – GeneticTeaching – класс обучения с помощью генетического алгоритма, или BackPropagationTeaching – класс обучения с помощью алгоритма обратного распространения ошибки

* 1. Интеллектуальный интерфейс

Интеллектуальный интерфейс реализован в приложении в виде парсера графа в формате XML, который преобразует это представление графа в json-представление, которое принимается компонентом построения нейронной сети. Для генерации XML с визуальным представлением графа решений, а также для рисования графа решений используется бесплатный сервис draw.io

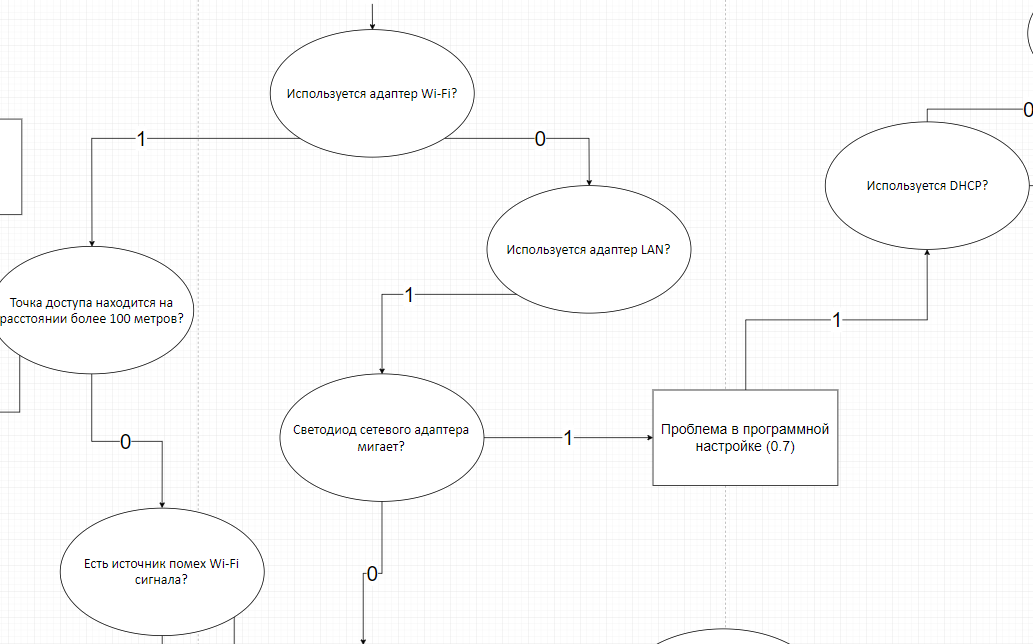


Рис. 12. Рисование графа решений, который преобразуется в json-представление

* 1. Компонент построения нейронной сети по графу решений

Данный компонент принимает на вход json заданного формата: корневой узел должен имееть два аттрибута: **nodes** – массив описаний вершин, каждый элемент которого должен иметь аттрибуты: type – тип вершины, text – текст вершины, id – уникальный в пределелах файла идентификатор вершины, coefficient – коэффициент вершины, если таковой имеется, и **links** – массив описаний связей, каждый элемент которого должен иметь аттрибуты weight – коэффициент уверенности связи, source – id вершины, для которой связь является исходящей и target – id вершины, для которой связь является входящей.

* 1. Компонент нейросетевого вывода

Основная функция API компонента GetAnswer принимает на вход массив вещественных чисел – коэффициенты уверенности соответствующих условий, и возвращает массив вещественных чисел – коэффициенты уверенности соответствующих окончательных выводов.

1. Полученные результаты

7.1. Тестовый базис

7.1.1. Фрагмент графа решений для области диагностики неполадок сетевого подключения

Граф решений, представляющий основу для тестового базиса модуля, иллюстрирует механизм принятия решений по устранению неполадок сетевого подключения ПК [13]

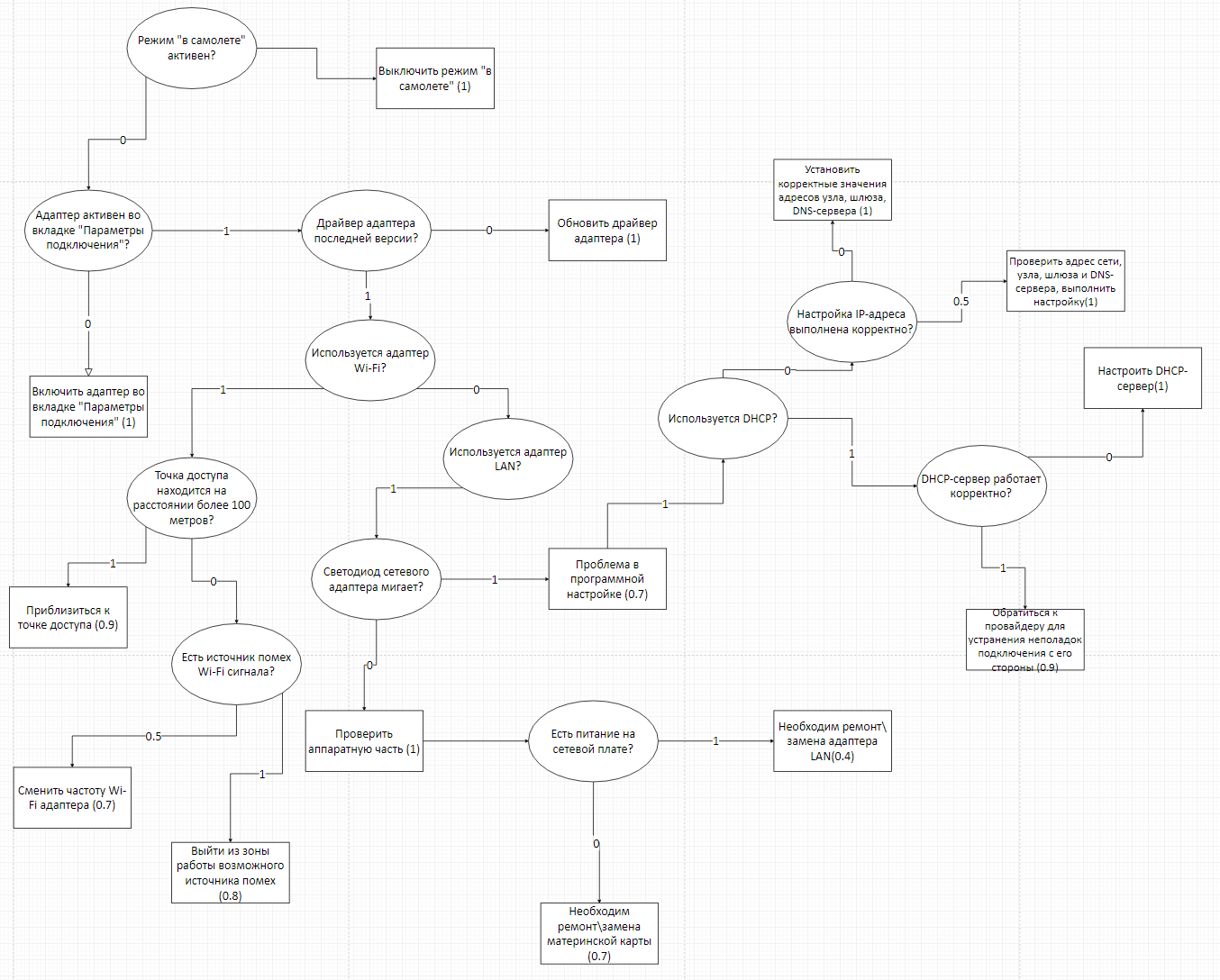


Рис. 13. Фрагмент графа решений для области диагностики неполадок сетевого подключения

7.1.2. Результаты работы нейросетевого модуля

Для получения оценки результатов использовался запуск нейронной сети, построенной по графу решений, представленному на рисунке 13. Для тестирования использовались а) Простые тесты – все входные данные нулевые, либо коэффициент уверенности одного из фактов равен единице. б) Тесты средней сложности, которые возможно проверить вручную в)Автоматически сгенерированные тесты, необходимые для сравнения времени работы и корректность ответов по сравнению с аналитическим модулем. На большинстве тестов нейронная сеть выдавала корректный(«логичный») результат.

1. Сравнение результатов работы нейросетевого и аналитического вывода

Таблица результатов работы нейросетевого модуля и аналитического модуля, доступна в Приложении 1. Тесты генерировались автоматически, значения входного вектора случаные. Корректность ответа оценивалась по максимальному коэффициенту уверенности в выходном векторе – окончательный вывод, который является «руководством к действию». На 70 из 78 тестов нейронная сеть выдает ответ, аналогичный ответу аналитического модуля, из которых 59 совпадают по значению максимального коэффициента уверенности более чем на 80 процентов. Отклонение в результатах можно устранить с помощью дообучения сети, а также варьированием вариантов построения нейронной сети по графу решений.

Время выполнения всех тестов на нейросетевом модуле: 0.018987655639648438 секунд, на аналитическом модуле: 9.46300220489502 секунд.

Заключение

В рамках данной работы был расмотрен и изучен теоретический подход к построению систем интеллектуальной поддержки, нейромодельный подход к созданию интеллектуальных систем, концепция гибридной интеллектуальной системы и, в результате, предложен подход к построению нейросетевого модуля гибридной системы интеллектуальной поддержки и написана его реализация.

В результате, создан программный модуль, который позволяет решать задачи в различных предментных областях, используя для решения нейронную сеть, которая строится по графу решений предметной области. Это дает возможность настраивать модуль на работу с различными предметными областями, меняя только граф решений и легко перестраивая нейронную сеть.

Программный модуль протестирован, соответсвует разработанной архитектуре и api.

Рассмотренный фрагмент графа решений предментой области диагностики неполадок сетевого подключения – лишь один из примеров, наглядно показывающий результаты работы данного модуля. Между тем, меняя предментую область, появляется возможность строить новые нейронные сети, или использовать связку нейронных сетей, которые будут адаптироваться под углубление знаний о предментой области и использовать результаты работы друг друга, обучаясь на результатах работа аналитического модуля.

В работе также предложены альтернативные варианты построения нейронной сети и механизмы дообучения нейронной сети, которые позволят добиться ещё большей точности результатов работы нейросетевого модуля.

Основные пути дальнейшего развития работы над нейросетевым модулем могут включать реализацию новых механизмов дообучения сети, сравнение работы различных механизмов построения нейронной сети по графу решений для устранения отклонения резульатов от результатов работы аналитического модуля.

Список литературы

1. Басалин П.Д., Безрук К.В. Средства интеллектуальной поддержки процессов проектирования и управления // Интеллектуальные системы: Труды Девятого Международного симпозиума / Под ред. К.А. Пупкова. — М.: РУСАКИ, 2010. — С. 217—221.
2. Басалин П.Д., Безрук К.В., Радаева М.В.Модели и методы интеллектуальной поддержки процессов принятия решений: Учебное пособие. – Нижний Новгород: Нижегородский госуниверситет, 2011. – 129 с.
3. Представление знаний в человеко-машинных и робототехнических системах: Том А. Фундаментальные исследования в области представления знаний. — М.: ВЦ АН СССР, ВИНИТИ, 1984. — 261 с.
4. Борисов В.В., Круглов В.В., Федулов А.С. Нечеткие модели и сети. — М.: Горячая линия — Телеком, 2007. — 284 с.
5. Малышев Н.Г., Берштейн Л.С., Боженюк А.В. Нечеткие модели для экспертных систем в САПР. — М.: Энергоатомиздат, 1991. — 136 с.
6. Частиков А.П., Гаврилова Т.А., Белов Д.Л. Разработка экспертных систем. Среда CLIPS. — СПб.: БХВ-Петербург, 2003. — 608 с.
7. Вагин В.Н., Головина Е.Ю., Загорянская А.А., Фомина М.В. Достоверный и правдоподобный вывод в интеллектуальных системах / Под ред. В.Н. Вагина, Д.А. Поспелова. — М.: Физматлит, 2004. — 704 с.
8. Леви Р., Дранг Д., Эдельсон Б. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на Бейсике. — М.: Финансы и статистика, 1990. — 239 с.
9. Таунсенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ: Пер. с англ. / Предисл. Г.С. Осипова. — М.: Финансы и статистика, 1990. — 320 с.
10. Aleksander I., Morton H. An Introduction to Neural Computing. — London: Chapman&Hall, 1990.
11. Фаулер М. UML. Основы. Третье издание. Краткое руководство по стандартному языку объектного моделирования: Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с.
12. Хайкин С. Нейронные сети: полный курс. — 2-е издание: Пер. с англ. — М.: Издательский дом «Вильямс», 2008. — 1104 с.
13. Поляк-Брагинский А. В. Локальные сети. Модернизация и поиск неисправностей. — СПб.: БХВ-Петербург, 2006. - 640 с : ил.
14. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
15. Mitchell T. Machine Learning. — WCB–McGraw–Hill, 1997.

Приложение

Приложение 1. Сравнение результатов работы нейросетевого и аналитического модуля

|  |  |  |  |
| --- | --- | --- | --- |
| **Нейросетевой** | **Аналитический** | **MATCH\_MAX** | **% совпадения коэффициента** |
| ('65', 0.6631326077637634) | ('65', 0.2015265644673308) | TRUE | 0.5383939567035674 |
| ('63', 0.6394369467352581) | ('63', 0.2652755828743179) | TRUE | 0.6258386361390598 |
| ('63', 0.7070084269658296) | ('63', 0.41113184163226457) | TRUE | 0.704123414666435 |
| ('63', 0.760797907112824) | ('63', 0.4867961117010605) | TRUE | 0.7259982045882365 |
| ('63', 0.7722914813847853) | ('63', 0.5090182920878775) | TRUE | 0.7367268107030922 |
| ('49', 0.6786555744183377) | ('49', 0.44223324522031243) | TRUE | 0.7635776708019748 |
| ('49', 0.6191589609629659) | ('49', 0.3847446311407449) | TRUE | 0.765585670177779 |
| ('49', 0.6755685901035252) | ('49', 0.4650221043766841) | TRUE | 0.7894535142731589 |
| ('49', 0.7463250904841635) | ('49', 0.5475743137886772) | TRUE | 0.8012492233045136 |
| ('49', 0.7098614800285399) | ('49', 0.5124948509858795) | TRUE | 0.8026333709573397 |
| ('49', 0.7217904590896335) | ('49', 0.527531280418378) | TRUE | 0.8057408213287446 |
| ('49', 0.7599355968275431) | ('49', 0.5891390357471216) | TRUE | 0.8292034389195785 |
| ('63', 0.8204566672436712) | ('63', 0.6604081519479827) | TRUE | 0.8399514847043116 |
| ('49', 0.7477927265736966) | ('49', 0.5880712070515332) | TRUE | 0.8402784804778366 |
| ('49', 0.7617544948918193) | ('49', 0.6226804715413663) | TRUE | 0.860925976649547 |
| ('49', 0.7664246075602065) | ('49', 0.6274883423458653) | TRUE | 0.8610637347856588 |
| ('49', 0.7670637106306601) | ('49', 0.6316494807189302) | TRUE | 0.8645857700882701 |
| ('64', 0.8780990177469524) | ('64', 0.9936375673598603) | TRUE | 0.8844614503870921 |
| ('64', 0.87114585837802) | ('64', 0.9777820860256936) | TRUE | 0.8933637723523264 |
| ('64', 0.8644411746023596) | ('64', 0.9631693938373409) | TRUE | 0.9012717807650187 |
| ('64', 0.8569182392033544) | ('64', 0.9474815724640613) | TRUE | 0.9094366667392931 |
| ('64', 0.8559961572438927) | ('64', 0.9456064754760911) | TRUE | 0.9103896817678016 |
| ('64', 0.8545255480759205) | ('64', 0.9426364793350338) | TRUE | 0.9118890687408867 |
| ('64', 0.8494124164446896) | ('64', 0.9324999925544567) | TRUE | 0.9169124238902329 |
| ('64', 0.8452407844889626) | ('64', 0.9244377695181298) | TRUE | 0.9208030149708327 |
| ('64', 0.843231491715045) | ('64', 0.9206178193058131) | TRUE | 0.9226136724092319 |
| ('49', 0.8137674216536184) | ('49', 0.7376913232804397) | TRUE | 0.9239239016268214 |
| ('49', 0.8047285952950486) | ('49', 0.7291957735004119) | TRUE | 0.9244671782053633 |
| ('64', 0.8391369374872858) | ('64', 0.9129551121507696) | TRUE | 0.9261818253365162 |
| ('64', 0.8382425422483956) | ('64', 0.9113023623804778) | TRUE | 0.9269401798679178 |
| ('49', 0.8126632163534269) | ('49', 0.7495916725712141) | TRUE | 0.9369284562177872 |
| ('49', 0.8163530515194126) | ('49', 0.7603529423957647) | TRUE | 0.943999890876352 |
| ('64', 0.8145096190528106) | ('64', 0.8698958029413879) | TRUE | 0.9446138161114227 |
| ('64', 0.8002614936771709) | ('64', 0.8469823746688365) | TRUE | 0.9532791190083344 |
| ('64', 0.797175429780542) | ('64', 0.842183336977608) | TRUE | 0.954992092802934 |
| ('64', 0.7954410423572823) | ('64', 0.8395101239308914) | TRUE | 0.9559309184263909 |
| ('64', 0.7951447809007697) | ('64', 0.8390551827707403) | TRUE | 0.9560895981300294 |
| ('64', 0.7926821885995569) | ('64', 0.8352923584168567) | TRUE | 0.9573898301827002 |
| ('64', 0.7916885891909111) | ('64', 0.8337834979029367) | TRUE | 0.9579050912879744 |
| ('64', 0.791081751225769) | ('64', 0.832864573228184) | TRUE | 0.958217177997585 |
| ('64', 0.7895165785299364) | ('64', 0.8305034850218944) | TRUE | 0.959013093508042 |
| ('64', 0.7844487707668624) | ('64', 0.8229456660814831) | TRUE | 0.9615031046853793 |
| ('64', 0.7842629464419721) | ('64', 0.8226710083774484) | TRUE | 0.9615919380645237 |
| ('49', 0.8295619223020062) | ('49', 0.7948263101697891) | TRUE | 0.9652643878677829 |
| ('64', 0.7639459934130975) | ('64', 0.7936091192756812) | TRUE | 0.9703368741374163 |
| ('64', 0.7603917447710311) | ('64', 0.7887071085667638) | TRUE | 0.9716846362042673 |
| ('64', 0.7541706806257962) | ('64', 0.780245309456333) | TRUE | 0.9739253711694632 |
| ('64', 0.7512876335189608) | ('64', 0.7763728762661171) | TRUE | 0.9749147572528437 |
| ('64', 0.7426139141510156) | ('64', 0.7648997603677729) | TRUE | 0.9777141537832428 |
| ('64', 0.7419300918541193) | ('64', 0.7640061282187315) | TRUE | 0.9779239636353878 |
| ('64', 0.7413484630787247) | ('64', 0.7632472592006558) | TRUE | 0.9781012038780689 |
| ('64', 0.7412473419350465) | ('64', 0.7631154367853822) | TRUE | 0.9781319051496643 |
| ('64', 0.7355332296206462) | ('64', 0.7557200267711955) | TRUE | 0.9798132028494506 |
| ('64', 0.7351193135898836) | ('64', 0.7551883332043436) | TRUE | 0.9799309803855399 |
| ('64', 0.7305332956059595) | ('64', 0.7493324931209068) | TRUE | 0.9812008024850527 |
| ('64', 0.7268176798461985) | ('64', 0.7446340625389077) | TRUE | 0.9821836173072908 |
| ('64', 0.7099560399384407) | ('64', 0.7237926386568055) | TRUE | 0.9861634012816352 |
| ('64', 0.6985913543216222) | ('64', 0.7101497464325801) | TRUE | 0.9884416078890421 |
| ('64', 0.6985473026149059) | ('64', 0.710097445938236) | TRUE | 0.98844985667667 |
| ('64', 0.697960361576618) | ('64', 0.7094010126895408) | TRUE | 0.9885593488870772 |
| ('64', 0.6929054744534145) | ('64', 0.7034344970969258) | TRUE | 0.9894709773564887 |
| ('64', 0.6783824043756363) | ('64', 0.6865869746561467) | TRUE | 0.9917954297194896 |
| ('64', 0.6773148476839131) | ('64', 0.6853647851623343) | TRUE | 0.9919500625215788 |
| ('64', 0.666093693090818) | ('64', 0.6726424763935667) | TRUE | 0.9934512166972513 |
| ('64', 0.6632400861819169) | ('64', 0.6694416977548622) | TRUE | 0.9937983884270547 |
| ('64', 0.6533315935929443) | ('64', 0.6584290196870896) | TRUE | 0.9949025739058548 |
| ('64', 0.6514174310625551) | ('64', 0.6563188807700411) | TRUE | 0.995098550292514 |
| ('64', 0.6438366523954995) | ('64', 0.6480138953450583) | TRUE | 0.9958227570504412 |
| ('64', 0.6286846633215999) | ('64', 0.6316445337837215) | TRUE | 0.9970401295378784 |
| ('64', 0.6123954233825954) | ('64', 0.6143481257454302) | TRUE | 0.9980472976371653 |
| ('49', 0.6234331112396371) | ('64', 0.5870619602943684) | FALSE |  |
| ('63', 0.5640181684644915) | ('64', 0.5459909943684675) | FALSE |  |
| ('71', 0.7276496159990372) | ('63', 0.32386159005372783) | FALSE |  |
| ('77', 0.6264113845726089) | ('64', 0.5135054543867182) | FALSE |  |
| ('76', 0.6847081350071448) | ('64', 0.5299836290416181) | FALSE |  |
| ('63', 0.6859198189080316) | (-1, -1) | FALSE |  |
| ('76', 0.6691235564062361) | ('65', 0.20064245263066646) | FALSE |  |
| ('49', 0.5836053525242035) | ('64', 0.5431405697508235) | FALSE |  |

Таблица 1. Сравнение результатов работы нейросетевого и аналитического вывода

Приложение 2. Исходные коды программного обеспечения

Исходные коды программного обеспечения, написанного в рамках данной работы, доступны на <https://github.com/sevoster/hybrid-decision-making-system>

Ниже приведен фрагмент кода(класс, представляющий один из классов шаблона стратегия), отвечающий за построение нейронной сети по графу решений.

**class** CoefsToWeightsBuilding(IBuildingStrategy):  
 **def** \_\_init\_\_(self):  
 self.nodes = []  
 self.arcs = []  
  
 @staticmethod  
 **def** build\_node\_dictionary(nodes):  
 node\_dict = {}  
 **for** node **in** nodes:  
 node\_dict[node[**'id'**]] = {  
 **'type'**: node[**'type'**],  
 **'text'**: node[**'text'**],  
 **'id'**: node[**'id'**]  
 }  
 **if 'coefficient' in** node:  
 node\_dict[node[**'id'**]][**'coefficient'**] = node[**'coefficient'**]  
 **return** node\_dict  
  
 **def** build\_net(self, building\_data):  
 self.neural\_net = NeuralNet()  
  
 self.nodes = self.build\_node\_dictionary(building\_data[**'nodes'**])  
 self.arcs = building\_data[**'links'**]  
  
 *# 1. добавляем все эллипсы на сенсорный слой* **for** id **in** self.nodes:  
 node = self.nodes[id]  
 **if** node[**'type'**] == **'a'**:  
 **for** arc **in** self.arcs:  
 **if** arc[**'source'**] == id:  
  
  
 self.neural\_net.sensor\_level.add\_neuron(  
 Neuron(id=id, func=**lambda** s, w = float(arc[**'weight'**]): 1 - fabs(float(s) - w), rule=id,  
 sensor\_weigth=arc[**'weight'**], text=node[**'text'**] + **'|'** + arc[**'weight'**]))  
  
 **for** neuron **in** self.neural\_net.sensor\_level.neurons:  
 **for** arc **in** self.arcs:  
 **if** arc[**'source'**] == neuron.id **and** arc[**'weight'**] == neuron.sensor\_weight:  
 self.traverse(sender=neuron, node=self.nodes[arc[**'target'**]],  
 index=0, weight=1)  
 **return** self.neural\_net  
  
 **def** traverse(self, node, sender, index, weight):  
 *# self.neural\_net.print\_net()* **if** node[**'type'**] == **'c'**:  
 **if** len(self.neural\_net.levels) < index + 1:  
 self.neural\_net.levels.append(NeuralLevel())  
  
 **if** len(self.get\_targets\_ids\_by\_id(node[**'id'**])) == 0:  
 neuron = self.get\_neuron\_by\_id(node[**'id'**])  
  
 **if** neuron:  
 sender.add\_link(weight=weight, target=neuron)  
 **else**:  
 m\_neuron = self.neural\_net.motor\_layer.add\_neuron(  
 Neuron(id=node[**'id'**] + **'m'**, text=node[**'text'**], func=**lambda** x: x))  
 h\_neuron = self.neural\_net.levels[index].add\_neuron(Neuron(id=node[**'id'**], text=node[**'text'**]))  
 sender.add\_link(weight=weight, target=h\_neuron)  
 h\_neuron.add\_link(weight=node[**'coefficient'**], target=m\_neuron)  
 **else**:  
 fneuron = self.get\_neuron\_by\_id(node[**'id'**])  
 **if** fneuron:  
 sender.add\_link(weight=weight, target=fneuron)  
 **else**:  
 h\_neuron = self.neural\_net.levels[index].add\_neuron(Neuron(id=node[**'id'**], text=node[**'text'**]))  
 sender.add\_link(weight=weight, target=h\_neuron)  
 **for** arc **in** self.arcs:  
 **if** arc[**'source'**] == h\_neuron.id:  
 self.traverse(self.nodes[arc[**'target'**]], h\_neuron, index + 1, node[**'coefficient'**])  
 **else**:  
 **for** arc **in** self.arcs:  
 **if** arc[**'source'**] == node[**'id'**]:  
 self.traverse(self.nodes[arc[**'target'**]], sender, index, weight)  
  
 **def** get\_targets\_ids\_by\_id(self, id):  
 target\_ids = list()  
 **for** arc **in** self.arcs:  
 **if** arc[**'source'**] == str(id):  
 target\_ids.append(str(id))  
 **return** target\_ids  
  
 **def** get\_neuron\_by\_id(self, id):  
 **for** level **in** self.neural\_net.levels:  
 **for** neuron **in** level.neurons:  
 **if** str(neuron.id) == str(id):  
 **return** neuron