Н. Ю. Прокопенко

СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

на базе Deductor Studio Academic 5.3

Учебное пособие

Нижний Новгород
2017

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Нижегородский государственный архитектурно-строительный университет»

Н. Ю. Прокопенко

СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

на базе Deductor Studio Academic 5.3

Утверждено редакционно-издательским советом университета
в качестве учебного пособия

Нижний Новгород
ННГАСУ
2017

ББК 32.813я73
П78
УДК 004.89(075.8)

Рецензенты:

Цветкова И.Н. – к.ф.-м.н., доцент, зав. кафедрой информатики и ИТ Нижегородского инсти тута управления Российской академии народного хозяйства и государст венной службы при Президенте РФ (НИУ РАНХиГС).

Елесин А.В. – к.ф.-м.н., ведущий сотрудник ННИМ ННГУ.

Прокопенко Н.Ю. Системы поддержки принятия решений [Электронный ресурс]: учеб. пособие /Н. Ю. Прокопенко; Нижегор. гос. архитектур.-строит. ун-т. – Н. Новгород: ННГАСУ, 2017. – 188 с. ISBN 978-5-528-00202-6. 1 электрон. опт. диск (DVD+R)

Рассматриваются вопросы автоматизации информационной подготовки принятия управленческих решений с использованием современных инструментальных средств, основные этапы проектирования и сопровождения информационных хранилищ, технологии оперативного и интеллектуального анализа данных, генетические алгоритмы и модели знаний в экспертных системах. Даются теоретические и практические основы использования аналитической платформы Deductor Studio Academic 5.3. Приведено много примеров, иллюстрирующих разработку и применение рассматриваемых методов и моделей.

Предназначено для подготовки студентов бакалавриата по направлению 09.03.03 Прикладная информатика, профиль «Прикладная информатика в экономике».

ISBN 978-5-528-00202-6

Содержание

Введение

Системы поддержки принятия решений (СППР) – компьютерные автоматизированные системы, целью которых является помощь людям, принимающим решение в сложных условиях, для полного и объективного анализа предметной деятельности. СППР возникли в результате слияния управленческих информационных систем и систем управления базами данных.

В СППР используются разные методы: информационный поиск, интеллектуальный анализ данных, поиск знаний в базах данных, рассуждение на основе прецедентов, имитационное моделирование, эволюционные вычисления и генетические алгоритмы, нейронные сети, ситуационный анализ, когнитивное моделирование и др. Некоторые из этих методов были разработаны в рамках искусственного интеллекта. Близкие к СППР классы систем – это экспертные системы и автоматизированные системы управления.

В данном пособии рассматриваются вопросы автоматизации информационной подготовки принятия управленческих решений с использованием свободно распространяемой аналитической платформы Deductor Studio Academic 5.3 (http://www.basegroup.ru). Реализованные в АП Deductor технологии позволяют пройти все этапы построения аналитической системы поддержки принятия решений: от создания хранилища данных до автоматического подбора моделей и визуализации полученных результатов.

Также в пособии рассматриваются генетические алгоритмы, вопросы инженерии знаний, автоматизированного получения логических выводов на основании вновь поступающих фактов и формализованных экспертных знаний.

Знание принципов построения современных систем поддержки принятия решений (СППР) на основе технологий хранилищ данных (Data Warehousing), оперативного анализа (OLAP) и добычи данных (Data Mining) для аналитической поддержки процессов принятия решений является необходимым компонентом в подготовке IT-специалистов.

1. История систем поддержки принятия решений

Система поддержки принятия решений − это компьютерная система, которая путем сбора и анализа большого количества информации может влиять на процесс принятия решений организационного плана в бизнесе и предпринимательстве. Интерактивные системы позволяют руководителям получить полезную информацию из первоисточников, проанализировать ее, а также выявить существующие бизнес-модели для решения определенных задач. С помощью СППР можно проследить за всеми доступными информационными активами, получить сравнительные значения объемов продаж, спрогнозировать доход организации при гипотетическом внедрении новой технологии, а также рассмотреть все возможные альтернативные решения.

Система поддержки принятия решений − комплекс математических и эвристических методов и моделей, объединенных общей методикой формирования альтернатив управленческих решений в организационных системах, определения последствий реализации каждой альтернативы и обоснования выбора наиболее приемлемого управленческого решения.

Поддержка принятия решений и заключается в помощи лицу, принимающему решение (ЛПР), в процессе принятия решений. Она включает:

- помощь ЛПР при анализе объективной составляющей, то есть в понимании и оценке сложившейся ситуации, и ограничений, накладываемых внешней средой;
- выявление предпочтений ЛПР, то есть выявление и ранжирование приоритетов, учет неопределенности в оценках ЛПР и формирование его предпочтений;
- генерацию возможных решений, то есть формирование списка альтернатив;
- оценку возможных альтернатив, исходя из предпочтений ЛПР, и ограничений, накладываемых внешней средой;

– анализ последствий принимаемых решений;
– выбор лучшего с точки зрения ЛПР варианта.

СППР в большинстве случаев это интерактивная автоматизированная система, которая помогает ЛПР использовать данные и модели для идентификации и решения задач и принятия решений. Система должна обладать возможностью работать с интерактивными запросами с достаточно простым для изучения языком.

СППР обладает следующими четырьмя основными характеристиками:

1) использует и данные, и модели;
2) помогает менеджерам в принятии решений для слабоструктурированных и неструктурированных задач;
3) поддерживает, а не заменяет выработку решений менеджерами;
4) повышает эффективность решений.

Идеальная СППР:

1) оперирует со слабоструктурированными решениями;
2) предназначена для ЛПР различного уровня;
3) может быть адаптирована для группового и индивидуального использования;
4) поддерживает как взаимозависимые, так и последовательные решения;
5) поддерживает три фазы процесса решения: интеллектуальную, проектирование и выбор;
6) поддерживает разнообразные стили и методы решения, что может быть полезно при решении задачи группой ЛПР;
7) является гибкой и адаптируется к изменениям как организации, так и ее окружения;
8) проста в использовании и модификации;
9) улучшает процесс принятия решений;
10) позволяет человеку управлять процессом принятия решений с помощью компьютера;

11) поддерживает эволюционное использование и легко адаптируется к изменяющимся требованиям;
12) может быть легко построена, если сформулирована логика конструкции СППР;
13) поддерживает моделирование;
14) позволяет использовать знания.

Компьютерная поддержка процесса принятия решений так или иначе основана на формализации методов получения рекомендаций, даваемых ЛПР, и алгоритмизации самого процесса выработки решения.

Формализация методов генерации решений, их оценка и согласование являются чрезвычайно сложной задачей. Эта задача стала интенсивно решаться с возникновением вычислительной техники.

В конце 60-х годов появляется новый тип информационных систем (ИС) – модель-ориентированные СППР (Model-oriented Decision Support Systems – DSS) или системы управленческих решений (Management Decision Systems – MDS) и соответствующая им информационная технология.

В 70-х годах были разработаны критерии проектирования систем поддержки принятия решений (СППР) в менеджменте, а также аспекты создания СППР: анализ, проектирование, внедрение, оценка и разработка.

В 1981 г. R. Sprague и E. Carlson описали, каким образом на практике можно построить СППР. Тогда же была разработана информационная система руководителя (Executive Information System – EIS) – компьютерная система, предназначенная для обеспечения текущей адекватной информации для поддержки принятия управленческих решений менеджером.

Появление в начале 80-х годов персональных компьютеров позволило автоматизировать ведение учета и обработку данных даже небольшим компаниям, не имеющим высококвалифицированного управленческого и технического персонала. Для этой категории потребителей программного обеспечения были созданы приложения нового, коммерческого типа, интегрирующие несколько

разных функций и позволяющие нескольким частям приложения манипулировать единожды введенными данными.

Начиная с 1990-х годов разрабатываются так называемые Data Warehouses (Хранилища данных).

В 1993 г. Е. Коддом (E.F. Codd) для СППР специального вида был предложен термин OLAP (Online Analytical Processing) – оперативный анализ данных, онлайновая аналитическая обработка данных для поддержки принятия важных решений. Исходные данные для анализа представлены в виде многомерного куба, по которому можно получать нужные разрезы – отчеты. Выполнение операций над данными осуществляется OLAP-машиной. По способу хранения данных различают MOLAP, ROLAP и HOLAP. По месту размещения OLAP-машины различаются OLAP-клиенты и OLAP-серверы. OLAP-клиент производит построение многомерного куба и вычисления на клиентском ПК, а OLAP-сервер получает запрос, вычисляет и хранит агрегатные данные на сервере, выдавая только результаты.

27 октября 2005 г. в Москве на Международной конференции «Информационные и телемедицинские технологии в охране здоровья» (ITTHC − 2005) А. Пастухов (Россия) представил СППР нового класса – PSTM (Personal Information Systems of Top Managers). Основным отличием PSTM от существующих СППР является построение системы для конкретного лица, принимающего решение, с предварительной логико-аналитической обработкой информации в автоматическом режиме и выводом ее на один экран.

При создании СППР учитывается ряд принципов:

1. Машина должна вычислять, рассчитывать варианты, а человек − принимать решение.
2. Принцип Шоу: система должна быть такой, чтобы с ней мог работать даже неподготовленный пользователь.

3. Принцип «бюрократичности». Этот принцип связан с уменьшением потока информации, который должен доставляться человеку для принятия решения.
4. Принцип объектно-ориентированного моделирования при построении картины предметной области.
5. Принцип динамической структуры.
6. Принцип полноты информационного пространства.
7. Принцип интеграции информационного пространства.
8. Принцип децентрализации информационного хранилища.
9. Принцип компонентной сборки прикладных режимов.

Поскольку принципы противоречивы, нужно искать компромисс между каждым из них.

Классификации СППР

Для СППР отсутствует не только единое общепринятое определение, но и исчерпывающая классификация. Разные авторы предлагают разные классификации.

На уровне пользователя Haettenschwiler (1999) делит СППР на пассивные, активные и кооперативные. Пассивной СППР называется система, которая помогает процессу принятия решения, но не может вынести предложение, какое решение принять. Активная СППР может сделать предложение, какое решение следует выбрать. Кооперативная позволяет ЛПР изменять, пополнять или улучшать решения, предлагаемые системой, посылая затем эти изменения в систему для проверки. Система изменяет, пополняет или улучшает эти решения и посылает их опять пользователю. Процесс продолжается до получения согласованного решения.

На концептуальном уровне Power (2003) отличает СППР, управляемые сообщениями (Communication-Driven DSS), СППР, управляемые данными (Data-Driven DSS), СППР, управляемые документами (Document-Driven DSS), СППР, управляемые знаниями (Knowledge-Driven DSS) и СППР, управляемые

моделями (Model-Driven DSS). СППР, управляемые моделями, характеризуются в основном доступ и манипуляции с математическими моделями (статистическими, финансовыми, оптимизационными, имитационными). Отметим, что некоторые OLAP-системы, позволяющие осуществлять сложный анализ данных, могут быть отнесены к гибридным СППР, которые обеспечивают моделирование, поиск и обработку данных.

Управляемая сообщениями (Communication-Driven DSS) (ранее групповая СППР – GDSS) СППР поддерживает группу пользователей, работающих над выполнением общей задачи.

СППР, управляемые данными (Data-Driven DSS) или СППР, ориентированные на работу с данными (Data-oriented DSS) (также известные как Business Intelligence) в основном ориентируются на доступ и манипуляции с данными. СППР, управляемые документами (Document-Driven DSS), управляют, осуществляют поиск и манипулируют неструктурированной информацией, заданной в различных форматах. Наконец, СППР, управляемые знаниями, (Knowledge-Driven DSS), обеспечивают решение задач в виде фактов, правил, процедур.

На техническом уровне Power (1997) различает СППР всего предприятия и настольную СППР. СППР всего предприятия подключена к большим хранилищам информации и обслуживает многих менеджеров предприятия. Настольная СППР – это малая система, обслуживающая лишь один компьютер пользователя.

В зависимости от данных, с которыми эти системы работают, СППР условно можно разделить на оперативные и стратегические. Оперативные СППР предназначены для немедленного реагирования на изменения текущей ситуации в управлении финансово-хозяйственными процессами компании. Стратегические СППР ориентированы на анализ значительных объемов разнородной информации, собираемых из различных источников. Важнейшей целью этих СППР является поиск наиболее рациональных вариантов развития бизнеса компании с учетом влияния различных факторов, таких как конъюнктура целе-

вых для компании рынков, изменения финансовых рынков и рынков капиталов, изменения в законодательстве и др. СППР первого типа получили название Информационных Систем Руководства (Executive Information Systems, ИСР). По сути они представляют собой конечные наборы отчетов, построенные на основании данных из транзакционной информационной системы предприятия, в идеале адекватно отражающей в режиме реального времени основные аспекты производственной и финансовой деятельности. Для ИСР характерны следующие основные черты:

- отчеты, как правило, базируются на стандартных для организации запросах; число последних относительно невелико;
- ИСР представляет отчеты в максимально удобном виде, включающем, наряду с таблицами, деловую графику, мультимедийные возможности;
- как правило, ИСР ориентированы на конкретный рынок, например финансы, маркетинг, управление ресурсами.

СППР второго типа предполагают достаточно глубокую проработку данных, специально преобразованных так, чтобы их было удобно использовать в ходе процесса принятия решений. Неотъемлемым компонентом СППР этого уровня являются правила принятия решений, которые на основе агрегированных данных дают возможность менеджерам компании обосновывать свои решения, использовать факторы устойчивого роста бизнеса компании и снижать риски. СППР второго типа в последнее время активно развиваются. Технологии этого типа строятся на принципах многомерного представления и анализа данных (OLAP).

При создании СППР можно использовать Web-технологии. В настоящее время СППР на основе Web-технологий для ряда компаний являются синонимами СППР предприятия.

Структура СППР

В состав СППР входят следующие компоненты (рис. 1): источники данных, модель данных, база моделей и программная подсистема, которая состоит

из системы управления базой данных (СУБД), системы управления базой моделей (СУБМ) и системы управления интерфейсом между пользователем и компьютером.

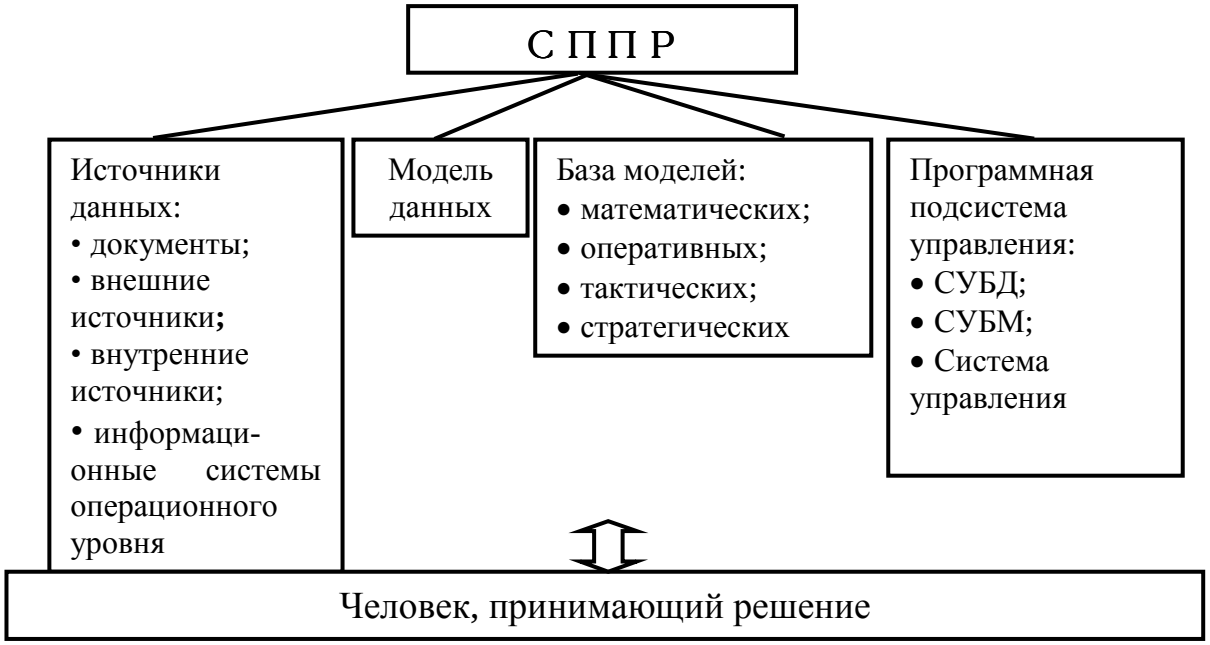


Рис.1. Структура СППР

Архитектура СППР представляется разными авторами по-разному. Например, Marakas (1999) предложил обобщенную архитектуру, состоящую из пяти различных частей: (a) система управления данными (the data management system – DBMS), (b) система управления моделями (the model management system – MBMS), (c) машина знаний (the knowledge engine (KE)), (d) интерфейс пользователя (the user interface) и (e) пользователи (the user(s)).

Информационной платформой СППР являются ***хранилища данных*** (Data Warehouse).

В основной ***функциональный набор*** СППР входят:

- формирование консолидированной отчетности (до 200 преднастроенных отчетов);
- многомерный анализ данных (OLAP);
- выявление скрытых закономерностей (Data Mining);
- статистический анализ и прогнозирование временных рядов;

– формирование преднастроенных запросов (до 500 – 600);
– интеллектуальный поиск (по неполным данным и неформальным запросам).

Инструментальная среда СППР – интеграционные системы, основанные на открытых стандартах. Эти системы соответствуют требованиям:

– информационной безопасности;
– масштабируемости;
– открытости;
– многомерного и многовариантного представления данных;
– интеллектуального интерфейса;
– интегрируемости с основными платформами и бизнес-приложениями, интеграция данных из разнообразных источников, сетевая интеграция (прежде всего web);
– обеспечивают сервис по «очистке» данных при их загрузке в хранилища.

Техническое обеспечение связано с:

– обработкой данных;
– надежным хранением данных и обеспечением целостности;
– архивацией и восстановлением данных;
– сетевым и телекоммуникационным обеспечением;
– криптографическим обеспечением;
– управлением доступом пользователей;
– загрузкой данных, в том числе с использованием средств интеллектуального интерфейса (распознавание образов: текста, речи, изображений).

Различные подходы к построению СППР

I. *Логический подход*

Основой для логического подхода служат Булева алгебра и нечеткая логика. Каждый программист знаком с нею и с логическими операторами (пример – оператор IF). Свое дальнейшее развитие Булева алгебра получила в виде ис-

числения предикатов, в котором она расширена за счет введения предметных символов, отношений между ними, кванторов существования и всеобщности. Практически каждая система СППР, построенная на логическом принципе, представляет собой машину доказательства теорем. При этом исходные данные хранятся в базе данных в виде аксиом, правила логического вывода – как отношения между ними. Кроме того, каждая такая машина имеет блок генерации цели, и система вывода пытается доказать данную цель как теорему. Если цель доказана, то трассировка примененных правил позволяет получить цепочку действий, необходимых для реализации поставленной цели. Мощность такой системы определяется возможностями генератора целей и машиной доказательства теорем.

Нечеткая логика. Основным ее отличием является то, что правдивость высказывания может принимать в ней, кроме да/нет (1/0), еще и промежуточные значения – не знаю (0.5): пациент скорее жив, чем мертв (0.75), пациент скорее мертв, чем жив (0.25). Данный подход больше похож на мышление человека, поскольку он на вопросы редко отвечает только «да» или «нет».

Для большинства логических методов характерна *большая трудоемкость, поскольку во время поиска доказательства возможен полный* перебор вариантов. Поэтому данный подход требует эффективной реализации вычислительного процесса и хорошая работа обычно гарантируется при сравнительно небольшом размере базы данных.

II. Под *структурным подходом* подразумевают попытки построения СППР путем моделирования структуры человеческого мозга. Такие информационные системы называются и*нтеллектуальными*.

Термин artificial intelligence (AI 1956 г.) означает «умение рассуждать разумно», а не «искусственный интеллект intellect» и обычно трактуется, как свойство автоматических систем выполнять отдельные разумные действия, свойственные человеку. Например, выбирать и принимать правильные решения

на основе ранее полученного опыта и (или) рационального анализа внешних воздействий.

III. Довольно большое распространение получил и *эволюционный подход*. При построении СППР по данному подходу основное внимание уделяется построению начальной модели и правилам, по которым она может изменяться (эволюционировать). Причем модель может быть составлена по самым различным методам, это может быть и ИНС, и набор логических правил, и модель, где есть генетические алгоритмы. После этого мы включаем компьютер, и он на основании проверки моделей отбирает самые лучшие, используя которые, по самым различным правилам генерируются новые модели, из которых опять выбираются самые лучшие, т. д.

В принципе можно сказать, что эволюционных моделей как таковых не существует, существуют только эволюционные алгоритмы обучения, но модели, полученные при эволюционном подходе, имеют некоторые характерные особенности, что позволяет выделить их в отдельный класс.

Такими особенностями являются *перенесение основной работы разработчика с построения модели на алгоритм ее модификации* и то, что полученные модели практически не сопутствуют извлечению новых знаний о среде, окружающей систему, то есть она становится как бы вещью в себе.

IV. Еще один широко используемый подход к построению СППР – *имитационный*. Данный подход является классическим для кибернетики с одним из ее базовых понятий – «черным ящиком» (ЧЯ). ЧЯ – устройство, программный модуль или набор данных, информация о внутренней структуре и содержании которых отсутствуют полностью, но известны спецификации входных и выходных данных. Объект, поведение которого имитируется, как раз и представляет собой такой «черный ящик». Нам не важно, что у него и у модели внутри и как он функционирует, главное, чтобы наша модель в аналогичных ситуациях вела себя точно так же.

Таким образом, здесь моделируется другое свойство человека – способ-

ность копировать то, что делают другие, не вдаваясь в подробности, зачем это нужно. Зачастую эта способность экономит ему массу времени, особенно в начале его жизни.

Основным недостатком имитационного подхода также является *низкая информационная способность* большинства моделей, построенных с его помощью.

Различные подходы существуют и сейчас. На практике очень четкой границы между ними нет. Очень часто встречаются смешанные системы, где часть работы выполняется по одному типу, а часть – по другому.

Вопросы для самопроверки

1. Определите понятие «Система поддержки принятия решений».
2. Типы СППР. Архитектура СППР.
3. Основные компоненты СППР. Какие подсистемы входят в системы поддержки принятия решений?
4. Как можно классифицировать систему поддержки принятия решений?
5. Функции систем поддержки принятия решений в оценке существующих и гипотетических ситуаций, в которых функционирует предприятие.
6. Какие системы поддержки принятия решений позволяют модифицировать решения системы, опирающиеся на большие объемы данных из разных источников?
7. Какие бывают архитектуры систем поддержки принятия решений?
8. Принципы загрузки, верификации и очистки данных.
9. Понятие качества данных. Основные причины низкого качества данных в СППР.
10. Методы и средства повышения качества исходных данных.
11. Охарактеризуйте возможные условия, в которых менеджеру приходится принимать решения: определенность, риск, неопределенность.

12. В чем проявляется неопределенность при принятии экономических решений?
13. Почему автоматизация процесса разработки и принятия решения может повысить его эффективность?
14. Любой ли процесс принятия решения можно полностью автоматизировать?
15. Каковы преимущества и недостатки автоматизации процесса разработки, принятия и реализации управленческого решения?

2. Создание систем поддержки принятия решений на основе хранилищ данных

Стратегические СППР, основанные на анализе большого количества информации из разных источников с привлечением сведений, содержащихся в системах, аккумулирующих опыт решения проблем, предполагают глубокую проработку данных, специально преобразованных так, чтобы их было удобно использовать в ходе процесса принятия решений. Неотъемлемым компонентом СППР этого уровня является хранилище данных.

Хранилище данных – разновидность систем хранения, ориентированная на поддержку процесса анализа данных, обеспечивающая целостность, непротиворечивость и хронологию данных, а также высокую скорость выполнения аналитических запросов.

Использование концепции хранилища данных (ХД) в СППР и анализе данных способствует достижению таких целей, как:

– своевременное обеспечение аналитиков и руководителей всей информацией, необходимой для выработки обоснованных и качественных управленческих решений;

– создание единой модели представления данных в организации;

– создание интегрированного источника данных, предоставляющего удобный доступ к разнородной информации и гарантирующего получение оди-

наковых ответов на одинаковые запросы из различных аналитических приложений.

Первой фазой разработки ХД является системный анализ объекта принятия решений. Например, в практике СППР для бизнеса известно два подхода к такому анализу. Первый ориентируется на описание *бизнес-процессов*, протекающих на предприятии, которое моделируется набором взаимосвязанных функциональных элементов. Второй подход основан на первичном анализе *бизнес-событий*. При проектировании СППР на основе ХД именно он обеспечивает наибольшую эффективность:

Через анализ бизнес-событий необходимо перейти к анализу данных. При этом должна быть собрана информация об используемых внешних данных и их источниках; о форматах данных, периодичности и форме их поступления; о внутренних информационных системах объекта, обслуживаемого СППР, их функциях и алгоритмах обработки данных, используемых при наступлении бизнес-событий. Такой анализ, как правило, производится при проектировании любой информационной системы. Особенность анализа данных при проектировании СППР состоит в необходимости создания моделей представления информации. То, что в обычных информационных системах является вторичным понятием, а именно состав и форма отображаемых данных, в СППР приобретает особую важность, так как нужно выявить все без исключения признаки, требуемые для принятия решений.

Модель представления данных является организационно-функциональным срезом модели системы, а при ее разработке последовательно изучаются:

– распределение пользователей системы: географическое, организационное, функциональное;

– доступ к данным: объем данных, необходимый для анализа, уровень агрегированности данных, источники данных (внешние или внутренние), описа-

ние информации, совместно используемой различными функциональными группами пользователей;

– аналитические характеристики системы: измерения данных, основные отчеты, последовательность преобразования аналитической информации, степень предопределенности анализа, существующие или находящиеся в стадии разработки средства анализа.

По результатам анализа бизнес-процессов и структур данных отбирается действительно значимая для принятия решений информация с учетом неопределенности будущих запросов.

Следующий шаг связан с пониманием того, в каком виде и на каких аппаратных и программных платформах размещать структуру данных СППР на основе хранилищ данных.

Хранилище данных Deductor Warehouse – это специально организованная база данных, ориентированная на решение задач анализа данных и поддержки принятия решений, обеспечивающая максимально быстрый и удобный доступ к информации.

Объекты хранилища данных Deductor Warehouse перечислены далее.

Измерение – это последовательность значений одного из анализируемых параметров. Например, для параметра «время» – это последовательность календарных дней, для параметра «место проживания» – список названий городов. Каждое значение измерения может быть представлено координатой в многомерном пространстве процесса, например город, клиент, дата.

Атрибут – это свойство измерения (т.е. точки в пространстве). Атрибут как бы скрыт внутри другого измерения и помогает пользователю полнее описать исследуемое измерение. Например, для измерения «Код региона» атрибутом является «Регион».

Факт – значение, соответствующее измерению. Факты – это данные, отражающие сущность события. В большинстве случаев фактами являются численные значения, например сумма, количество, объем.

Ссылка на измерение – это установленная связь между двумя и более измерениями. Некоторые понятия, соответствующие измерениям в хранилище данных, могут образовывать иерархии. Например, «Недвижимость» может включать «Новостройки» и «Вторичное жилье», которые, в свою очередь, подразделяются на группы. В этом случае первое измерение содержит ссылку на второе, второе – на третье и т.д.

Процесс – совокупность измерений, фактов и атрибутов. Процесс описывает определенное действие, например продажа, отгрузка, мониторинг. *Атрибут процесса* – свойство процесса. Атрибут процесса в отличие от измерения не определяет координату в многомерном пространстве. Это справочное значение, относящееся к процессу. Значение атрибута процесса, в отличие от измерения, может быть не всегда определено.

Все загружаемые в ХД данные обязательно должны быть определены как измерение, атрибут либо факт.

Принадлежность данных к типу (измерение, ссылка на измерение, атрибут или факт) содержится в семантическом слое хранилища.

Семантический слой – механизм, позволяющий оперировать данными посредством терминов предметной области.

2.1. Практическая работа «Создание ХД в аналитической платформе Deductor Studio Academic 5.3»

Имеется история продаж и поступлений различных товаров по дням в нескольких торговых объектах. Данные представлены текстовыми файлами, которые включают в себя выгрузку информации о поступлении товара, продаже товара и предоставляемую скидку при продаже товара, а также справочники: Артикулы, Единицы измерения, Группы клиентов, Группы товаров, Номера клиентов, Обобщенные группы товаров, Список городов.

Артикулы (фрагмент) Таблица 1

Артикул	Наименование товара	Группа товаров	Группа товаров
			Обобщенная группа товаров
108006	Балка декоративная DECOSA Рустик 120x120x20(	Потолочные покрытия	Отделочные материалы
108003	Балка декоративная DECOSA Рустик 60x90x2000,	Потолочные покрытия	Отделочные материалы
108004	Балка декоративная DECOSA Рустик 60x90x3000,	Потолочные покрытия	Отделочные материалы
108005	Балка декоративная DECOSA Рустик 60x90x4000,	Потолочные покрытия	Отделочные материалы
107902	Бамбук половина ствола, диаметр 30-40 мм, 2 м,	Стеновые покрытия	Отделочные материалы
105423	Болт 10x20 цинк, шестигранная головка, 8 штук, 8	Метизы и крепёж	Метизы и крепёж
105412	Болт 6x80 цинк, шестигранная головка, 3 штуки, 3	Метизы и крепёж	Метизы и крепёж
105413	Болт 6x90 цинк, шестигранная головка, 6 штук, 6	Метизы и крепёж	Метизы и крепёж

Группа клиентов Таблица 2

Группа клиентов
VIP клиент
Клиент
Постоянный клиент

Группа товаров (фрагмент) Таблица 3

Группа товаров	Обобщенная группа товаров
Армирующие материалы	Армирующие материалы
Гидроизоляция	Изоляционные материалы
Грунтовка	Лакокрасочные материалы и строительная химия
Изоляция	Изоляционные материалы
Краска	Лакокрасочные материалы и строительная химия
Лаки, морилки	Лакокрасочные материалы и строительная химия
Металлолом	Металлолом
Метизы и крепёж	Метизы и крепёж
Напольные покрытия	Отделочные материалы
Плитка	Отделочные материалы
Потолочные покрытия	Отделочные материалы
Профиль заказной	Металлический профиль и комплектующие
Профиль и комплектующие для вентфасадов	Металлический профиль и комплектующие PRIMET
Профиль и комплектующие для ГКЛ	Металлический профиль и комплектующие PRIMET

Единицы измерения Таблица 4

Единица измерения
кв.м
кг
л
м
уп
шт

Клиенты (фрагмент) Таблица 5

Дата продажи	№ Клиента	Артикул	Единица измерения	№ Клиента			
				Город	Город		Группа клиентов
					Экономический район	Федеральный округ	
01.03.2004	00000010	105285	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105290	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105291	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105412	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105417	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105418	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент

Клиенты (фрагмент) Таблица 5 (продолжение)

Артикул		Цена за единицу	Количество	Сумма с учетом скидки
Наименование товара	Группа товаров			
Эмаль алкидная SADOLIN Master-30 WO, 1л, ц	Эмаль	213,8	40	8124,4
Эмаль алкидная MARSHALL Pastel Yarimat 111	Эмаль	161,18	40	6124,84
Эмаль алкидная MARSHALL Enamel Parlak для	Эмаль	124,36	40	4974,4
Эмаль алкидная MARSHALL Enamel Parlak для	Эмаль	124,36	34	4101,3928
Эмаль алкидная ALPINA Direkt auf Rost антико	Эмаль	184,37	100	17515,15
Эмаль алкидная MARSHALL Enamel Parlak для	Эмаль	124,36	40	4974,4
Эмаль НЦ-132П OLEKOLOR черная, 1.8 кг, шт	Эмаль	135,11	76	9754,942

Номера клиентов (фрагмент) Таблица 6

№ Клиента	Город	Группа клиента
00000003	Москва	Клиент
00000004	Тверь	Клиент
00000010	Нижний Новгород	Постоянный клиент
00000011	Балашиха	Клиент
00000014	Москва	Клиент
00000015	Нижний Новгород	Постоянный клиент
00000016	Нижний Новгород	Клиент
00000017	Москва	Клиент
00000019	Владимир	VIP клиент
00000020	Кострома	VIP клиент
00000031	Москва	Клиент
00000032	Москва	Клиент
00000038	Мытищи	Клиент

Обобщенные группы товаров Таблица 7

Обобщенные группы товаров
Армирующие материалы
Изоляционные материалы
Лакокрасочные материалы и строительная химия
Металлический профиль и комплектующие
Металлический профиль и комплектующие PRIMET
Металлолом
Метизы и крепёж
Отделочные материалы
Сыпучие и вяжущие материалы и смеси

Приход (фрагмент) Таблица 8

Дата поставки	№ Счет-фактуры	Номер поставщика	Артикул	Артикул			Цена за единицу	Количество
				Наименование товара	Группа товаров	Группа товаров / Обобщенная группа товаров		
01.03.2004	553	5	100429	Лак акриловый ОПТИМИСТ бесцвет	Лаки, морилки	Лакокрасочные материалы и	140,3	81
01.03.2004	553	5	105025	Краска колеровочная ПРЕМИЯ черна	Тонер, колер	Лакокрасочные материалы и	88,76	92
01.03.2004	3997	3	105412	Болт 6х80 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	9,55	225
01.03.2004	3997	3	105414	Болт 8х25 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	14,69	225
01.03.2004	3997	3	105415	Болт 8х35 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	16,73	225
01.03.2004	3997	3	105417	Болт 8х50 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	16,73	225
01.03.2004	3997	3	105419	Болт 8х70 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	21,13	226
01.03.2004	3997	3	105421	Болт 8х90 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	11,59	225
01.03.2004	3997	3	105422	Болт 8х100 цинк, шестигранная голо	Метизы и крепёж	Метизы и крепёж	11,59	226
01.03.2004	3997	3	105556	Дюбель с шурупом забивной WKRET	Метизы и крепёж	Метизы и крепёж	66,69	225
01.03.2004	9085	4	102438	Профиль перегородочный стоечный (	Профиль ПРОФИ	Металлический профиль и ко	22,36	137
01.03.2004	9085	4	102439	Профиль перегородочный стоечный (	Профиль СТАНДА	Металлический профиль и ко	18,86	138

Скидка (фрагмент) Таблица 9

Дата продажи	№ Клиента	Артикул	№ Клиента				
			Город	Город		Группа клиентов	Наименование товара
				Экономический район	Федеральный округ		
01.03.2004	00000081	102313	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Профиль перегородочный стоечный
01.03.2004	00000081	102339	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Профиль перегородочный стоечный
01.03.2004	00000081	102578	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Профиль потолочный направляющий
01.03.2004	00000081	104361	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Лента малярная KLEBEBANDER 50м
01.03.2004	00000081	113741	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113744	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113745	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113746	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113747	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113748	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113749	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2

Артикул		Цена за единицу	Количество	Скидка по группе клиента %	Скудка по сумме покупки %	Общая скидка %
Группа товаров	Группа товаров					
	Обобщенная группа товаров					
Профиль укороченный	Металлический профиль	2,89	21	5	0	5
Профиль укороченный	Металлический профиль	2,6	21	5	0	5
Профиль укороченный	Металлический профиль	1,42	21	5	0	5
Армирующие материалы	Армирующие материалы	16,24	28	5	0	5
Напольные покрытия	Отделочные материалы	734,4	28	5	0	5
Напольные покрытия	Отделочные материалы	440,64	28	5	0	5
Напольные покрытия	Отделочные материалы	660,96	28	5	0	5

Список городов (фрагмент) Таблица 10

Город	Экономический район	Федеральный округ
Архангельск	Северный экономический район	Северо -Западный федеральный округ
Астрахань	Поволжский экономический район	Южный федеральный округ
Балашиха	Центральный район	Центральный федеральный округ
Барнаул	Западно-Сибирский округ	Сибирский федеральный округ
Белгород	Центрально-Черноземный район	Центральный федеральный округ
Великие Луки	Северный экономический район	Северо -Западный федеральный округ
Владивосток	Дальневосточный экономический район	Дальневосточный федеральный округ
Владимир	Центральный район	Центральный федеральный округ
Волгоград	Поволжский экономический район	Южный федеральный округ
Вологда	Северный экономический район	Северо -Западный федеральный округ
Воронеж	Центрально-Черноземный район	Центральный федеральный округ
Екатеринбург	Уральский экономический район	Уральский федеральный округ
Зеленоград	Центральный район	Центральный федеральный округ
Иваново	Центральный район	Центральный федеральный округ
Ижевск	Уральский экономический район	Приволжский федеральный округ

Указания

При проектировании ХД необходимо учитывать следующее:

– совокупность измерений процесса должна однозначно определять единственную запись в таблице процесса («точку» в многомерном пространстве);

– если существуют иерархии, то выбор должен быть в пользу измерения;

– если по объекту хранилища данных предполагается в будущем делать частые «срезы», то снова лучше отдать предпочтение измерению;

– таблицы измерений содержат только справочную информацию (коды, наименования и т.п.) и ссылки на другие измерения при необходимости;

– таблица процесса содержит только факты и коды измерений (без их атрибутов);

– наличие возможных пропусков (необязательное поле) говорит о том, что объект лучше сделать атрибутом процесса.

Покажем, какие данные являются измерениями, какие атрибутами, а какие фактами и что представляют собой процессы.

В табл. 1 «Артикулы» измерениями являются следующие поля: Артикул, Группа товаров, Группа товаров | Обобщенная группа товаров, а поле Наименование товара является атрибутом.

Таблица «Группа клиентов» (табл. 2) содержит в себе всего 1 поле «Группа клиентов», которое является измерением.

В таблице «Группа товаров» (табл. 3) измерениями являются следующие поля: Группа товаров, Обобщенная группа товаров.

Таблица «Единицы измерения» (табл. 4) содержит в себе всего 1 поле «Единица измерения», которое является измерением.

В таблице «Клиенты» (табл. 5) измерениями являются следующие поля: Дата продажи, Номер клиента, Артикул, Единицы измерения, Номер клиента | Город, Номер клиента | Группа клиентов, Артикул | Группа товаров. Атрибутами являются поля: Номер клиента | Город | Экономический район, Номер клиента | Город | Федеральный округ, Артикул | Наименование товара, а такие поля, как Цена за единицу, Количество и Сумма с учетом скидки являются фактами. Т.е. табл. 5 является описанием процесса продажи товаров.

В таблице «Номер клиента» (табл. 6) поля Номер клиента, Город и Группа клиентов являются измерениями.

Таблица «Обобщенная группа товаров» (табл. 7) содержит в себе всего 1 поле «Обобщенная группа товаров», которое является измерением.

В таблице «Приход» (табл. 8), измерениями являются следующие поля: Дата прихода, Артикул, Артикул|Группа товаров, Артикул|Группа товаров|Обобщенная группа товаров, Номер счет-фактуры, и Номер поставщика. Поле Артикул | Наименование товара является атрибутом, а такие поля как Цена за единицу и Количество являются фактами. Т.е. таблица 8 является описанием процесса поступления товаров.

В таблице «Скидка» (табл. 9), измерениями являются следующие поля: Дата продажи, Номер клиента, Артикул, Номер клиента|Город, Номер клиента|Группа клиентов, Артикул|Группа товаров, Артикул|Группа товаров|Обобщенная группа товаров. Атрибутами являются поля: Номер клиента|Город|Экономический район, Номер клиента|Город|Федеральный округ, Артикул|Наименование товара, а такие поля, как Цена за единицу, Количество, Скидка по группе клиента %, Скидка по сумме клиента %, Общая сумма скидки % являются фактами. Т.е. табл. 9 является описанием процесса предоставления скидки клиентам при покупке товаров.

В таблице «Список городов» (табл. 10) поле Город является измерением, а Экономический район и Федеральный округ являются его атрибутами.

Стоит отметить, что таблицу «Продажи» можно объединить с таблицей «Скидки» с помощью обработки «Слияние с узлом». Целесообразность данного действия заключается в том, что в данных таблицах хранятся практически одни и те же данные, различие лишь в том, что в одной таблице есть скидки, а в другой – сумма с учетом скидок, поэтому для наглядного отображения данных решено два процесса объединить в один процесс – Продажи.

Таким образом, было выделено два основных процесса: Поступление и Продажи товаров.

Основные этапы создания ХД в АП Deductor:

1. Для создания нового хранилища данных в Deductor или подключения к существующему нужно перейти на закладку «Подключения» и запустить «Мастер подключений» (рис. 2).

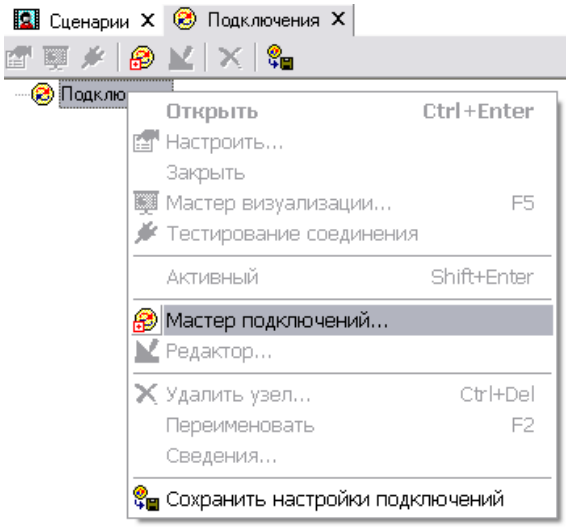


Рис. 2. Создание (подключение) хранилища данных

2. Пройти первые два шага, выбрав тип приемника (источника) Deductor Warehouse и тип базы данных Firebird.

3. На третьем шаге нужно задать параметры базы данных (рис. 3), в которой будет создана физическая и логическая структура хранилища данных: база данных – ***материалы.gdb;***

- логин – ***sysdba,*** пароль – ***masterkey***;
- установить флажок ***Сохранять пароль***.

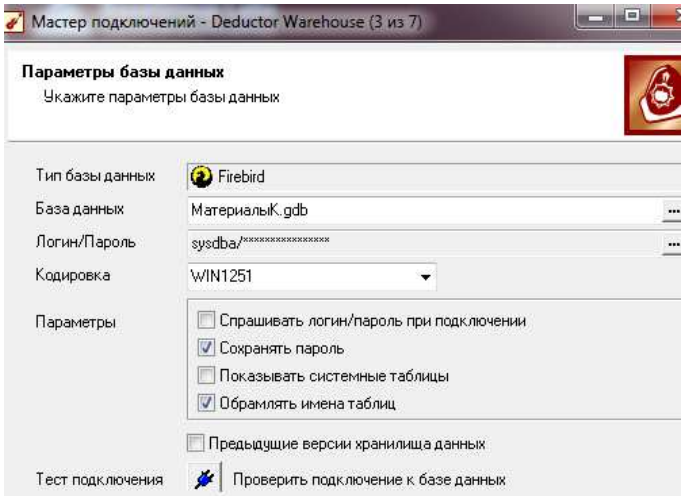


Рис. 3. Установка параметров базы данных

4. На следующем шаге выбирается версия для работы с ХД ***Deductor Warehouse 6***.

5. На пятом шаге нажать кнопку ***Создать файл базы данных с необходимой структурой метаданных***

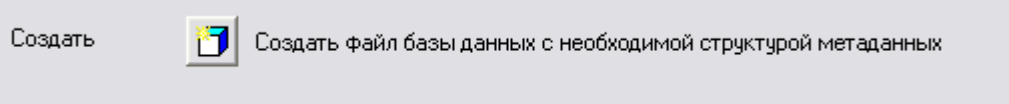


Рис. 4. Вкладка Мастера подключения «Инструменты работы с ХД»

При этом выборе по указанному ранее пути будет создан файл материалы.gdb. (появится сообщение об успешном создании). Это и есть пустое хранилище данных, готовое к работе.

6. На последних двух шагах осталось выбрать визуализатор для подключения (здесь это Сведения и Метаданные) и задать для нового хранилища имя «material», метку «Материалы» и описание «Хранилище данных с информацией о продажах».

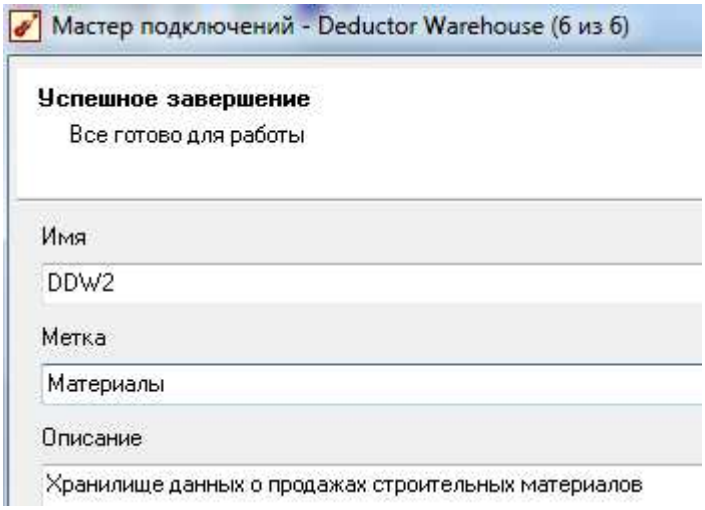


Рис. 5. Настройка семантики узла подключения

7. После нажатия на кнопку **Готово** на дереве узлов подключений появится метка хранилища.

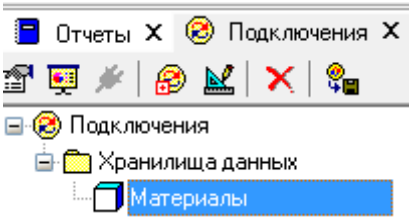


Рис.6. Хранилище данных «Материалы»

8. Для проверки доступа к новому ХД воспользуйтесь кнопкой . Если спустя некоторое время появится сообщение «Тестирование соединения прошло успешно», то хранилище готово к работе. Иначе нужно внести изменения в параметры подключения ХД, используя кнопку .

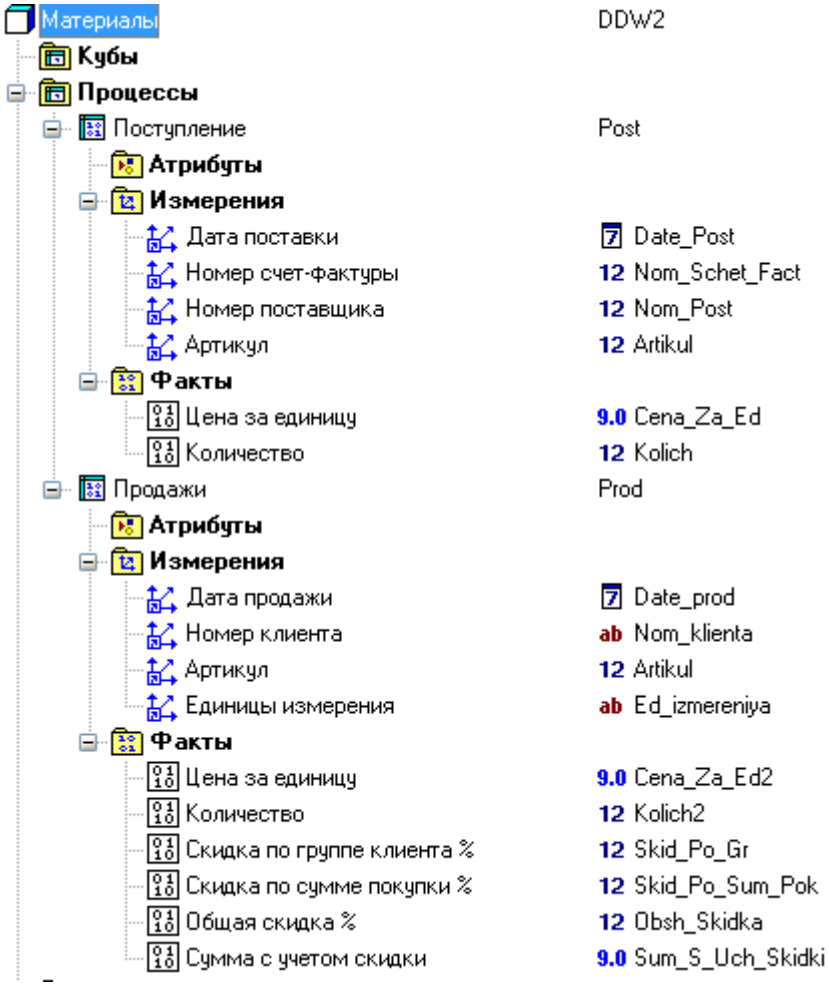
9. Сохраните настройки подключений, нажав на кнопку .

10. Для проектирования структуры ХД вызвать **Редактор метаданных** кнопкой на вкладке **Подключения**.

11. В открывшемся окне редактора нужно нажать кнопку (разрешить редактирование).

12. Встав на узле **Измерения**, при помощи кнопки **Добавить** добавьте в метаданные по очереди все объекты ХД.

Структура хранилища данных представлена на рис 7:



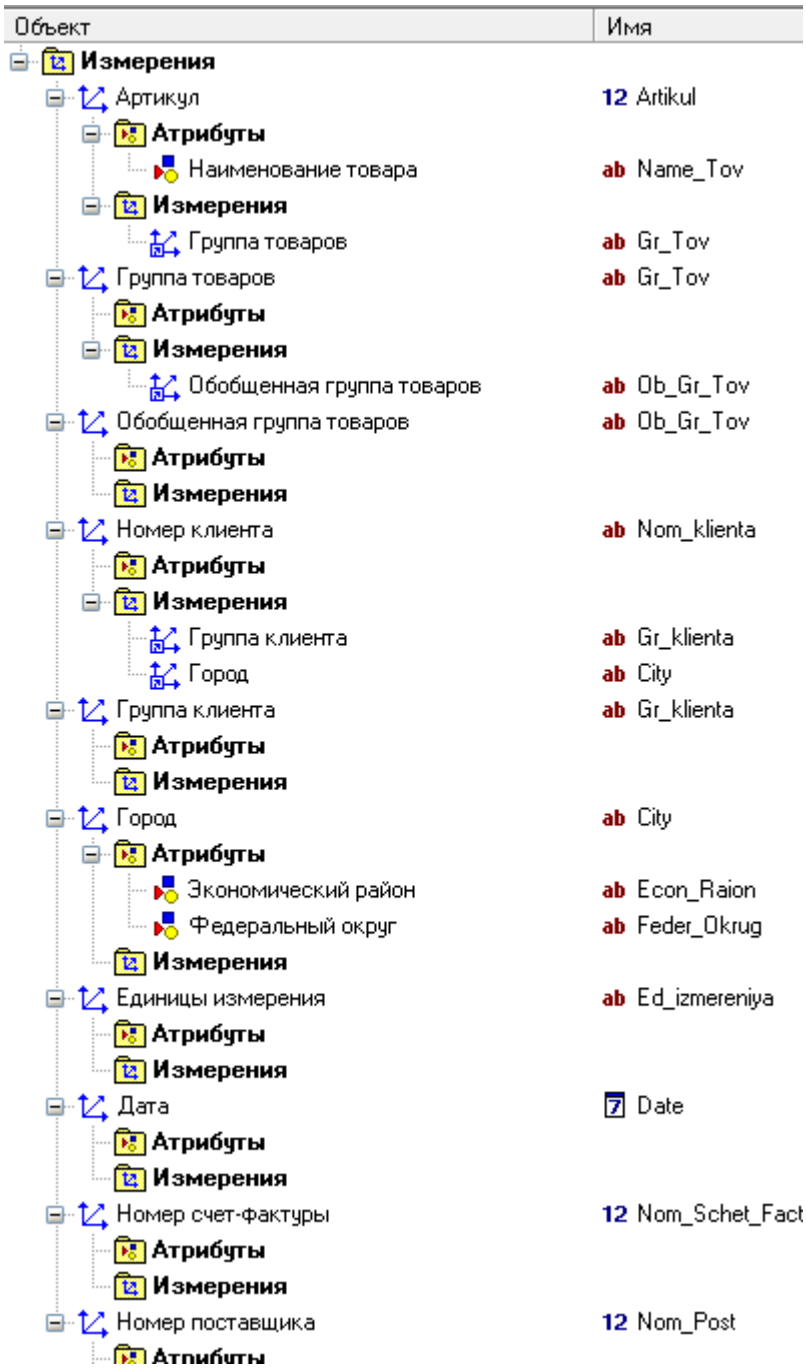


Рис. 7. Структура ХД

Наполнение ХД

Структура хранилища данных представляет собой «пустое» ХД Deductor Warehouse с настроенным семантическим слоем. В таком виде оно готово к загрузке в него данных из внешних структурированных источников. Для этого необходимо написать соответствующий сценарий в Deductor Studio.

Сценарий загрузки должен выполнять следующие функции:

1. Импорт данных в Deductor Studio из базы данных, учетной системы или предопределенных файлов;

2. Опциональная предобработка данных, например очистка или преобразование формата;

3. Загрузка данных в измерения и процессы хранилища Deductor Warehouse.

Исходными данными для ХД служат 10 текстовых файлов:

Артикул.txt, Группа клиентов.txt, Группа товаров.txt, Единицы измерения.txt, Клиенты.txt, Номер клиента.txt, Обобщенная группа товаров.txt, Приход.txt, Скидка.txt, Список городов.txt. Поэтому сценарий загрузки должен быть настроен на использование в качестве источников данных на эти файлы.

При создании сценария необходимо строго придерживаться следующих правил:

1. Первыми загружаются все измерения, имеющие атрибуты. Только после загрузки всех измерений загружаются данные в процессы.
2. Также имеется правило на порядок загрузки: загружать измерения нужно, начиная с самого верхнего уровня иерархии и спускаться по иерархии ниже, в противном случае иерархия не будет создана.
3. Допускается не загружать отдельно измерения, не имеющие атрибутов и не состоящие в иерархии измерений. Значения таких измерений можно при использовании специальной опции создавать во время загрузки в процесс.

В ходе наполнения ХД данными могут быть некоторые ошибки. Ниже представлено описание двух типов ошибок и пути их решения:

– ошибка 303 возникает в случае, когда длина поля в ХД не соответствует длине этого же поля в текстовом файле, т.е. получается ситуация, когда длинное название не помещается в хранилище из-за мало выделенного под это название места. Возможное решение: просмотреть текстовые файлы и найти поля,

которые на первый взгляд являются очень длинными. В режиме редактирования хранилища удалить и заново добавить необходимый атрибут/измерение и увеличить длину поля со стандартно заданных 100 символов до нужного размера.

– ошибка 206 возникает, когда у наполненного данными хранилища меняют структуру и снова наполняют его, не очистив хранилище от старых данных. Возможное решение: заново добавить текстовый файл, необходимый для загрузки данных в хранилище и удалить, а затем повторно добавить необходимые измерения/атрибуты, либо в режиме редактирования структуры ХД воспользоваться командой «Очистить» и удалить ненужные старые данные из выбранного измерения в Хранилище данных.

Ниже представлена схема загрузки данных:

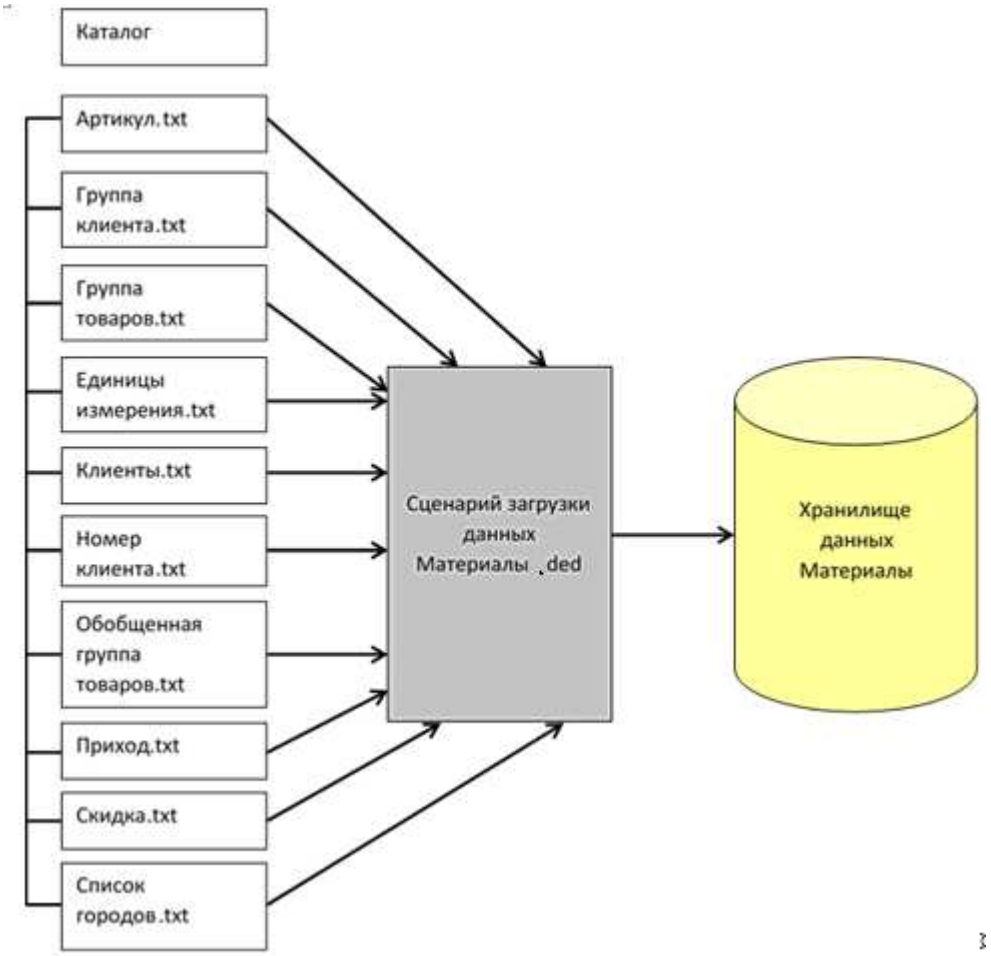


Рис. 8. Схема загрузки данных

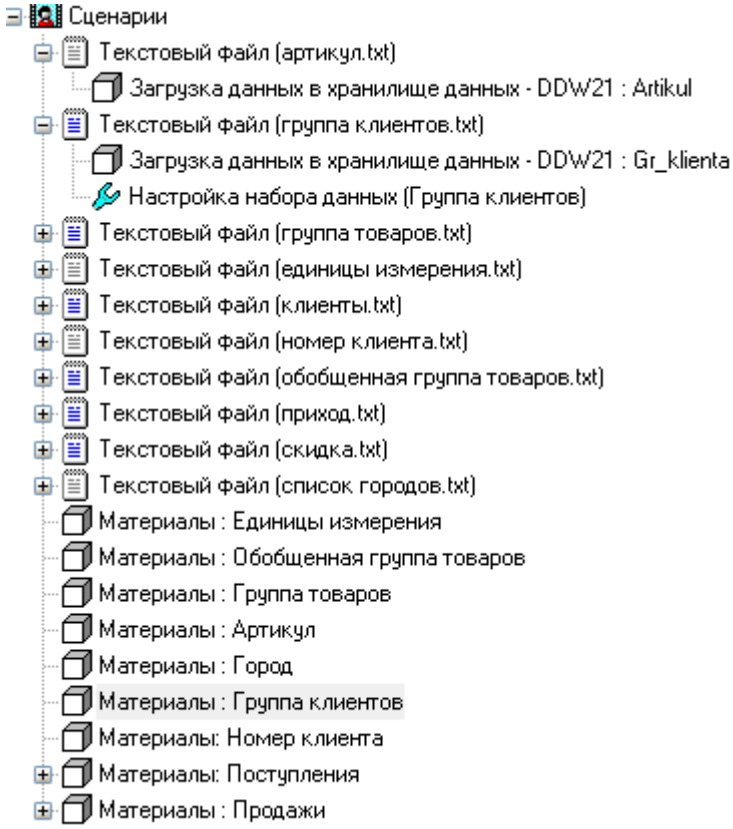


Рис. 9. Сценарий загрузки данных в хранилище

Такого рода сценарий привязан не к самим данным, а только к их структуре, то есть в нем смоделирована последовательность действий, которую нужно выполнить для загрузки данных в хранилище: указаны имена файлов источников, соответствие полей и т.д. Таким образом, сценарий может использоваться неоднократно для пополнения ХД.

Созданное ХД позволяет обеспечить целостность и непротиворечивость данных, их централизованное хранение, автоматически обеспечивает всю необходимую поддержку процесса анализа данных.

В завершении работы с ХД нужно выполнить выгрузку данных из хранилища, чтобы убедиться в правильности загруженной информации и сравнить выгруженные файлы с текстовыми файлами по количеству элементов.

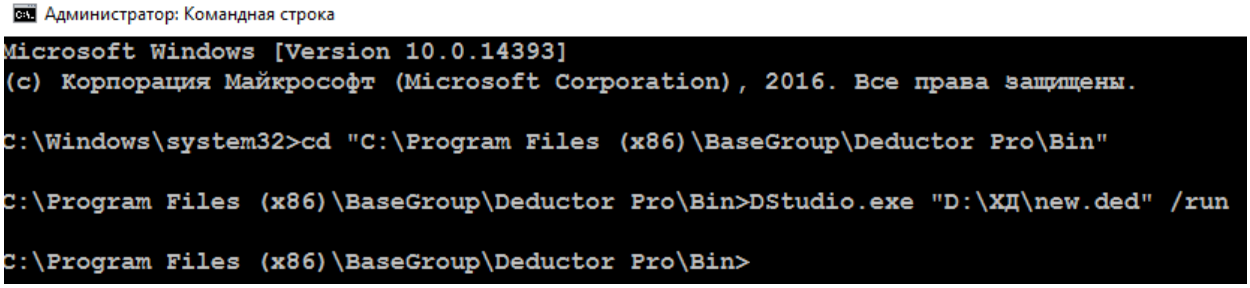
Из внешних источников данных информация в соответствии с некоторым регламентом должна перемещаться в ХД. Автоматическая загрузка в ХД настраивается с помощью пакетной обработки.

Указания

1. Чтобы запустить пакетное выполнение сценария с помощью командной строки, необходимо зайти в папку bin, которая располагается в директории установленной программы, воспользовавшись командой windows интерпретатора – cd (change directory). Форма записи:

> cd “C:\Program Files (x86)\BaseGroup\Deductor Pro\Bin”

2. После этого запустить исполняемый файл DStudio.exe, передав в качестве аргументов полный адрес к файлу сценария (в кавычках) и команду /run для запуска пакетной обработки: > DStudio.exe "D:\ХД\new.ded" /run



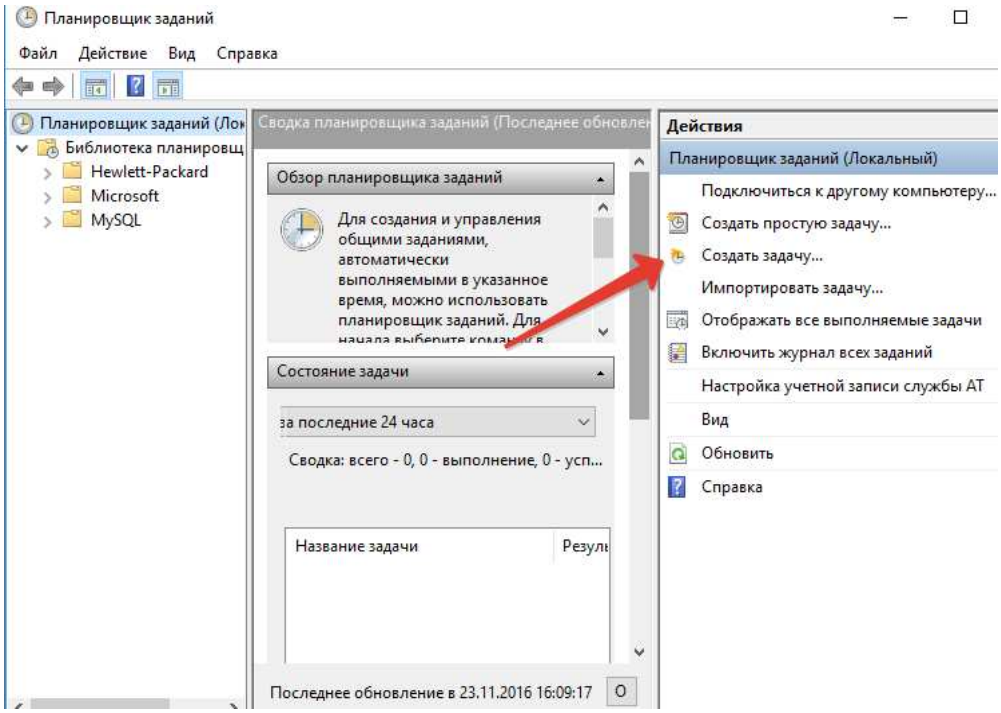
3. Если сообщения об ошибке нет, значит команда отработала. Далее необходимо проверить, появилась ли запись в ХД.

Примечание. Для корректной работы пакетной обработки необходимо, чтобы все файлы (сценарий, ХД, excel) лежали на диске D, т.к. политика безопасности диска С не всегда позволяет выполнить пакетную обработку правильно. *Важно:* При переносе с диска С на диск D убедитесь, что в самом сценарии для excel и ХД файлов указаны относительные пути, иначе при обработке файлы не будут найдены и новые записи в самом ХД не появятся.

4. Пакетное выполнение настроить на запуск по расписанию с помощью планировщика заданий, например стандартного Windows Scheduler. Такая возможность удобна для автоматического запуска процесса загрузки в хранилище данных из учетной системы в нужное время. Для этого создается ярлык для файла DStudio.exe, для которого в строке «Объект» вводится командная строка запуска Deductor Studio в пакетном режиме. Затем в Windows Scheduler настраивается время запуска этого задания.

Порядок действий:

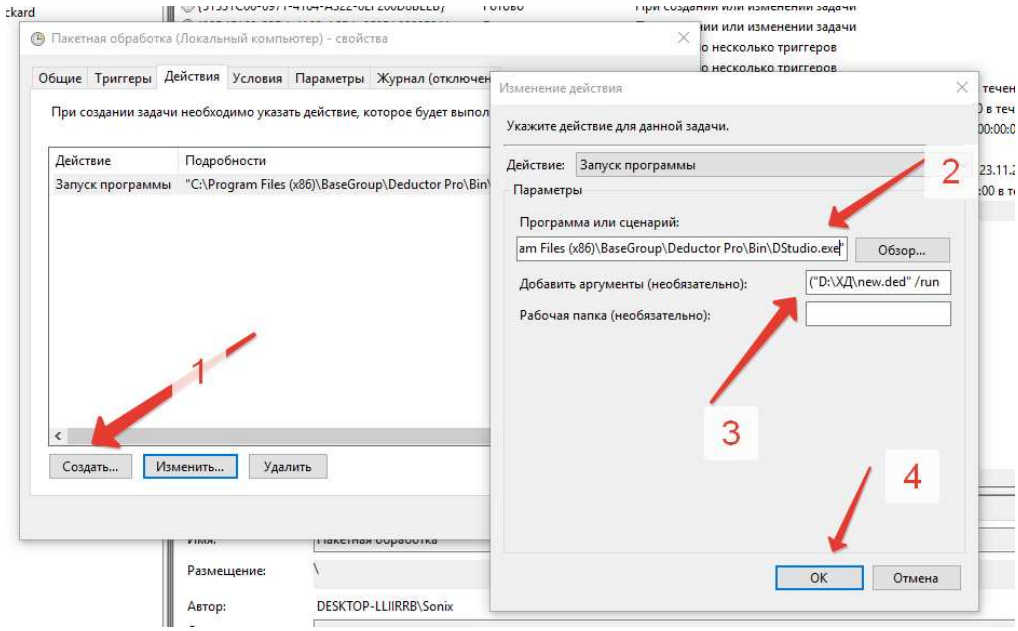
1. Открыть планировщик заданий Windows и создать задачу:



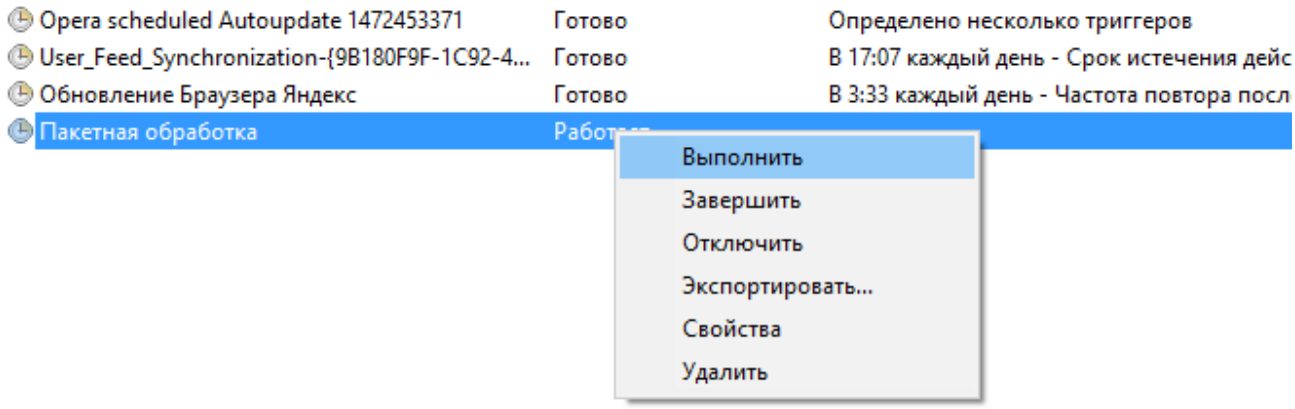
2. Ввести название задачи, перейти на вкладку действия и выбрать программу для запуска (C:\Program Files (x86)\BaseGroup\Deductor Pro\Bin\DStudio.exe);

3. В качестве аргументов передать те же самые, что и при запуске через командную строку ("D:\ХД\new.ded" /run);

4. Нажать «ок»:



5. Перейти в библиотеку планировщика, найти созданную задачу, вызвать контекстное меню и нажать «Выполнить».



Планировщик заданий необходим в том случае, когда требуется периодическая выгрузка данных в СППР, в нашем случае в АП Deductor. Данный механизм мог бы пригодиться для загрузки данных о покупках и продажах товаров в ХД.

2.2. Многомерный анализ данных и оперативная аналитическая обработка (On-line Analytical Processing) OLAP

Технологии, которые реализуют аналитическую обработку информации, ориентированы на решение стратегических вопросов компаний. Этим занимаются менеджеры всех уровней корпорации в границах той или другой СППР (Decision Support System, или DSS). Эти технологии получили название аналитического оснащения процессов обработки информации в реальном времени (On-Line Analytical Processing, или OLAP). Технологии OLAP никогда ни оперируют данными реального времени. Эти данные постоянно изменяются, поэтому строить анализ на их основе невозможно. В границах DSS, как правило, проводят сравнительный анализ, для которого требуются данные, не изменяющиеся во времени.

Технология OLAP оперирует итоговыми значениями: например, общий объем продаж за определенный период времени без учета любой специфики отдельной продажи.

OLAP – это информационная технология, которая предоставляет руководителям различного уровня возможность получения необходимой информации для принятия управленческих, финансовых и кадровых решений.

OLAP-системы построены на двух базовых принципах:

1) Все данные, необходимые для принятия решений, предварительно агрегированы на всех соответствующих уровнях и организованы так, чтобы обеспечить максимально быстрый доступ к ним;
2) Язык манипулирования данными основан на использовании бизнес-понятий.

В основе концепции OLAP лежит принцип многомерного представления данных. Первое четкое определение OLAP предложено в 1993 году Е.Ф.Коддом (E.F.Codd), он предложил двенадцать правил для оценки программных продуктов класса OLAP.

1. Многомерное концептуальное представление данных (Multi-Dimensional Conceptual View)	Концептуальное представление модели данных в продукте OLAP должно быть многомерным по своей природе, то есть позволять аналитикам выполнять интуитивные операции «анализа вдоль и поперек» (slice and dice), вращения (rotate) и размещения (pivot) направлений консолидации.
2. Прозрачность (Transparency)	Пользователь не должен знать о том, какие конкретные средства используются для хранения, обработки данных, как данные организованы, откуда берутся.
3. Доступность (Accessibility)	Аналитик должен иметь возможность выполнять анализ в рамках общей концептуальной схемы, но при этом данные могут оставаться под управлением оставшихся от старого наследства СУБД, будучи при этом привязанными к общей аналитической модели. То есть инструментарий OLAP должен накладывать свою логическую схему на физические массивы данных, выполняя все преобразования, требующиеся для обеспечения единого, согласованного и целостного взгляда пользователя на информацию.

4. Устойчивая производительность (Consistent Reporting Performance)	С увеличением числа измерений и размеров базы данных аналитики не должны столкнуться с каким бы то ни было уменьшением производительности. Устойчивая производительность необходима для поддержания простоты использования и свободы от усложнений, которые требуются для доведения OLAP до конечного пользователя.
5. Клиент-серверная архитектура (Client-Server Architecture)	Большая часть данных, требующих оперативной аналитической обработки, хранится в мэйнфреймовых системах, а извлекается с персональных компьютеров. Поэтому одним из требований является способность продуктов OLAP работать в среде клиент-сервер. Главной идеей здесь является то, что серверный компонент инструмента OLAP должен быть достаточно интеллектуальным и обладать способностью строить общую концептуальную схему на основе обобщения и консолидации различных логических и физических схем корпоративных баз данных для обеспечения эффекта прозрачности.
6. Равноправие измерений (Generic Dimensionality)	Все измерения данных должны быть равноправны. Дополнительные характеристики могут быть предоставлены отдельным измерениям, но поскольку все они симметричны, данная дополнительная функциональность может быть предоставлена любому измерению. Базовая структура данных, формулы и форматы отчетов не должны опираться на какое-то одно измерение.
7. Динамическая обработка разреженных матриц (Dynamic Sparse Matrix Handling)	Инструмент OLAP должен обеспечивать оптимальную обработку разреженных матриц. Скорость доступа должна сохраняться вне зависимости от расположения ячеек данных и быть постоянной величиной для моделей, имеющих разное число измерений и различную разреженность данных.
8. Поддержка многопользовательского ре-	Зачастую несколько аналитиков имеют необходимость работать одновременно с одной аналитической

жима (Multi-User Support)	моделью или создавать различные модели на основе одних корпоративных данных. Инструмент OLAP должен предоставлять им конкурентный доступ, обеспечивать целостность и защиту данных.
9. Неограниченная поддержка кроссмерных операций (Unrestricted Cross-dimensional Operations)	Вычисления и манипуляция данными по любому числу измерений не должны запрещать или ограничивать любые отношения между ячейками данных. Преобразования, требующие произвольного определения, должны задаваться на функционально полном формульном языке.
10. Интуитивное манипулирование данными (Intuitive Data Manipulation)	Переориентация направлений консолидации, детализация данных в колонках и строках, агрегация и другие манипуляции, свойственные структуре иерархии направлений консолидации, должны выполняться в максимально удобном, естественном и комфортном пользовательском интерфейсе.
11. Гибкий механизм генерации отчетов (Flexible Reporting)	Должны поддерживаться различные способы визуализации данных, то есть отчеты должны представляться в любой возможной ориентации.
12. Неограниченное количество измерений и уровней агрегации (Unlimited Dimensions and Aggregation Levels)	Настоятельно рекомендуется допущение в каждом серьезном OLAP инструменте как минимум пятнадцати, а лучше двадцати, измерений в аналитической модели. Более того, каждое из этих измерений должно допускать практически неограниченное количество определенных пользователем уровней агрегации по любому направлению консолидации.

В 1995 году на основе требований, изложенных Коддом, был сформулирован так называемый тест FASMI (Fast Analysis of Shared Multidimensional Information – быстрый анализ разделяемой многомерной информации), включающий следующие требования к приложениям для многомерного анализа:

– предоставление пользователю результатов анализа за приемлемое время (обычно не более 5 с), пусть даже ценой менее детального анализа;

– возможность осуществления любого логического и статистического анализа, характерного для данного приложения, и его сохранения в доступном для конечного пользователя виде;
– многопользовательский доступ к данным с поддержкой соответствующих механизмов блокировок и средств авторизованного доступа;
– многомерное концептуальное представление данных, включая полную поддержку для иерархий и множественных иерархий (это ключевое требование OLAP);
– возможность обращаться к любой нужной информации, независимо от ее объема и места хранения.

В OLAP-системах предварительно подготовленная информация преобразуется в форму многомерного куба; такими данными гораздо легче манипулировать, используя необходимые для анализа срезы (рис. 10).

Многомерный куб можно рассматривать как систему координат, осями которой являются измерения, например *Дата*, *Товар*, *Покупатель*. По осям будут откладываться значения измерений – даты, наименования товаров, названия фирм-покупателей, ФИО физических лиц и т. д.

В такой системе каждому набору значений измерений (например, дата – товар – покупатель) будет соответствовать ячейка, в которой можно разместить числовые показатели (то есть факты), связанные с данным набором.

Таким образом, между объектами бизнес-процесса и их числовыми характеристиками будет установлена однозначная связь.

Принцип организации многомерного куба поясняется на рис. 10. В серой ячейке будут располагаться факты, относящиеся к продаже цемента ЗАО «Пирамида» 6 ноября, а в черной ячейке будут располагаться факты, относящиеся к продаже плит ООО «Спецстрой» 4 ноября.

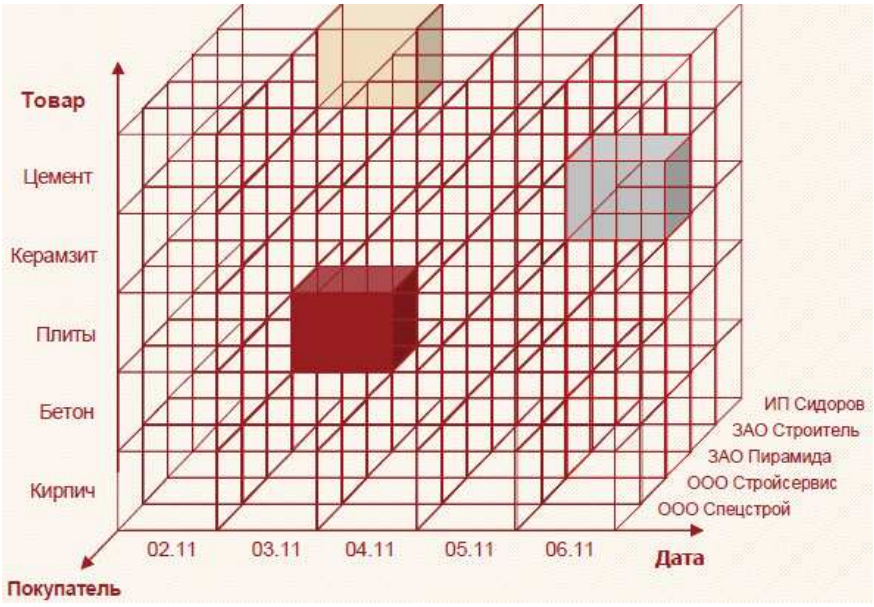


Рис. 10. Принцип организации многомерного куба

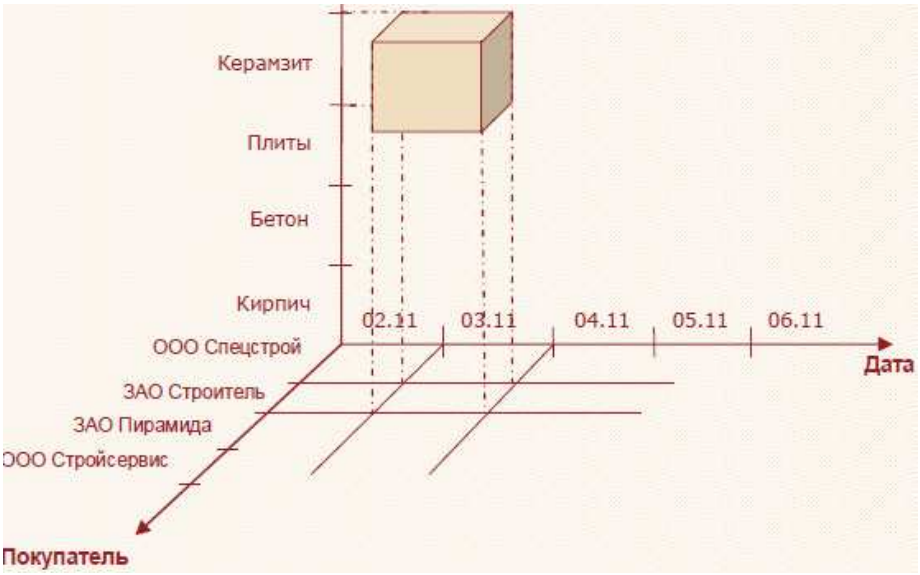


Рис. 11. Измерения и факты в многомерном кубе

Многомерный взгляд на измерения *Дата*, *Товар* и *Покупатель* представлен на рис. 11. Фактами в данном случае являются *Цена*, *Количество*, *Сумма*. Выделенный сегмент содержит информацию о том, сколько плит, на какую сумму и по какой цене приобрела фирма ЗАО «Строитель» 3 ноября.

Визуализация OLAP-куба производится с помощью специального вида таблиц, которые строятся на основе срезов OLAP-куба, содержащих необходимую пользователю информацию. Срезы являются результатом выполнения соответствующего запроса к базе данных. Как правило, в процессе построения

срезов пользователь с помощью мыши и клавиатуры манипулирует заголовками измерений, добиваясь наиболее информативного представления данных в кубе. В зависимости от положения заголовков измерений в таблице автоматически формируется запрос к базе или хранилищу данных. Запрос извлекает данные из базы или хранилища, после чего OLAP-ядро системы визуализирует их.

Общую схему работы настольной OLAP системы можно представить следующим образом:

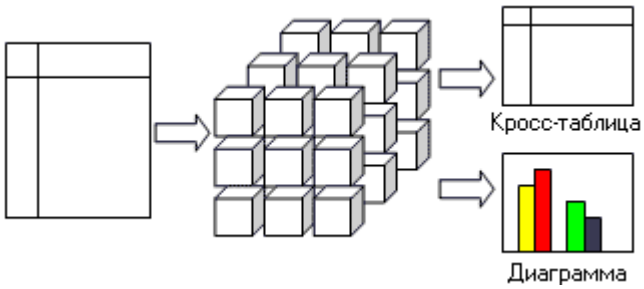


Рис. 12. Технология OLAP

Отображения, используемые в OLAP-системах, чаще всего бывают двух видов – кросс-таблицы и кросс-диаграммы. Кросс-таблица является основным и наиболее распространенным способом отображения куба. Она отличается от обычной плоской таблицы наличием нескольких уровней вложенности (например, она допускает разбиение строк на подстроки, а столбцов – на подстолбцы). Кросс-диаграмма представляет собой диаграмму заданного типа (гистограмму, линейную диаграмму и т.д.), построенную на основе кросс-таблицы. Основное отличие кросс-диаграммы от обычной диаграммы в том, что она однозначно соответствует текущему состоянию куба и при любых его изменениях (транспонирование, фильтрация по измерениям и т.д.) также синхронно изменяется.

OLAP-куб можно использовать не только как метод визуализации, но и как средство оперативного формирования отчетов и представления информации в нужном разрезе (так называемая аналитическая отчетность).

2.3. OLAP-отчеты в АП Deductor Studio 5.3

Для построения аналитической отчетности в АП Deductor предназначена вкладка **Отчеты**, способ открытия: «Вид – Отчеты», или кнопка  , после нажатия на которую в рабочей части экрана появится панель **Отчеты**.

Отчеты строятся в виде древовидного иерархического списка (рис. 13), каждым узлом которого является отдельный отчет или папка, содержащая несколько отчетов. Каждый узел дерева отчетности связан со своим узлом в дереве сценария. Для каждого отчета настраивается свой способ отображения (таблица, гистограмма, кросс-таблица, кросс-диаграмма и т.п.). Это удобно, так как несколько отчетов могут быть связаны с одним узлом дерева сценария.

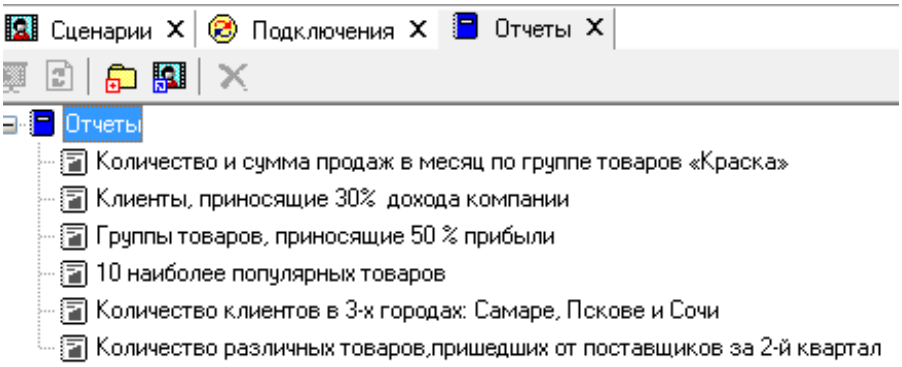


Рис. 13. Панель отчетов по данным ХД «Материалы»

Чтобы добавить новый отчет, нужно щелкнуть по кнопке **Добавить узел** или выбрать соответствующую команду из контекстного меню. В результате откроется окно «Выбор узла», в котором следует выделить узел дерева сценария, где содержится нужная выборка данных, и щелкнуть по кнопке **Выбрать.**

Следует отметить, что операция добавления нового отчета доступна, только если выделена папка или корневой пункт **Отчеты** списка отчетов. Если выделить узел, содержащий отдельный отчет, команда создания нового отчета будет недоступна.

Чтобы добавить новую папку, нужно щелкнуть по кнопке **Добавить папку**  или выбрать соответствующую команду в контекстном меню. В результате в списке отчетов появится новая папка с открытым полем имени, куда следует ввести имя папки. После ввода имени для его сохранения щелкнуть по

любому узлу списка. Чтобы поместить отчет в папку, нужно перед вызовом команды **Добавить узел** выделить эту папку.

Таким образом, было построено 6 OLAP-отчетов на основе подготовленных (преобразованных) данных ХД «Материалы»:

Для построения первого OLAP-отчета необходимо привести данные к нужному виду. Для этого был использован обработчик «Дата/Время» для агрегации данных в разрезе месяца. С помощью обработчика «Группировка» данные были сгруппированны по полю «Дата» и «Группы товаров», а обработчиком «Фильтр» были выбраны необходимые данные из группы товаров «Краска».

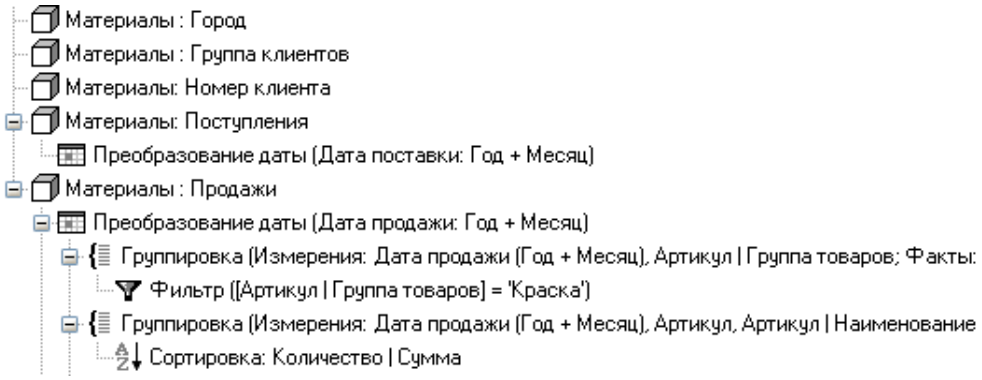


Рис. 14. Преобразование данных для OLAP-анализа

Первый OLAP-отчет построен для определения количества продаж и суммы продаж помесячно по группе товаров «Краска». Он построен на основании измерений *Дата продажи* и *Группа товаров*, а также фактов «Количество» и «Сумма с учетом скидки»:

Таблица × Куб ×

	Артикул | Группа товаров			
	Краска		Итого:	
Дата продажи (Год + Месяц)	Σ Количество	Σ Сумма с учетом скидки	Σ Количество	Σ Сумма с учетом скидки
01.03.2004	23 083,00	21 361 356,81	23 083,00	21 361 356,81
01.04.2004	15 055,00	12 670 427,67	15 055,00	12 670 427,67
01.05.2004	19 107,00	17 455 930,21	19 107,00	17 455 930,21
01.06.2004	17 976,00	16 135 497,39	17 976,00	16 135 497,39
01.07.2004	28 103,00	24 842 715,88	28 103,00	24 842 715,88
01.08.2004	33 436,00	27 191 120,73	33 436,00	27 191 120,73
01.09.2004	35 831,00	32 628 371,47	35 831,00	32 628 371,47
01.10.2004	31 566,00	27 404 441,91	31 566,00	27 404 441,91
01.11.2004	28 690,00	25 592 482,67	28 690,00	25 592 482,67
01.12.2004	258,00	195 922,38	258,00	195 922,38
Итого:	233 105,00	205 478 267,10	233 105,00	205 478 267,10

Рис. 15. OLAP-отчет «Количество и сумма продаж в месяц по группе товаров «Краска»

По данному отчету можно сделать вывод, что в сентябре было продано максимальное количество товаров группы «Краски» на максимальную сумму. Также данный отчет можно представить в виде диаграммы, на которой наглядно представлено, когда и какое максимальное количество товаров было продано за год:

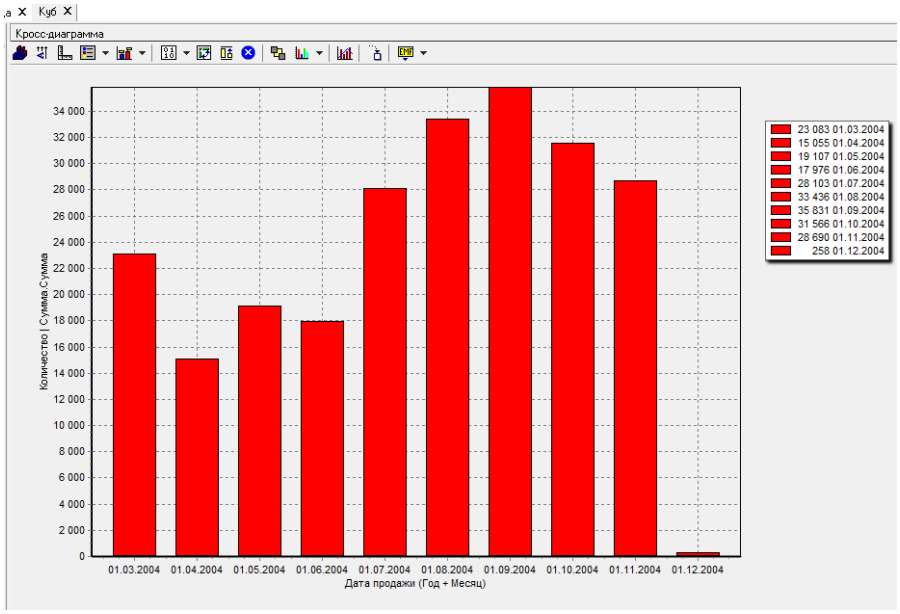


Рис. 16. OLAP-отчет в виде диаграммы «Количество продаж в месяц по группе товаров «Краска»

Второй OLAP-отчет построен по измерениям «Номер клиента» и «Группа товаров», а также по факту «Сумма с учетом скидки», в отчете отображены суммы проданных товаров каждым клиентом:

Таблица | Куб

	Номер клиента | Группа клиента			
Номер клиента	VIP клиент	Клиент	Постоянный клиент	**Итого:**
00000003		7 654 066,02		7 654 066,02
00000004		22 829,94		22 829,94
00000010			4 540 088,82	4 540 088,82
00000011		130 504,28		130 504,28
00000014		27 234,51		27 234,51
00000015			9 080 035,66	9 080 035,66
00000016		4 183 675,25		4 183 675,25
00000017		13 671,30		13 671,30
00000019	41 078 054,30			41 078 054,30
00000020	53 799 967,08			53 799 967,08
00000031		185 644,20		185 644,20
00000032		238 598,13		238 598,13

Рис. 17. OLAP-отчет «Суммы проданных товаров каждым клиентом»

Чтобы увидеть не все суммы проданных товаров, а только те группы товаров, клиенты которых приносят в общем 30% дохода компании, нужно в самом визуализаторе «OLAP-куб» воспользоваться фильтром. Настройки данного фильтра представлены на рис. 18:

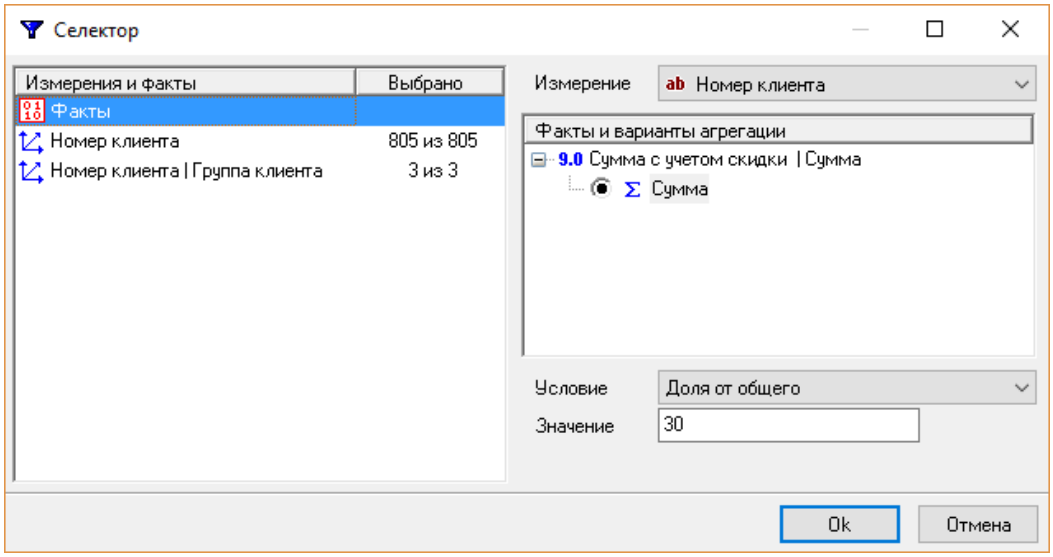


Рис. 18. Настройка фильтра для OLAP-отчета «Клиенты, приносящие 30 % дохода компании»

Таблица | Куб

	Номер клиента | Группа клиента			
Номер клиента	VIP клиент	Клиент	Постоянный клиент	**Итого:**
00000019	41 078 054,30			41 078 054,30
00000020	53 799 967,08			53 799 967,08
00000069			30 060 766,02	30 060 766,02
00001315	32 035 539,67			32 035 539,67
00003495		28 879 423,04		28 879 423,04
00011533			47 771 820,10	47 771 820,10
99999999	200 049 195,71			200 049 195,71
EL001176		29 710 072,18		29 710 072,18
Итого:	326 962 756,76	58 589 495,22	77 832 586,12	463 384 838,10

Рис. 19. OLAP-отчет «Клиенты, приносящие 30 % дохода компании»

Отчет показывает, что представленные клиенты приносят 30 % дохода компании, такие клиенты очень важны для компании.

Следующий OLAP-отчет построен по измерениям «Артикул» и «Наименование товара», а также по факту «Количество», т.е. в отчете будет отображена статистика проданных товаров. Для начала преобразуем данные: с помощью

обработчика «Группировка» сгруппируем данные по артикулу и наименованию товара (количество будет фактом). Затем отсортируем количество по убыванию. По настроенным данным сформируем OLAP-куб. Воспользовавшись фильтром, выберем первые 10 элементов:

Таблица | Куб

Артикул | Наименование товара ▾ Артикул	Σ Количество | Сумма
Болт 10x20 цинк, шестигранная головка, 8 штук, 8 категория, уп	41 926,00
Болт 8x100 цинк, шестигранная головка, 2 штуки, 4 категория, уп	46 545,00
Болт 8x25 цинк, шестигранная головка, 8 штук, 6 категория, уп	43 033,00
Болт 8x40 цинк, шестигранная головка, 8 штук, 8 категория, уп	41 351,00
Болт 8x50 цинк, шестигранная головка, 6 штук, 7 категория, уп	42 034,00
Болт 8x70 цинк, шестигранная головка, 6 штук, 9 категория, уп	44 005,00
Болт 8x90 цинк, шестигранная головка, 4 штук, 4 категория, уп	44 023,00
Дюбель с шурупом забивной WKRET-MET 6x60 грибок SMK, 200 штук, уп	43 798,00
Дюбель с шурупом забивной WKRET-MET 6x80 грибок SMK, 100 штук, уп	43 833,00
Шуруп универсальный 4.5x40, 50 штук, 9 категория, уп	44 486,00
Итого:	435 034,00

Рис. 20. OLAP-отчет «10 наиболее популярных товаров»

Отчет показывает, какие 10 товаров являются наиболее популярными, т.е. более продаваемыми.

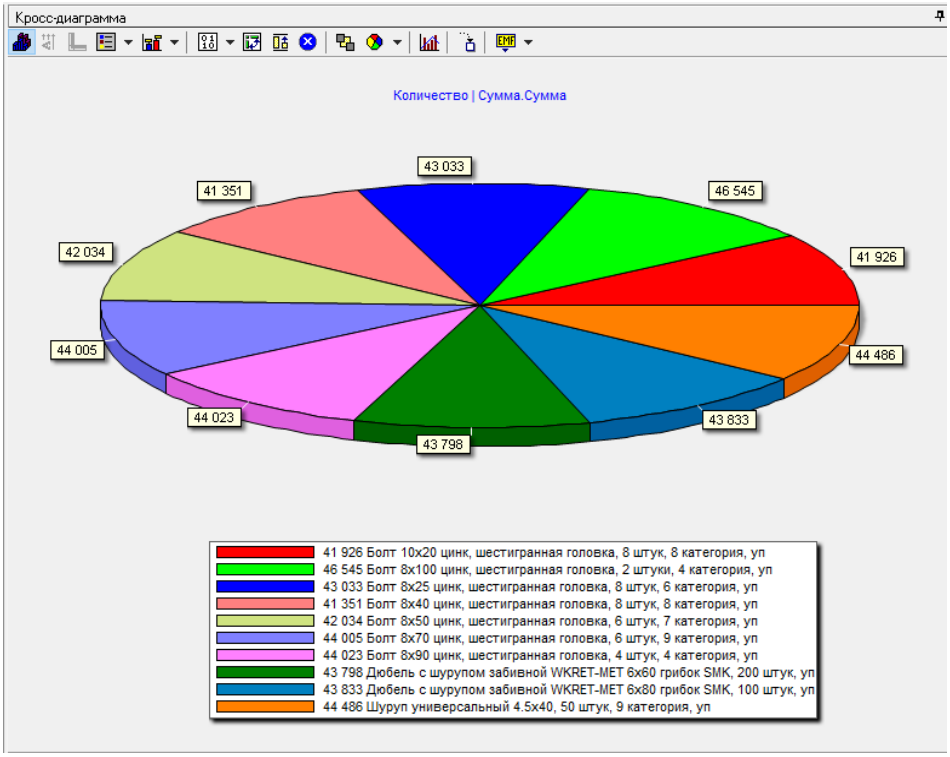


Рис. 21. Кросс-диаграмма OLAP-отчета «10 наиболее популярных товаров»

Следующий OLAP-отчет построен по измерениям «Группа клиента» и «Город», а также по факту «Номер клиента», т.е. в отчете будет отображена статистика по количеству клиентов в каждом городе с детализацией по группе клиента. При настройке фактов указываем, что факт «Номер клиента» будет иметь агрегацию «Количество уникальных», таким образом будет считаться правильное количество клиентов (без повторений):

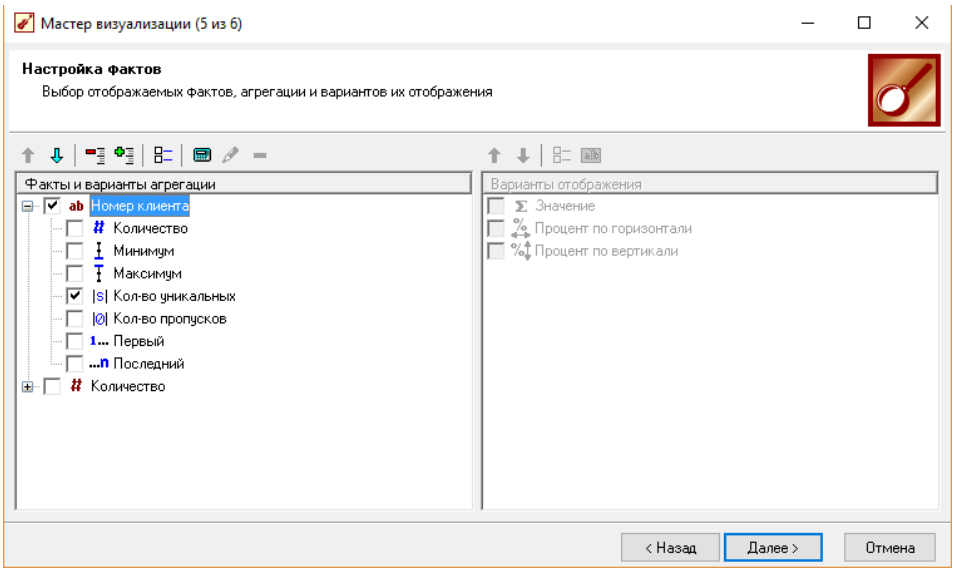


Рис. 22. Настройка OLAP-отчета «Количество клиентов в каждом городе»

Таблица | Куб

	Группа клиента			
Город	VIP клиент	Клиент	Постоянный клиент	**Итого:**
Архангельск		6		6
Астрахань		5		5
Балашиха		6		6
Барнаул		17		17
Белгород		6		6
Великие Луки		5		5
Владивосток		6		6
Владимир	1	16		17
Волгоград		12		12
Вологда		18		18
Воронеж	1	5		6
Екатеринбург		12		12
Зеленоград	1	5		6

Рис. 23. OLAP-отчет «Количество клиентов в каждом городе»

Можно настроить фильтр по определенным городам и посмотреть, сколько в выбранных городах клиентов. Например, посмотрим, сколько покупателей есть в Пскове, Самаре и Сочи:

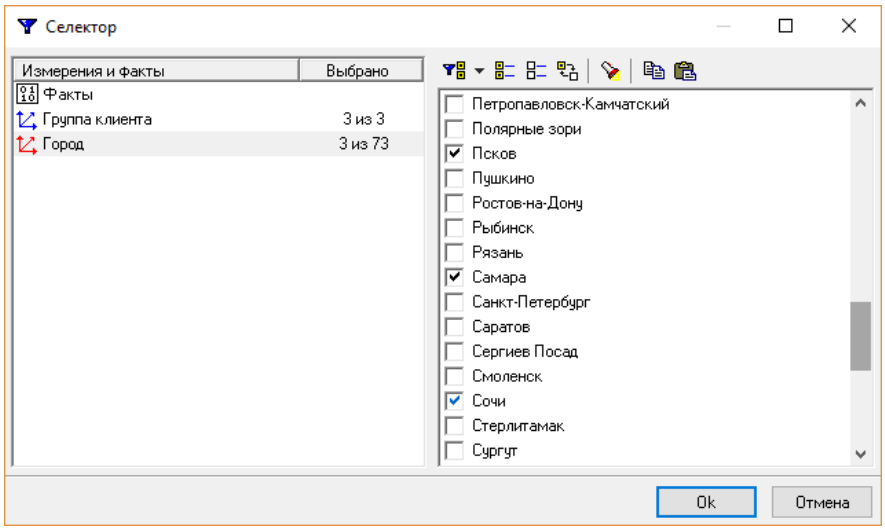


Рис. 24. Выбор городов

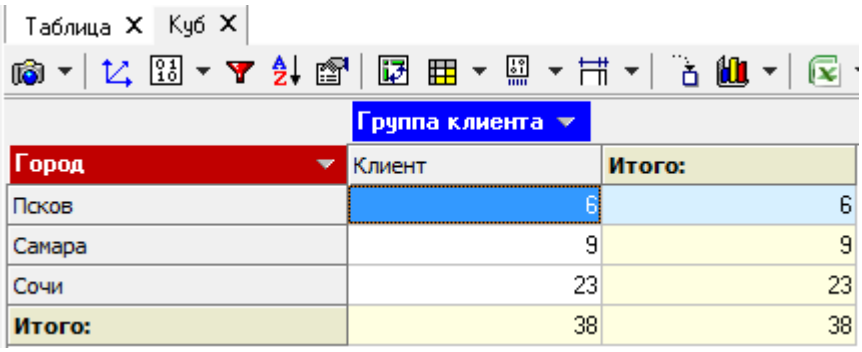


Рис. 25. OLAP-отчет «Количество клиентов в городах»

По данному отчету можно сделать вывод, что в Пскове есть 6 клиентов, в Самаре – 9, а в Сочи – 23, причем все они относятся к группе клиента «Клиент».

Последний OLAP-отчет построен по измерениям «Номер поставщика» и «Дата поставки», а также по факту «Наименование товара». Дата поставки преобразована с помощью обработки «Дата/время», т.е. в отчете будет отображена статистика по номеру поставщика и дате поставки помесячно, а на пересечении столбцов и строк будет указано количество уникальных товаров. Построим OLAP-отчет за 2-й квартал по поставщикам:

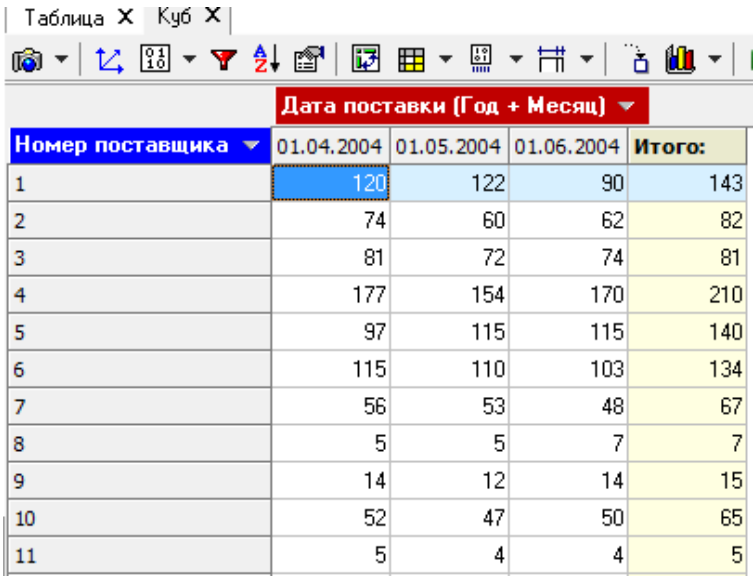


Рис. 26. OLAP-отчет «Отчет за второй квартал по уникальным товарам»

Данный отчет показывает, сколько различных товаров 28 поставщиков продали компании за квартал.

Вопросы для самопроверки

1. Каковы объекты хранилища Deductor Warehouse?
2. Какие типы данных могут быть у объектов хранилища Deductor Warehouse?
3. Чем отличается атрибут процесса от измерения?
4. Как должна выглядеть структура таблицы-справочника, если имеются иерархии?
5. Что такое *Редактор ХД* в Deductor Studio?
6. Как создать новое пустое ХД?
7. Как сделать иерархию измерений?
8. Какие срезы для измерений типа дата/время предусмотрены в Deductor Warehouse?
9. Что такое статический и пользовательский фильтр?
10. Пусть при данной структуре ХД предполагается, что уникальность точки в пространстве определяется совокупностью измерений Дата + Код региона. Что можно предпринять в случае, когда уникальность точки в многомерном пространстве этими измерениями не обеспечивается, других измерений нет, и при-

этом в хранилище нужно сохранить исходную детализацию данных? Как это сделать в Deductor Studio?

11. В чем заключается OLAP-анализ и каковы его цели?

12. Какова структура OLAP-куба?

13. Какие манипуляции с измерениями можно производить, чтобы сделать представление куба более информативным?

14. В чем заключаются операции транспонирования и детализации, каковы их цели?

15. Что такое кросс-диаграмма и для каких целей она используется?

Задания для самостоятельной работы

Задание 1. Создание ХД в Deductor Studio 5.3

1. Для создания нового хранилища данных или подключения к существующему нужно перейти на закладку **Подключения** и запустить **Мастер подключений.**

2. Пройти первые два шага, выбрав тип приемника (источника) Deductor Warehouse и тип базы данных Firebird.

3. На третьем шаге нужно задать параметры базы данных, в которой будет создана физическая и логическая структура хранилища данных:

- база данных – **farma.gdb**;
- логин – **sysdba**, пароль – **masterkey**;
- установить флажок **сохранять пароль**.

4. На следующем шаге выбирается версия для работы с ХД Deductor Warehouse

5. На пятом шаге при нажатии кнопки **Создать файл базы данных с необходимой структурой метаданных** по указанному ранее пути будет создан файл **farma.gdb** (появится сообщение об успешном создании). Это и есть пустое хранилище данных, готовое к работе.

6. На последних двух шагах осталось выбрать визуализатор для подключения (здесь это Сведения и Метаданные) и задать для нового хранилища имя **Farma**, метку **Фармация** и описание **«Хранилище данных с информацией о прода-**

жах в аптечной сети компании Фарма-Х».

6. После нажатия на кнопку **Готово** на дереве узлов подключений появится метка хранилища.

7. Для проверки доступа к новому ХД воспользуйтесь кнопкой  . Если спустя некоторое время появится сообщение «Тестирование соединения прошло успешно», то хранилище готово к работе. Иначе нужно внести изменения в параметры подключения ХД, используя кнопку  .

8. Сохраните настройки подключений, нажав на кнопку  .

9. Для проектирования структуры ХД вызвать **Редактор метаданных** кнопкой  на вкладке **Подключения**.

10. В открывшемся окне редактора, встав на узле **Измерения**, при помощи кнопки **Добавить** добавьте в метаданные первое измерение *Код группы* со следующими параметрами:

- Имя – GR_ID;
- Метка – *Группа.Код*;
- Тип данных – целый.

11. Проделать аналогичные действия для создания всех остальных измерений, взяв параметры из таблицы:

Измерение	**Имя**	**Метка**	**Тип данных**
Код группы	GR_ID	Группа.Код	целый
Код товара	TV_ID	Товар.Код	целый
Код отдела	PART_ID	Отдел.Код	целый
Дата	S_DATE	Дата	дата/время
Час покупки	S_HOUR	Час	целый

12. К каждому измерению, кроме *ДАТА* и *ЧАС*, нужно добавить по текстовому атрибуту. Для измерения *Группа.Код* это будет *Группа.Наименование* GR_NAME, для измерения *Товар.Код* это будет *Товар.Наименование* TV_NAME, для измерения *Отдел.Код* это будет *Отдел.Наименование* PART_NAME.

13. Путем простого добавления (ссылка на измерение отображается иконкой) установить ссылку измерения *Товар.Код* на измерение *Группа.Код*.

14. В окне редактора, встав на узле **Процессы**, при помощи кнопки **Добавить** введите в метаданные следующие параметры: Имя – SALES; Метка – *Продажи*.

15. Добавьте в процесс ссылки на 4 существующих измерения: *Дата*, *Отдел.Код*, *Товар.Код*,*Час* и добавьте ссылки на два факта: целочисленный – *Количество* F_COUNT и вещественный – *Сумма* F_SUM.

16. Проектирование структуры и метаданных ХД закончено, можно закрыть окно редактора.

Задание 2. Загрузка информации в ХД

I. С помощью **Мастер импорта** последовательно импортируйте 4 текстовых файла в Deductor в следующей последовательности:

- groups.txt – товарные группы;
- produces.txt – товары;
- stores.txt – отделы;
- sales.txt – продажи товаров по дням.

Замечание. В сценарии загрузки используйте относительные пути к текстовым файлам. При выборе файла для импорта используйте относительный путь, это означает, что файл должен находиться в той же папке, что и файл со сценарием.

II. При загрузке данных в хранилище сначала загружаются таблицы измерений со своими атрибутами, и только после этого загружается таблица процесса sales.txt.

1. Для загрузки первого измерения *Группа.Код*, встав на первом узле сценария, вызовите **Мастер экспорта**, из списка приемников нужно выбрать Deductor Warehouse, далее из списка доступных хранилищ укажите ХД Фармация. На следующей вкладке требуется указать, в какое именно измерение будет загружаться информация *Группа.Код*. Последнее, что осталось, это установить соответствие элементов объекта в ХД с полями вход-

ного источника данных (т.е. таблицы groups.txt). Нажатие кнопки **Пуск** загружает в измерение данные. При этом старые данные, если они были, будут обновлены.

2. Проделайте аналогичные действия еще для двух измерений – *Отдел.Код*, *Товар.Код*.
3. Загрузите данные в процесс *Продажи*, повторив сначала действия, аналогичные перечисленным выше. В отличие от загрузки измерений здесь появляются два специфических шага. На одном нужно задать параметры для контроля непротиворечивости информации в ХД – указать измерения, по которым следует удалять данные из хранилища (в нашем примере нужно поставить флажок по полю *Дата*). На последней странице настроек оставьте параметры по умолчанию.
4. Сохраните файл сценария под именем load.ded в той же папке, где находятся текстовые файлы таблиц.

Задание 3. Извлечение информации из ХД

1. Импортируйте данные из процесса *Продажи* за последние три месяца.

1) С помощью **Мастера импорта** выберите тип источника данных Deductor Warehouse, на следующем шаге – ХД *Фармация*, а затем процесс – *Продажи*.
2) Определите, какие измерения и атрибуты из выбранного на предыдущем шаге процесса должны быть импортированы (поставьте флажки у следующих полей: *Дата*, *Отдел.Наименование*, *Товар.Наименование*, *Группа.Наименование*, *Час*).
3) Определите импортируемые факты (*Количество* и *Сумма*) и виды их агрегаций (в большинстве случаев требуется агрегация в виде суммы).
4) Для измерения *Дата* определите срез «Все продажи за последние три месяца от текущей даты» (нужно выделить это измерение, выбрать условие «последний» и значение «3 месяца от имеющихся данных»).

5) На следующем шаге настройте динамический фильтр по полю *Дата* (установите флажок).
6) Для результирующего набора данных определите способ его отображения в виде таблицы.

2. Импортируйте информацию о продажах из ХД, включая атрибуты товара. Установите следующие срезы:

- «кроме последнего периода, 1 месяц от имеющихся данных»;
- срез только по одной товарной группе.

3. Создайте новый сценарий farma.ded.

1) Создайте узлы импорта из ХД Фармация:

a) Список товарных групп.

b) Продажи по дням в Аптеке 1 по всем товарным группам, за исключением иммуномодуляторов за последние 10 недель от имеющихся данных.

2) Скопируйте последний узел и настройте в нем динамический фильтр на наименование отдела.

Задание 4. Требуется разработать систему аналитической отчетности в Deductor. Система отчетности строится на основе хранилища данных **Фармация.** Все требуемые отчеты должны быть вынесены на **Панель отчетов**.

1. Диаграммы и гистограммы

1) Постройте отчет – гистограмму распределения средних цен всех товаров, которые продавались за последние 5 месяцев от имеющихся данных. Назовите отчет «Гистограмма средних цен».
2) Постройте временной ряд продаж по месяцам, используя все имеющиеся данные (по оси ОУ откладывается сумма продаж).

2. OLAP-кубы

1. Построить куб по трем измерениям (отдел, месяц года, товарная группа), в ячейках которого отображается сумма и объем (количество проданных единиц продукции) продаж за все периоды, имеющиеся в ХД.

2. То же, что в п.1, но за последние три месяца от имеющихся данных.
3. Сформировать многомерный отчет и график загруженности торговых точек по дням месяца.
4. Сформировать многомерный отчет и график загруженности торговых точек по дням недели.
5. 10 самых продаваемых товаров.
6. 5 самых популярных товаров в каждой товарной группе.
7. 10 самых продаваемых товаров с 18 до 21 часа.
8. 10 самых продаваемых товаров по пятницам.
9. 5 самых популярных товаров в товарной группе «Местные анестетики»
10. Товары, дающие 80 % объема продаж в летние месяцы.
11. Пять товаров, пользующихся наибольшим спросом по понедельникам до 12 часов дня.

3. Интеллектуальные информационные системы (ИИС)

Интеллектуальная информационная система (ИИС) – это ИС, которая основана на концепции использования базы знаний и моделей Data Mining для генерации алгоритмов решения экономических задач различных классов в зависимости от конкретных информационных потребностей пользователей.

Data Mining (DM) – обнаружение в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. К базовым методам интеллектуального анализа данных прежде всего относят нейронные сети, деревья решений, логистическую регрессию, ассоциативные правила.

Информация, найденная в процессе применения методов Data Mining, должна быть нетривиальной и ранее неизвестной, например знания должны

описывать новые связи между свойствами, предсказывать значения одних признаков на основе других.

Характерные признаки интеллектуальных информационных систем:

- *развитые коммуникативные способности*, которые характеризуют естественный способ взаимодействия (интерфейса) конечного пользователя с системой;
- *способность к самообучению;*
- *умение решать сложные плохо формализуемые задачи*, т.е. задачи, которые требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, для которой могут быть характерны неопределенность и динамичность исходных данных и знаний.

Условно каждому из признаков соответствует свой класс ИИС:

- системы с интеллектуальным интерфейсом;
- самообучающиеся системы;
- экспертные системы.

Исторически именно экспертные системы были первыми СППР, которые привлекли внимание потребителей.

Система с интеллектуальным интерфейсом – это ИИС, предназначенная для поиска неявной информации в базе данных или тексте, для произвольных запросов, составляемых, как правило, на ограниченном естественном языке

Интеллектуальные БД отличаются от обычных БД возможностью выборки по запросу необходимой информации, которая может явно не храниться, а выводиться из имеющейся в базе данных информации. Примерами таких запросов могут быть следующие: «Вывести список товаров, цена которых выше среднеотраслевой»; «Вывести список товаров-заменителей некоторой продукции»; «Вывести список потенциальных покупателей некоторого товара».

В запросе требуется осуществить поиск по условию, которое должно быть доопределено в ходе решения задачи. Формулирование запроса осуществ-

ляется в диалоге с пользователем, последовательность шагов которого выполняется в максимально удобной для пользователя форме. Запрос к базе данных может формулироваться и с помощью *естественно-языкового* интерфейса.

Естественно-языковой интерфейс предполагает трансляцию естественно-языковых конструкций на внутримашинный уровень представления знаний.

Естественно-языковый интерфейс используется для:

- доступа к интеллектуальным базам данных;
- контекстного поиска документальной текстовой информации;
- голосового ввода команд в системах управления;
- машинного перевода с иностранных языков.

Гипертекстовые системы предназначены для реализации поиска по ключевым словам в базах текстовой информации. Механизм поиска работает прежде всего с базой знаний ключевых слов, а уже затем непосредственно с текстом.

Системы контекстной помощи можно рассматривать как частный случай интеллектуальных гипертекстовых и естественно-языковых систем. В системах контекстной помощи пользователь описывает проблему (ситуацию), а система с помощью дополнительного диалога ее конкретизирует и сама выполняет поиск относящихся к ситуации рекомендаций. Такие системы относятся к классу систем распространения знаний (Knowledge Publishing) и создаются как приложение к системам документации (например, технической документации по эксплуатации товаров).

Системы когнитивной графики позволяют осуществлять интерфейс ИИС с помощью графических образов, которые генерируются в соответствии с происходящими событиями.

Самообучающаяся система – это ИИС, которая на основе примеров реальной практики автоматически формирует единицы знаний.

В основе самообучающихся систем лежат методы автоматической классификации примеров ситуаций реальной практики (обучения на примерах).

Примеры реальных ситуаций накапливаются за некоторый исторический период и составляют обучающую выборку. Эти примеры описываются множеством признаков классификации. Причем обучающая выборка может быть:

– «с учителем», когда для каждого примера задается в явном виде значение признака его принадлежности некоторому классу ситуаций;

– «без учителя», когда по степени близости значений признаков классификации система сама выделяет классы ситуаций.

В результате обучения системы автоматически строятся обобщенные правила или функции, определяющие принадлежность ситуаций классам, которыми обученная система пользуется при интерпретации новых возникающих ситуаций. Таким образом, автоматически формируется база знаний, используемая при решении задач классификации и прогнозирования. Эта база знаний периодически автоматически корректируется по мере накопления опыта реальных ситуаций, что позволяет сократить затраты на ее обновление.

3.1. Модели Data Mining. Применение метода деревьев решений для оценки и выбора управленческих решений

Деревья решений (decision trees) относятся к числу самых популярных и мощных инструментов Data Mining, позволяющих эффективно решать задачи классификации (например, отнесение региона к определенному классу, типу, виду), задачи регрессии и прогнозирования основных экономических, социальных, экологических показателей: ВРП, объема промышленного производства, уровня доходов бюджета, населения и других.

Первые идеи создания деревьев решений восходят к работам Ховленда (Hoveland) и Ханта (Hunt) конца 50-х годов XX века. Основополагающей работой, давшей импульс для развития этого направления, явилась книга Ханта (Hunt, E.B.), Мэрина (Marin J.) и Стоуна (Stone, P.J) «Experiments in Induction», вышедшая в 1966 г.

Область применения деревьев решений в настоящее время широка, но все решаемые задачи могут быть объединены в следующие три класса:

- **Описание данных:** Деревья решений позволяют хранить информацию о данных в компактной форме, вместо них мы можем хранить дерево решений, которое содержит точное описание объектов.
- **Классификация:** Деревья решений отлично справляются с задачами классификации, т.е. отнесения объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.
- **Регрессия:** Если целевая переменная имеет непрерывные значения, деревья решений позволяют установить зависимость целевой переменной от независимых (входных) переменных. Например, к этому классу относятся задачи численного прогнозирования (предсказания значений целевой переменной).

Деревья решений – иерархические древовидные структуры, состоящие из решающих правил вида «если – то», которые могут быть сформулированы на естественном языке. Поэтому деревья решений являются наиболее наглядными и легко интерпретируемыми моделями.

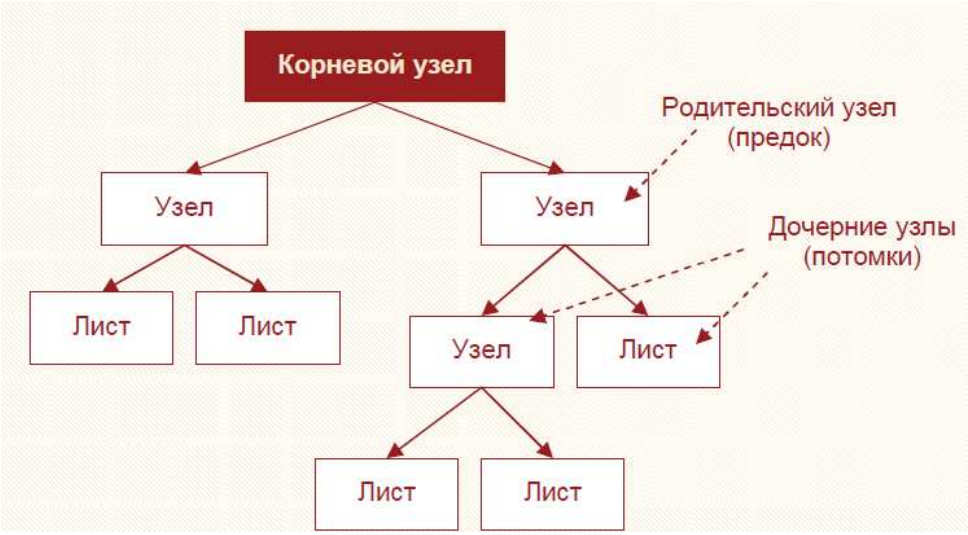


Рис. 27. Структура *Дерева решений*

Каждый узел дерева имеет следующие характеристики:

– количество примеров, попавших в узел;

– доли примеров, относящихся к каждому из классов;

– число классифицированных примеров (для узлов);

– процент записей, верно классифицированных данным узлом.

Особый интерес для оценки качества классификации имеют два показателя:

– ***поддержка*** (support) – отношение числа правильно классифицированных примеров в данном узле или листе к общему числу попавших в него примеров: $S = \frac{N_{\text{кл}}}{N_{\text{общ}}} \cdot 100\%$. Значение поддержки находится в интервале [0;100 %].

– ***достоверность*** (confidence) – отношение числа правильно классифицированных примеров к числу неправильно классифицированных: $S = \frac{N_{\text{кл}}}{N_{\text{ошиб ка}}} \cdot 100\%$.

Чем больше число правильно классифицированных примеров в узле, тем выше достоверность. Поддержка и достоверность могут использоваться в качестве параметров построения дерева решений. Например, можно задать, что разбиение должно производиться до тех пор, пока в узле не будет достигнут заданный порог поддержки.

В отличие от методов, использующих статистический подход, деревья решений основаны на машинном обучении и в большинстве случаев не требуют предположений о статистическом распределении значений признаков. Для определения меры эффективности деревьев решений используют тестовое множество – набор примеров, которые ранее не использовались при построении дерева решений. Пропуская набор тестовых примеров через построенное дерево решений, вычисляем, для какого процента примеров класс был определен правильно. Это позволяет оценить качество всего классификатора и качество решения задачи классификации отдельных ветвей в дереве.

Для эффективного построения дерева решений должны выполняться следующие условия:

1) Описание атрибутов – анализируемые данные должны быть представлены в виде структурированного набора, в котором вся информация об объекте или наблюдении должна быть выражена совокупностью атрибутов (признаков, описывающих классифицируемые объекты).

2) Предварительное определение классов – категории, к которым относятся наблюдения (метки классов), должны быть заданы предварительно, то есть имеет место обучение с учителем.

3) Различимость классов – должна обеспечиваться принципиальная возможность установления факта принадлежности или непринадлежности примера к определенному классу. При этом количество примеров должно быть намного больше, чем количество классов.

4) Полнота данных – обучающее множество должно содержать достаточно большое количество различных примеров. Необходимая численность зависит от таких факторов, как количество признаков и классов, сложность классификационной модели.

В основе работы деревьев решений лежит процесс рекурсивного разбиения исходного множества наблюдений или объектов на подмножества, которые ассоциированы с классами. Алгоритм построения дерева следующий:

дерево строится «сверху вниз» от корня. Начинается процесс с определения, какой атрибут следует выбрать для проверки в корне дерева. Для этого каждый атрибут исследуется на предмет, как хорошо он в одиночку классифицирует набор данных (разделяет на классы по целевому атрибуту). Когда атрибут выбран, для каждого его значения создается ветка дерева, набор данных разделяется в соответствии со значением к каждой ветке, процесс повторяется рекурсивно для каждой ветки. Также следует проверять критерий остановки.

Более структурно алгоритм можно представить следующим образом:

1. Если $\max(Gain(X,A)) < \Theta$, создать лист с меткой преобладающего класса.

2. Если не осталось атрибутов для разбиения, создать лист с меткой преобладающего класса.
3. Иначе:
    A. выбрать атрибут, соответствующий максимуму $Gain(X,A)$;
    B. создать ветку для каждого значения атрибута;
    C. для каждой ветки:
        I. построить подмножество X_a, исключив при этом атрибут A из множества атрибутов;
        II. рекурсивно вызвать алгоритм для подмножества X_a.

Если атрибут количественный (например, вещественное число), то разбиение по нему выполняется в форме теста $A \leq a_0$ и формируются две ветки (истина и ложь). Существует способ оптимального подбора a_0 также с использованием идеи прироста информации.

Рис. 28. Пример разбиения по непрерывному атрибуту

Алгоритмы, реализующие деревья решений: ID3, C4.5, CART

1. **ID3 (**Iterative Dichotomizer**).** В основе этого алгоритма лежит понятие информационной энтропии, то есть меры неопределенности информации. Для того чтобы определить следующий атрибут, необходимо подсчитать энтропию всех неиспользованных признаков относительно тестовых об-

разцов и выбрать тот, для которого энтропия минимальна. Этот атрибут и будет считаться наиболее целесообразным признаком классификации.

2. **C4.5.** Этот алгоритм – усовершенствование предыдущего метода, позволяющее, в частности, «усекать» ветви дерева, если оно слишком сильно «разрастается», а также работать не только с атрибутами-категориями, но и с числовыми.
3. **CART.** Алгоритм разработан в целях построения бинарных деревьев решений, то есть тех деревьев, каждый узел которых при разбиении «дает» только двух потомков. Грубо говоря, алгоритм действует путем разделения на каждом шаге множества примеров ровно напополам – по одной ветви идут те примеры, в которых правило выполняется (правый потомок), по другой – те, в которых правило не выполняется (левый потомок). Таким образом, в процессе «роста» на каждом узле дерева алгоритм проводит перебор всех атрибутов и выбирает для следующего разбиения тот, который максимизирует значение показателя, вычисляемого по математической формуле и зависящего от отношений числа примеров в правом и левом потомке к общему числу примеров.

В алгоритмах ID3 и C4.5 выбор атрибута осуществляется на основе теоретико-информационного подхода: $Gain(X)=Info(T)-Info_X(T)$, (1),

где $Info(T)$ – энтропия множества T, $Info_x(T) = \sum_i Info(T_i) \cdot \ln \frac{|T_i|}{|T|}$ (2)

y – зависимая переменная; y = {C_1, C_2…$C_к$} – ее значения; T = $C_1 \cup C_2 \cup \ldots \cup C_k$ – объединение К подмножеств, соответствующих различным классам.

Среднее количество информации, необходимое для определения одного класса, определяется в битах по формуле энтропии: $H(x) = -\sum p(x_i) * \log_2 p(x_i)$.

$Info(T) = -\sum_{i=1}^{K} \frac{|Ci|}{|T|} * \log_2 \frac{|Ci|}{|T|}$, где |T| и $|C_i|$ – мощности всей выборки и отдельных классов.

Множества T_1, T_2,...T_n получены при разбиении исходного множества T по проверке атрибута X. Выбирается атрибут, дающий максимальное значение по критерию (1).

На практике в результате работы этих алгоритмов часто получаются слишком детализированные деревья, которые при их дальнейшем применении делают много ошибок. Это связано с явлением переобучения. К тому же ветвистое дерево, имеющее много узлов, разбивает обучающее множество на все большее количество подмножеств, состоящих из все меньшего количества объектов. Ценность правила, справедливого, скажем, для 2 – 3 объектов, крайне низка, и в целях анализа данных такое правило практически непригодно. Гораздо предпочтительнее иметь дерево, состоящее из малого количества узлов, которым бы соответствовало большое количество объектов из обучающей выборки.

Для сокращения деревьев используется «отсечение ветвей» (pruning).

Качество классификационной модели, построенной при помощи дерева решений, характеризуется двумя основными признаками: точностью распознавания и ошибкой. Точность распознавания рассчитывается как отношение объектов, правильно классифицированных в процессе обучения, к общему количеству объектов набора данных, которые принимали участие в обучении. Ошибка рассчитывается как отношение объектов, неправильно классифицированных в процессе обучения, к общему количеству объектов набора данных, которые принимали участие в обучении. Отсечение ветвей или замену некоторых ветвей поддеревом следует проводить там, где эта процедура не приводит к возрастанию ошибки.

Для извлечения правил необходимо исследовать все пути от корня до каждого листа дерева. Каждый такой путь даст правило, где условиями будут являться проверки из узлов, встретившихся на пути.

Пример. Чтобы решить задачу, т.е. принять решение, играть ли в гольф, следует отнести текущую ситуацию к одному из известных классов (в данном

случае «играть» или «не играть»). Для этого требуется ответить на ряд вопросов, которые находятся в узлах этого дерева, начиная с его корня.

Исходные данные:

№	Наблюдение	Температура	Влажность	Ветер	Игра
1	солнце	жарко	высокая	нет	нет
2	солнце	жарко	высокая	есть	нет
3	облачно	жарко	высокая	нет	да
4	дождь	норма	высокая	нет	да
5	дождь	холодно	норма	нет	да
6	дождь	холодно	норма	есть	нет
7	облачно	холодно	норма	есть	да
8	солнце	норма	высокая	нет	нет
9	солнце	холодно	норма	нет	да
10	дождь	норма	норма	нет	да
11	солнце	норма	норма	есть	да
12	облачно	норма	высокая	есть	да
13	облачно	жарко	норма	нет	да
14	дождь	норма	высокая	есть	нет

Применение алгоритма ID3 для данного примера:

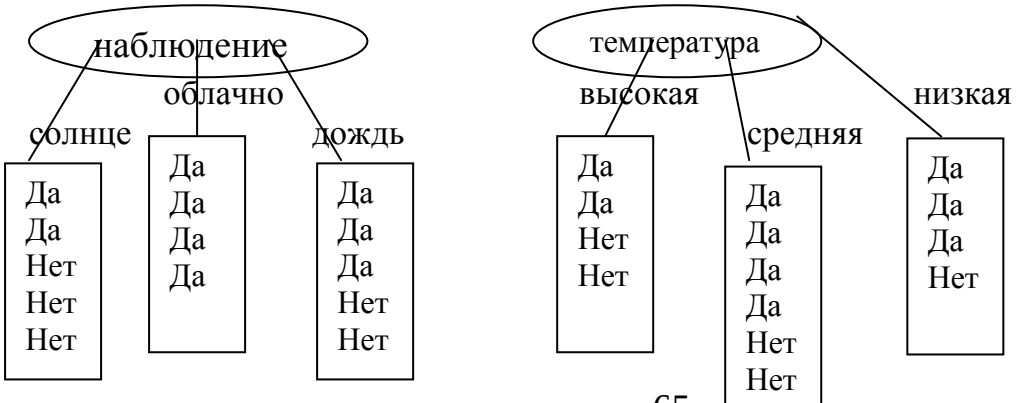
В данном примере 9 записей относятся к классу C_1, а пять – к классу C_2. Энтропия множества Info(T) = $-9/14 \log_2(9/14) - 5/14 \log_2(5/14) = 0.940$ бит.

Критерий для выбора атрибута – максимальное значение прироста информации по формуле: $Gain(X)=Info(T)-Info_X(T)$.

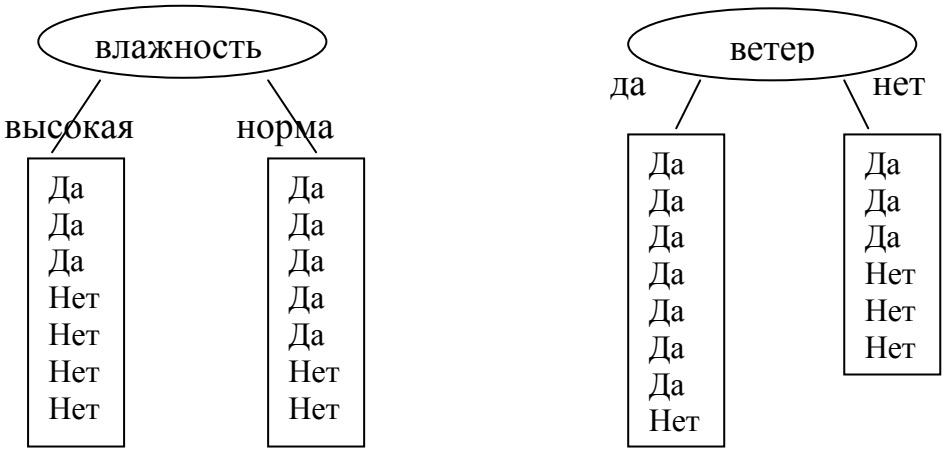
В случае, когда все классы равновероятны, условно полагаем, что $\log_2(0)=0$.

Вычислим энтропию для атрибута *«наблюдение»* в случае разбиения исходного множества на три подмножества:

$Info_{наблюдение} = (5/14)*0.971 + (4/14)*0 + (5/14)*0.971 = 0.693$ бит



Таким образом, прирост информации при использовании атрибута *«наблюдение»* для разбиения исходного подмножества составит: *Gain* (наблюдение)=0.247 бит.

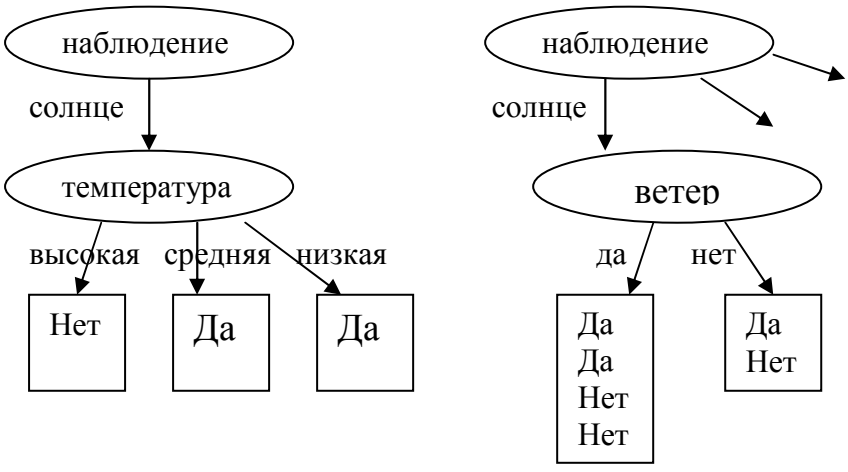


Считаем для всех независимых переменных:

Gain (температура) = 0.029 бит; *Gain* (влажность) = 0.152 бит;

Gain (ветер) = 0.048 бит.

Выбирается переменная с максимальным значением Gain. В данном примере этот атрибут *наблюдение*. Затем производится дальнейшее построение дерева.



Для $T_{солнечно}$: *Gain* (температура) = 0.571 бит; *Gain* (ветер)=0.020 бит;

Gain (влажность) = 0.971 бит.

Таким образом, следующая переменная – влажность.

Пример построения модели отклика получателей рассылки на активных и неактивных при помощи алгоритма построения дерева решений в аналитической платформе Deductor Studio Academic.

Торговая компания, осуществляющая продажу товаров, располагает информацией о своих клиентах и их покупках. Компания провела рекламную рассылку 13 504 клиентам и получила отклик в 14,5 % случаев. Необходимо построить модели отклика и проанализировать результаты, чтобы предложить способы минимизации издержек на новые почтовые рассылки.

Данные находятся в файлах *responses1.txt* (обучающее множество), и *responses2.txt* (тестовое множество) представляет собой таблицу со следующими полями:

Поля наборов данных «Отклики» Таблица 11

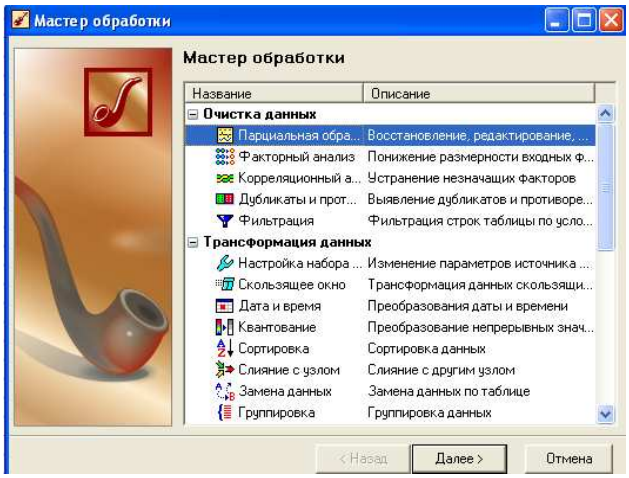
N	Поле	Описание	Тип
1	Код клиента	Уникальный идентификатор	целый
2	Пол	Пол клиента	строковый
3	Сколько лет клиенту	Число лет с момента первой покупки. Если менее года, то в поле стоит 0	целый
4	Кол-во позиций товаров	Сколько уникальных товаров приобретал клиент	целый
5	Доход с клиента, тыс. ед.	Суммарная стоимость всех заказов клиента	вещест.
6	Число покупок в тек. году	Сколько раз клиент делал заказ в текущем году	целый
7	Обращений в службу поддержки	Сколько раз клиент обращался в службу поддержки	целый
8	Задержки платежей	Задержки клиента фиксируются, когда длительное время после заказа оплата не поступает	целый
9	Дисконтная карта	Является ли клиент участником дисконтных программ, дающих право на скидки	целый
10	Возраст	Возраст клиента	целый
11	Отклик	Отклик клиента на последнюю рассылку. Значение «1» означает, что клиент совершил покупку после прямой адресной рассылки	целый
12	Дата отклика	Информационное поле (пустое, если отклика не было)	дата

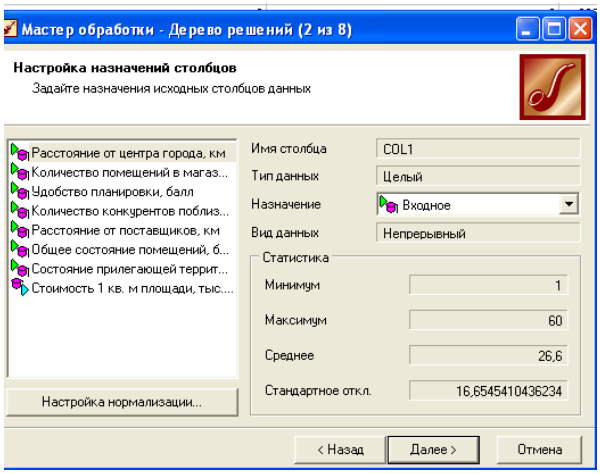
1) Построена матрица корреляции для оценки влияния входных переменных на выходную

Рис. 29. Матрица корреляции для оценки влияния входных переменных на выходную

Анализ матрицы корреляции показал, что наибольшее влияние на выходную переменную оказывает фактор – *Число покупок в тек. году*, однако стоит также обратить внимание на два других фактора – *Количество позиций товаров* и *Доход с клиента*.

2) Рассмотрим решение задачи классификации с применением модели **Деревья решений**. Если выходной столбец имеет непрерывный тип, необходимо с помощью обработчика «Квантование» сделать его дискретным. Следуя по шагам, зададим параметры обработчика **Деревья решений** (рис. 30).





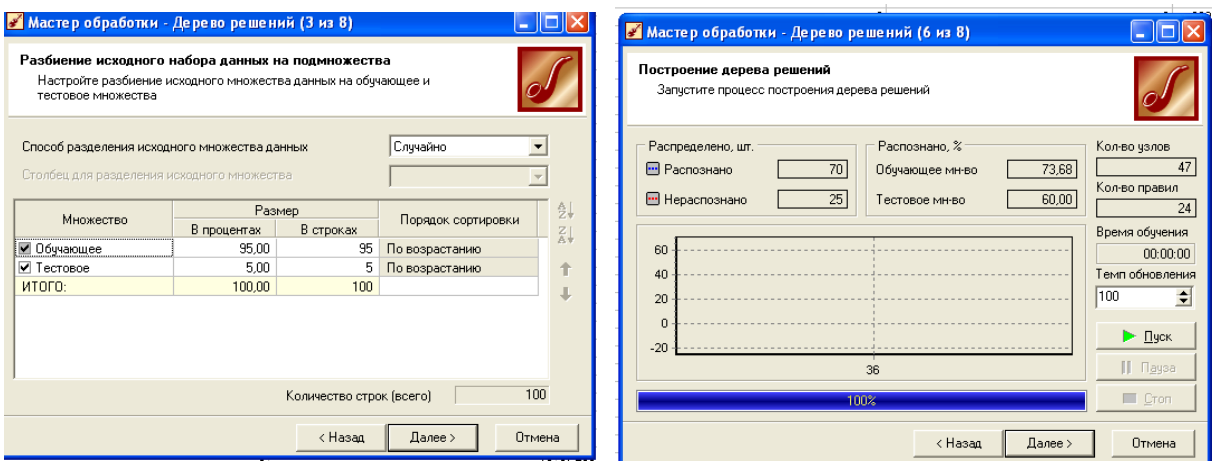


Рис. 30. Шаги настройки модели «Деревья решений»

Определим столбец «Отклик» как выходное поле, разделим выборку на обучающее и тестовое множества. В результате получим дерево решений (рис. 31), демонстрирующее зависимость выходного поля от входных факторов.

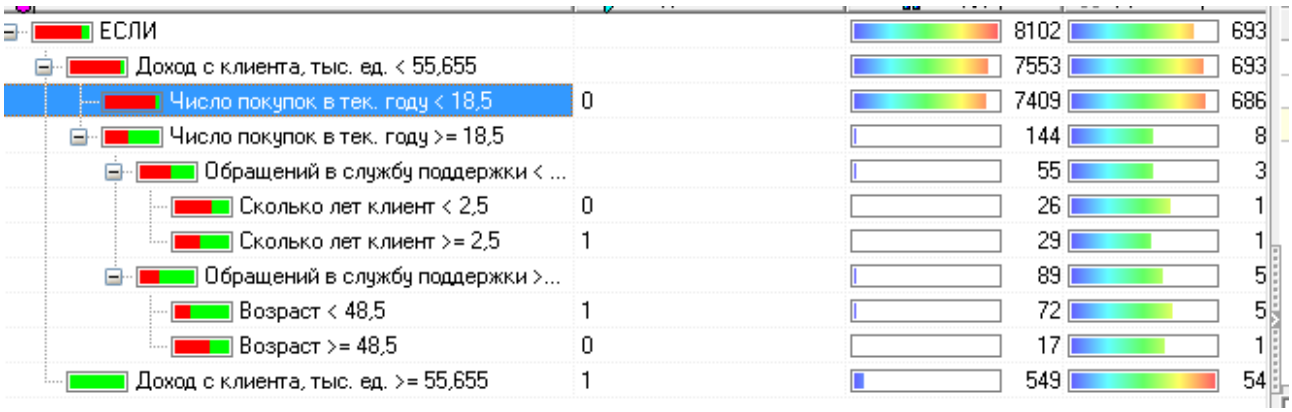


Рис. 31. Дерево решений

3) Проанализируем визуализаторы «Дерево решения», «Правила», «Значимость атрибутов», «Таблица сопряженности».

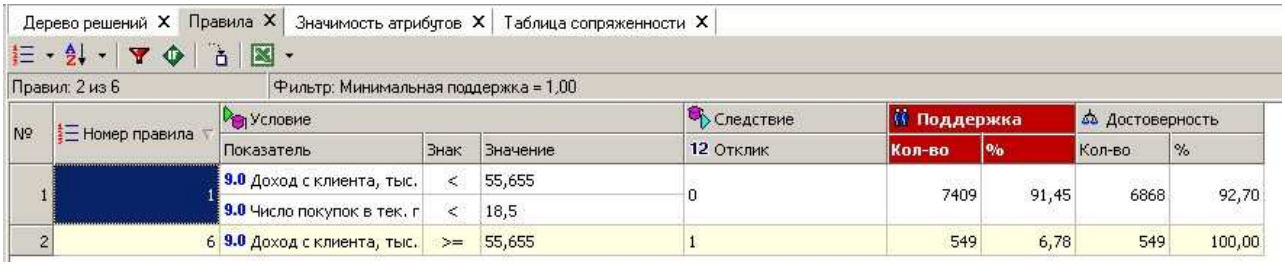
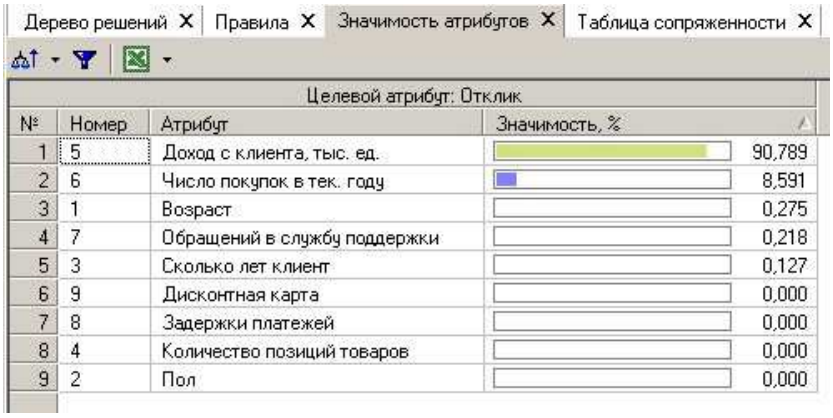


Рис. 32. Визуализатор «Правила»

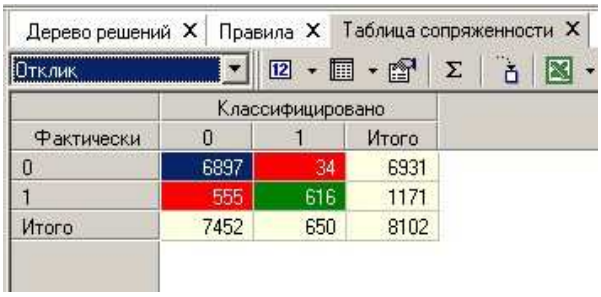
Для данной модели с уровнем доверия в 20 % можно выписать следующие правила:

- ЕСЛИ *Доход с клиента* < 55,655 и *Число покупок в тек. году* меньше 18,5, ТО Клиент не откликается с достоверностью в 92,7%.
- ЕСЛИ *Доход с клиента* >= 55,655, ТО клиент откликается с достоверностью в 100 % (всегда).



№	Номер	Атрибут	Значимость, %
1	5	Доход с клиента, тыс. ед.	90,789
2	6	Число покупок в тек. году	8,591
3	1	Возраст	0,275
4	7	Обращений в службу поддержки	0,218
5	3	Сколько лет клиент	0,127
6	9	Дисконтная карта	0,000
7	8	Задержки платежей	0,000
8	4	Количество позиций товаров	0,000
9	2	Пол	0,000

Рис. 33. Визуализатор «Значимость атрибутов»



	Классифицировано		
Фактически	0	1	Итого
0	6897	34	6931
1	555	616	1171
Итого	7452	650	8102

Рис. 34. Визуализатор «Таблица сопряженности»

Таблица сопряженности является визуализатором, позволяющим оценить качество классификации данных. Она позволяет наиболее наглядно оценить результаты классификации, полученные с помощью той или иной модели. Таблица сопряженности показывает результаты сравнения категориальных значений выходного поля исходной (обучающей) выборки и категориальных значений выходного поля, рассчитанных с помощью модели.

В таблице сопряженности ячейки с числом правильно распознанных примеров отображаются в зеленых ячейках, а неправильно распознанных – в красных. Итоги (общее количество примеров) – серым. Чем большее число примеров попали в зеленные ячейки, тем лучше результаты классификации. Кроме этого, в таблице сопряженности хорошо видно, по каким значениям выходного поля было допущено наибольшее число ошибок классификации.

Рис. 35. Диаграмма визуализатора «Таблица сопряженности»

Для нашего примера при уровне доверия в 20 % модель совершает ошибку в 7,27 % случаев. Это меньше, чем 10 %, т.е. модель приемлема.

4) Изменяя порог отсечения, построим новые модели, чтобы выбрать модель, лучшую с точки зрения точности и интерпретации. Выпишем наиболее значимые правила.

При понижении уровня доверия до 5 % значимые правила остаются те же. Модель совершает ошибку в 7,31 %, что меньше, чем 10 %, однако ошибка незначительно возросла, по сравнению с моделью с уровнем доверия в 20 %.

Рис. 36. Диаграмма визуализатора «Таблица сопряженности»

5) При повышении уровня доверия до 50 % число правил повышается до 13. Значимые правила будут следующие:

- ЕСЛИ *Доход с клиента* < 55,655 и *число покупок в тек. году* < 0,5, ТО клиент не откликается с достоверностью в 100 %.

- ЕСЛИ *Доход с клиента* < 55,655 И 0,5 <=*число покупок в тек. году* < 6,5, ТО клиент не откликается с достоверностью в 81,43 %.
- ЕСЛИ *Доход с клиента* >= 55,655, ТО клиент откликается с достоверностью в 100 %.

Модель совершает ошибку 7,13 %, что незначительно меньше, чем модель с уровнем доверия в 20 %.

6) Так как у третьей модели с уровнем доверия в 50 % самый низкий процент ошибки, то был сделан вывод, что из текущих моделей именно она является наиболее пригодной и эффективной.

7) Построим дерево решений на сбалансированном обучающем множестве и посмотрим те же визуализаторы.

Для построения сбалансированного множества нужно применить те обработчики, которые представлены на рисунке ниже:

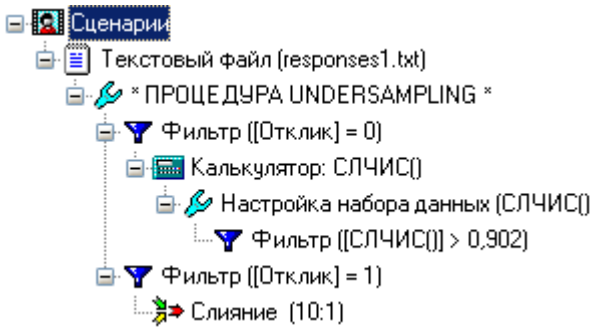


Рис. 37. Сценарий «Случайное удаление примеров мажоритарного класса (Random Undersampling) в соотношении 1:10»

8) В результате получилось следующее дерево решений:

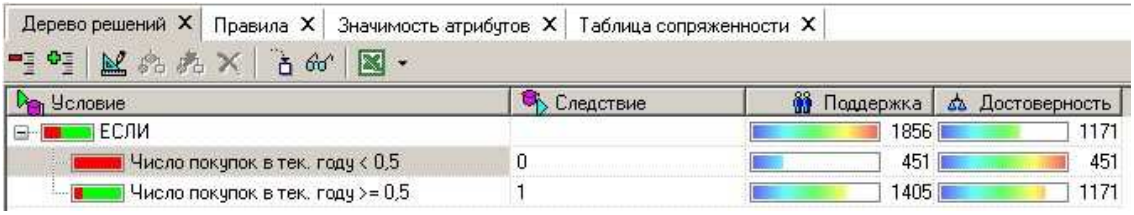


Рис. 38. Дерево решений для сбалансированного обучающего множества

Получены следующие правила:

- ЕСЛИ *число покупок в тек. году* < *0,5*, ТО клиент не откликается с достоверностью в 100 % (всегда).

- ЕСЛИ *число покупок в тек. году* >= 0,5, ТО клиент откликается с достоверностью в 83,35 %.

Рис. 39. Диаграмма визуализатора «Таблица сопряженности» для модели дерево решений на сбалансированном обучающем множестве

Для последней модели значение ошибки 12,44 %, что больше, чем 10 %, а значит, модель не приемлема.

9) Построим интерактивное дерево решений на сбалансированной выборке, приняв во внимание пожелания экспертов:

- Первым атрибутом должен быть «*Сколько лет клиент*».
- Вторым атрибутом – «*Доход с клиента*». Всех клиентов нужно разбить на 3 категории: малоприбыльные (до 20 тыс. ед.), дающие умеренный (от 20 до 50 тыс. ед.) и высокий доход (свыше 50 тыс. ед.).

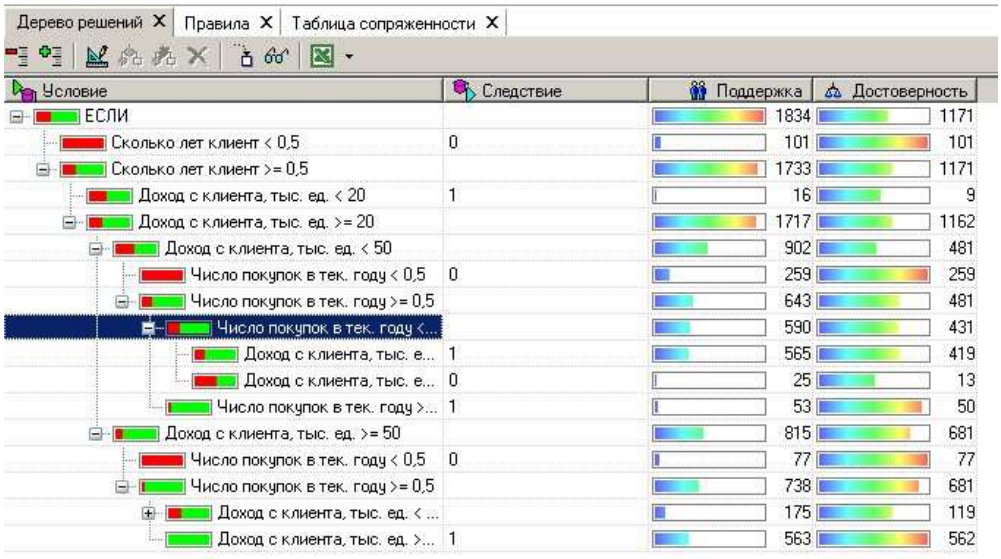


Рис. 40. Дерево решений, построенное в интерактивном режиме

10) При изучении правил были выделены наиболее значимые:

- ЕСЛИ *Сколько лет клиент* >= 0,5 и 20<=*Доход с клиента* <= 50 и 0,5 <=*Число покупок в тек. году* < =19,5, ТО клиент откликается с достоверностью 74,16 %.
- ЕСЛИ *Сколько лет клиент* >= 0,5 И 50<=*Доход с клиента* < 54,945 и *Число покупок в тек. году* >= 0,5, ТО клиент откликается с достоверностью 71,52 %.
- ЕСЛИ *Сколько лет клиент* >= 0,5 И 20<=*Доход с клиента* < 50 и *Число покупок в тек. году* >= 19,5 , ТО клиент откликается с достоверностью 94,34 %.
- ЕСЛИ *Сколько лет клиент* >= 0,5 И *Доход с клиента* >= 55,4 и *Число покупок в тек. году* >= 0,5 , ТО клиент откликается с достоверностью 99,82 %.

Модель совершает ошибку в 12,71 %, что больше 10 %, а значит, модель не является приемлемой.

11) Нужно прогнать через лучшую модель тестовое множество и сделать выводы о качестве классификации.

В результате сравнения ошибок различных моделей была выбрана модель дерева решений с доверием в 50 % на несбалансированном множестве с ошибкой в 7,13 %. Применим её к тестовому множеству с помощью обработчика Скрипт.

Выпишем наиболее значимые правила:

- ЕСЛИ *Доход с клиента* < 55,655 и *Число покупок в тек. году* < 0,5, ТО клиент не откликается с достоверностью 100 % (никогда).
- ЕСЛИ *Доход с клиента* < 55,655 и 0,5 <=*Число покупок в тек. году* < 6,5, ТО клиент не откликается с достоверностью 81,43 %.
- ЕСЛИ *Доход с клиента* >= 55,655, ТО клиент откликается с достоверностью 100 % (всегда).

Модель совершает ошибку в 7,63 %, что меньше 10 %, а значит, модель является приемлемой.

Рис. 41. Диаграмма визуализатора «Таблица сопряженности» для модели дерево решений на тестовом множестве

3.2. Применение ассоциативных правил для оценки и выбора управленческих решений

Ассоциативные правила – установление закономерностей между связанными событиями.

Впервые эта задача была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому ее еще называют анализом рыночной корзины (market basket analysis).

Ассоциация имеет место в том случае, если несколько событий связаны друг с другом. Например, исследование, проведенное в компьютерном супермаркете, может показать, что 55 % купивших компьютер берут также и принтер или сканер, а при наличии скидки за такой комплект принтер приобретают в 80 % случаев. Располагая сведениями о подобной ассоциации, менеджерам легко оценить, насколько действенна предоставляемая скидка.

Если существует цепочка связанных во времени событий, то говорят о последовательности. Так, например, после покупки дома в 45 % случаев в течение месяца приобретается и новая кухонная плита, а в пределах двух недель 60 % новоселов обзаводятся холодильником.

Примеры применения ассоциативных правил:

- ✓ выявление наборов товаров, которые в супермаркетах часто покупаются вместе или никогда не покупаются вместе;
- ✓ определение доли клиентов, положительно относящихся к нововведениям в их обслуживании;
- ✓ определение профиля посетителей веб-ресурса;
- ✓ определение доли случаев, в которых новое лекарство оказывает опасный побочный эффект.

Пусть имеется база данных, состоящая из покупательских транзакций. Каждая *транзакция* – это набор товаров, купленных покупателем за один визит. *Предметный набор* – это непустое множество предметов, появившихся в одной транзакции.

Целью анализа является установление следующих зависимостей: если в транзакции встретился некоторый набор элементов X, то на основании этого можно сделать вывод о том, что другой набор элементов Y также должен появиться в этой транзакции. Установление таких зависимостей дает возможность находить очень простые и понятные правила.

Ассоциативное правило состоит из двух наборов предметов: *условие (antecedent)* и *следствие (consequent).*

«Если условие, то следствие»; $X \rightarrow Y$; «Из X следует Y».

Условие и следствие часто называются соответственно левосторонним (left-hand side – LHS) и правосторонним (right-hand side – RHS) компонентами ассоциативного правила.

Объективными показателями значимости ассоциативных таких правил являются ***поддержка*** и ***достоверность***.

Поддержка – количество или процент транзакций, содержащих как условие, так и следствие. Правило $X \rightarrow Y$ имеет *поддержку* S (support), если:

$$S(X \rightarrow Y) = \frac{количество\ транзакций,\ содержащих\ X\ и\ Y}{общее\ количество\ транзакций} \cdot 100\% .$$

Достоверность ассоциативного правила (confidence) C представляет собой меру точности правила и определяется, как отношение количества транзакций, содержащих условие и следствие, к количеству транзакций, содержащих только следствие:

$$C(X \to Y) = \frac{\textit{количество транзакций, содержащих } X \textit{ и } Y}{\textit{общее количество транзакций, содержащих } X} \cdot 100\%.$$

Пример. Пусть 75 % транзакций, содержащих хлеб, также содержат молоко, а 3 % от общего числа всех транзакций содержат оба товара, тогда 75 % – это достоверность правила, а 3 % – это поддержка.

Если поддержка и достоверность достаточно высоки, можно с большой вероятностью утверждать, что любая будущая транзакция, которая включает условие, будет также содержать и следствие.

Сильные правила – правила, для которых значения поддержки и достоверности превышают заданный порог.

Алгоритмы поиска ассоциативных правил предназначены для нахождения всех правил вида X → Y, причем поддержка и достоверность этих правил должны находиться в рамках некоторых наперед заданных границ, называемых соответственно минимальной и максимальной поддержкой и минимальной и максимальной достоверностью.

Границы значений параметров поддержки и достоверности выбираются таким образом, чтобы ограничить количество найденных правил. Если поддержка имеет большое значение, то алгоритмы будут находить правила, хорошо известные аналитикам или настолько очевидные, что нет никакого смысла проводить такой анализ. С другой стороны, низкое значение поддержки ведет к генерации огромного количества часто статистически необоснованных правил. Тем не менее большинство интересных правил находится именно при низком значении порога поддержки. Еще одним параметром, ограничивающим количество найденных правил, является максимальная мощность часто встречающихся множеств. Если этот параметр указан, то при поиске правил будут рассмат-

риваться только множества, количество элементов которых будет не больше данного параметра k, следовательно, любое найденное правило будет состоять не больше чем из *k* элементов.

Субъективные показатели значимости ассоциативных правил: *Лифт, Левередж, Улучшение* также могут использоваться для последующего ограничения набора рассматриваемых ассоциаций путем установки порога значимости, ниже которого ассоциации отбрасываются.

Лифт *(lift)* – это отношение частоты появления условия в транзакциях, которые также содержат и следствие, к частоте появления следствия в целом:

$L(A \rightarrow B) = C(A \rightarrow B) / S(B)$.

Лифт является обобщенной мерой связи двух предметных наборов: при значениях лифта > 1 связь положительная, при 1 она отсутствует, а при значениях < 1 – отрицательная. Значения лифта большие, чем единица, показывают, что условие чаще появляется в транзакциях, содержащих следствие, чем в остальных.

Левередж *(leverage)* – это разность между наблюдаемой частотой, с которой условие и следствие появляются совместно (то есть поддержкой ассоциации), и произведением частот появления (поддержек) условия и следствия по отдельности: $T(A \rightarrow B) = S(A \rightarrow B) - S(A)S(B)$.

Если Левередж≈0, то правило не значимо.

Улучшение (improvement) показывает, полезнее ли правило случайного угадывания. $I(X \rightarrow Y) = \frac{S(X \rightarrow Y)}{S(X) \cdot S(Y)}$.

Если $I(X \rightarrow Y) > 1$, это значит, что вероятнее предсказать наличие набора B с помощью правила, чем угадать случайно.

Поиск ассоциативных правил *алгоритм Apriori* (1994 г.)

Методика поиска:

1. Следует найти частые наборы.

Частый предметный набор – предметный набор с поддержкой больше заданного порога либо равной ему. Этот порог называется минимальной поддержкой.

При поиске частых наборов используется свойство антимонотонности: Если предметный набор Z не является частым, то добавление некоторого нового предмета A к набору Z не делает его более частым. Другими словами, если Z не является частым набором, то и набор $Z \cup A$ также не будет являться таковым.

Данное полезное свойство позволяет значительно уменьшить пространство поиска ассоциативных правил.

2. На основе найденных наборов необходимо сгенерировать ассоциативные правила, удовлетворяющие условиям минимальной поддержки и достоверности.

– Генерируются все возможные поднаборы s.
– Если поднабор ss является непустым поднабором s, то рассматривается ассоциативное правило R: $ss \rightarrow (s - ss)$, где $s - ss$ представляет собой набор s без поднабора ss. R будет считаться ассоциативным правилом, если будет удовлетворять условию заданного минимума поддержки и достоверности.

Данная процедура повторяется для каждого подмножества ss из s.

Пример. Пусть имеется база транзакций, представленная в табл. 12.

База данных транзакций Таблица 12

Номер чека	Товар
160698	кетчупы, соусы, аджика, чай, макаронные изделия
160747	макаронные изделия, сыры
161243	кетчупы, соусы, аджика, макаронные изделия, сыры
161354	макаронные изделия, соусы, сыры
162915	вафли, сухари, чай
167414	чай, вафли, мед, сухари
167465	вафли, сыры, мед, сухари, чай
166474	кетчупы, сыры, макаронные изделия

Представим базу данных транзакций в нормализованном виде:

База данных транзакций в нормализованном виде Таблица 13

№ транзакции	кетчупы	соусы	аджика	чай	сыры	макар. изделия	вафли	сухари	мед
1	1	1	1	1	0	1	0	0	0
2	0	0	0	0	1	1	0	0	0
3	1	1	1	0	1	1	0	0	0
4	0	1	0	0	1	1	0	0	0
5	0	0	0	1	0	0	1	1	0
6	0	0	0	1	0	0	0	0	1
7	0	0	0	1	1	0	1	1	1
8	1	0	0	0	1	1	0	0	0
частота	3	3	2	4	5	4	2	2	2
поддержка	0,375	0,375	0,25	0,5	0,625	0,625	0,25	0,25	0,25

Будем считать популярными наборами те, которые имеют поддержку $S \geq 30$ %. Выпишем множество популярных однопредметных наборов:

$F_1 = \{кетчупы, соусы, чай, сыры, макар.изделия\}$.

Теперь переходим к поиску популярных двухпредметных наборов:

Двухпредметные наборы Таблица 14

набор	кол-во	поддержка	набор	кол-во	поддержка
кетчупы, соусы	2	0,25	соусы, чай	1	0,125
кетчупы, чай	1	0,125	соусы, сыры	1	0,125
кетчупы, сыры	2	0,25	соусы, макар. изделия	3	0,375
кетчупы, макар. изделия	3	0,375	чай, сыры	1	0,125
сыры, макар. изделия	4	0,5	чай, макар. изделия	1	0,125

Выпишем множество популярных двухпредметных наборов, которые имеют поддержку $S \geq 30$ %: $F_2 = \begin{matrix} \{кетчупы, макар.изделия\} \\ \{сыры, макар.изделия\} \\ \{соусы, макар.изделия\} \end{matrix}$

Выпишем трехпредметные наборы, используя операцию связывания популярных двухпредметных наборов.

$\{кетчупы, макар.изделия\}+\{соусы, макар.изделия\}=$

$=\{кетчупы, соусы, макар.изделия\}$, его поддержка S=0,25.

$\{кетчупы, макар.изделия\}+\{сыры, макар.изделия\}=$

$=\{кетчупы, сыры, макар.изделия\}$, его поддержка S=0,25.

Таким образом, трехпредметных популярных наборов нет, и задача поиска частных предметных наборов завершена.

Переходим к генерации ассоциативных правил на найденных популярных двухпредметных наборах.

Ассоциативные правила Таблица 15

Правило	Поддержка	Достоверность	Лифт	Левередж	Улучшение
Если кетчупы, то макар. изделия	3/8=0,375	3/3=1	1/0,625=1,6	0,375-0,375*0,625=0,14	0,375/(0,375*0,625)==1,6
Если макар. изделия, то кетчупы	3/8=0,375	3/4=0,75	0,75/0,375=2	0,375-0,375*0,625=0,14	0,375/(0,375*0,625)==1,6
Если сыры, то макар. изделия	4/8=0,5	4/5=0,6	0,6/0,625=0,96	0,5-0,625*0,625=0,11	0,5/(0,625*0,625)==1,28
Если соусы, макар. изделия	3/8=0,375	3/3=1	1/0,625=1,6	0,375-0,375*0,625=0,14	0,375/(0,375*0,625)==1,6

Рассмотрим первое правило *кетчупы→макаронные изделия:* S=37,5%; C=100 %; L=1,6; T=0,14; I=1,6. Это означает следующее:

Так как S=37,5 %, ожидаемая вероятность покупки набора *кетчупы+макаронные изделия* равна 37,5 %.

Так как C=100 %, это значит, что если покупатель купит *кетчупы*, то с вероятностью 100 % он купит и *макаронные изделия*.

Так как лифт L=1,6, то покупатель, купивший *кетчупы*, в 1,6 раза чаще выбирает *макаронные изделия*, нежели любой другой товар.

Так как левередж для данного правила T=0,14≠0, то правило значимо.

Так как улучшение для данного правила I=1,6>1, предсказать покупку макаронных изделий вероятнее, чем угадать случайно.

Из всех рассмотренных правил незначимое одно *сыры→макаронные изделия*, так как у него C=60 % и L=0,96.

Пример нахождения ассоциативных правил в аналитической платформе Deductor Studio Academic

Рассмотрим механизм поиска ассоциативных правил на примере данных о продажах товаров в некоторой торговой точке. Данные находятся в файле «Supermarket.txt». В таблице представлена информация по покупкам продуктов нескольких групп. Она имеет всего два поля «Номер чека» и «Товар». Необходимо решить задачу анализа потребительской корзины с целью последующего применения результатов для стимулирования продаж. Для этого производится поиск товаров, присутствие которых в транзакции влияет на вероятность наличия других товаров или комбинаций товаров.

Импортируем данные из текстового файла и представим в виде таблицы (при этом тип импортируемых данных – *строковый*):

Номер чека	Товар
160698	КЕТЧУПЫ, СОУСЫ, АДЖИКА
160698	МАКАРОННЫЕ ИЗДЕЛИЯ
160698	ЧАЙ
160747	МАКАРОННЫЕ ИЗДЕЛИЯ
160747	МЕД
160747	ЧАЙ
161217	КЕТЧУПЫ, СОУСЫ, АДЖИКА
161217	МАКАРОННЫЕ ИЗДЕЛИЯ

Рис. 42. Фрагмент базы данных транзакций

Для поиска ассоциативных правил запустим мастер обработки. В нем выберем тип обработки «Ассоциативные правила». На втором шаге мастера необходимо указать, какой столбец является идентификатором транзакции (чек), а какой элементом транзакции (товар).

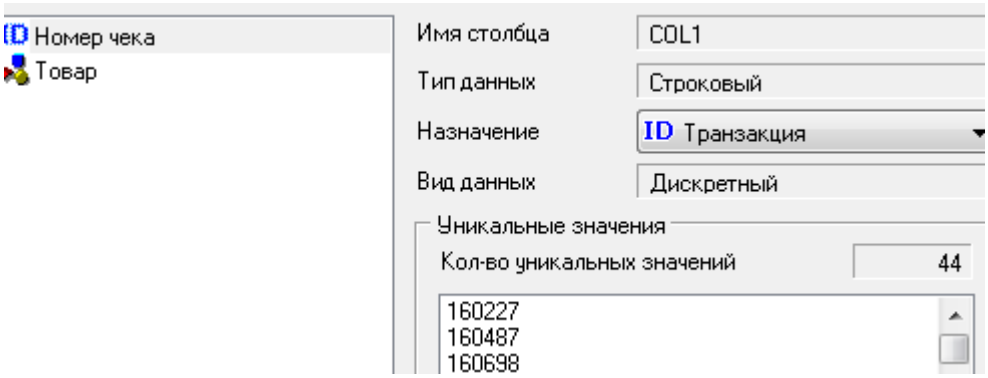


Рис. 43. Настройка назначения входных полей для решения задачи ассоциации

Следующий шаг позволяет настроить параметры построения ассоциативных правил: минимальную и максимальную поддержку, минимальную и максимальную достоверность, а также максимальную мощность множества.

По умолчанию в обработчике установлены следующие границы поддержки – 1 % и 20 %, и достоверности 40 % и 90 %, при которых для загруженных данных количество популярных наборов и ассоциативных правил равно 0. В этом случае следует увеличить максимальную поддержку.

Изменим верхнюю границу границы поддержки на 40 %.

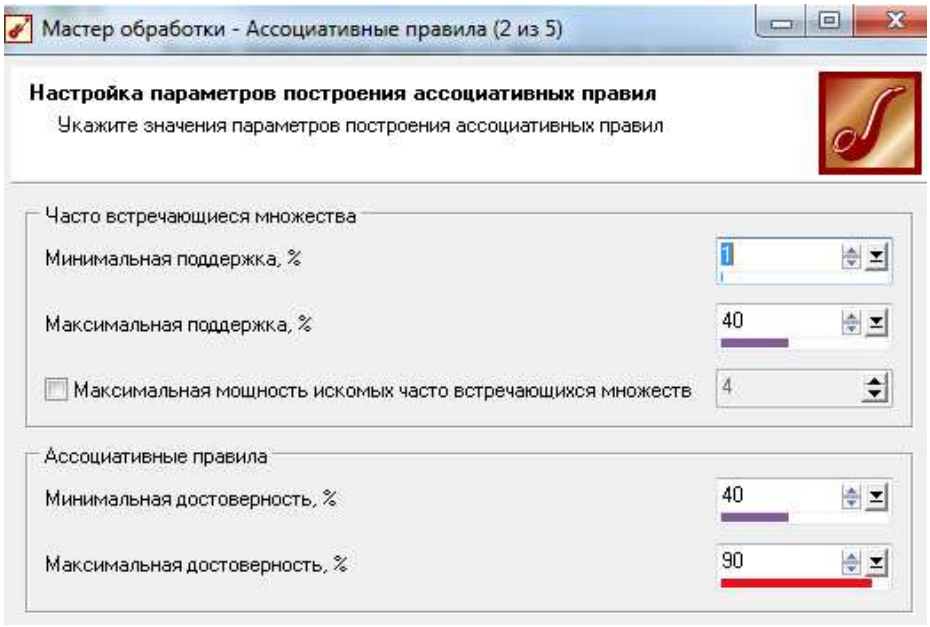


Рис. 44. Параметры алгоритма a priori

Следующий шаг позволяет запустить процесс поиска ассоциативных правил. На экране отображается информация о количестве множеств, количестве

найденных правил, а также гистограмма распределения найденных часто встречающихся множеств по мощности.

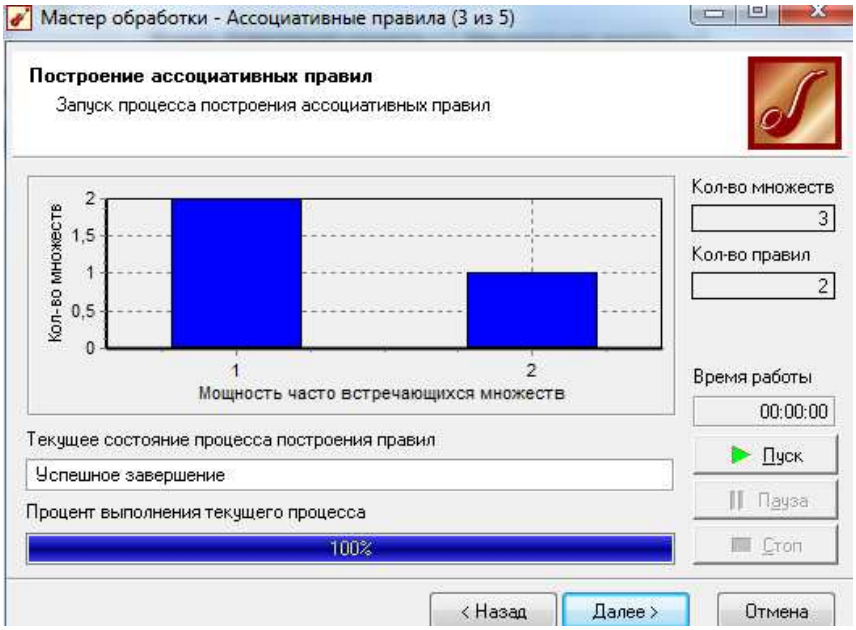


Рис. 45. Процесс выявления ассоциаций

После завершения процесса поиска полученные результаты можно посмотреть, используя появившиеся специальные визуализаторы «Популярные наборы», «Правила», «Дерево правил», «Что если». Популярные наборы – это множества, состоящие из одного и более элементов, которые наиболее часто встречаются в транзакциях одновременно. Насколько часто встречается множество в исходном наборе транзакций, можно судить по поддержке. Данный визуализатор отображает множества в виде списка.

Номер множества	Элемент	Поддержка		Мощность
		Кол-во	%	
1	ВАФЛИ	14	31,82	1
3	ВАФЛИ СУХАРИ	10	22,73	2
2	СУХАРИ	14	31,82	1

Рис. 46. Визуализатор «Популярные наборы»

Визуализатор «Правила» отображает ассоциативные правила в виде списка правил. Этот список представлен таблицей со столбцами: «номер правила», «условие», «следствие», «поддержка, %», «поддержка, количество», «достоверность».

g9

Номер правила	Условие	Следствие	Поддержка		Достоверность	Лифт
			Кол-во	%		
1	ВАФЛИ	СУХАРИ	10	22,73	71,43	2,245
2	СУХАРИ	ВАФЛИ	10	22,73	71,43	2,245
3	КЕТЧУПЫ, СОУСЫ	МАКАРОННЫЕ ИЗДЕЛИЯ	20	45,45	86,96	1,594

Рис. 47. Визуализатор «Правила»

Таким образом, эксперту предоставляется набор правил, которые описывают поведение покупателей. Например, если покупатель купил вафли, то он с вероятностью 71,4 % также купит и сухари.

Визуализатор «Дерево правил» – это всегда двухуровневое дерево. Оно может быть построено либо по условию, либо по следствию. При построении дерева правил по условию на первом (верхнем) уровне находятся узлы с условиями, а на втором уровне – узлы со следствием. Второй вариант дерева правил – дерево, построенное по следствию. Здесь на первом уровне располагаются узлы со следствием. Справа от дерева находится список правил, построенный по выбранному узлу дерева. Для каждого правила отображаются поддержка и достоверность. Если дерево построено по условию, то вверху списка отображается условие правила, а список состоит из его следствий. Тогда правила отвечают на вопрос, что будет при таком условии. Если же дерево построено по следствию, то вверху списка отображается следствие правила, а список состоит из его условий. Эти правила отвечают на вопрос, что нужно, чтобы было заданное следствие. Данный визуализатор отображает те же самые правила, что и предыдущий, но в более удобной для анализа форме.

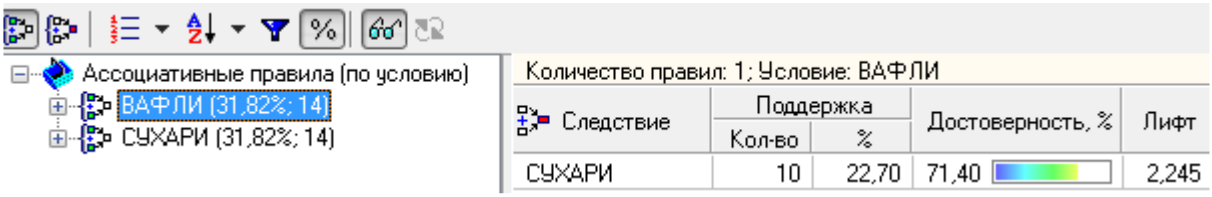


Рис. 48. Визуализатор «Дерево правил»

В данном случае правила отображены по условию. Тогда отображаемый в данный момент результат можно интерпретировать следующим образом:

Если покупатель приобрел вафли, то он с вероятностью 71 % также

приобретет сухари.

Выявление действительно интересных правил – это одна из главных подзадач при вычислении ассоциативных зависимостей. Для того чтобы получить действительно интересные зависимости, нужно разобраться с несколькими эмпирическими правилами:

1) Уменьшение минимальной поддержки приводит к тому, что увеличивается количество потенциально интересных правил, однако это требует существенных вычислительных ресурсов. Одним из ограничений уменьшения порога минимальной поддержки является то, что слишком маленькая поддержка правила делает его статистически необоснованным.
2) Правило со слишком большой поддержкой с точки зрения статистики представляет собой большую ценность, но с практической точки зрения это, скорее всего, означает то, что либо правило всем известно, либо товары, присутствующие в нем, являются лидерами продаж, откуда следует их низкая практическая ценность. Правило со слишком большой достоверностью практической ценности в контексте решаемой задачи не имеет, т.к. товары, входящие в следствие, покупатель, скорее всего, уже купил. Но ассоциативные правила с высокой поддержкой могут применяться для формализации хорошо известных правил, например в автоматизированных системах для управления процессами или персоналом.
3) Уменьшение порога достоверности также приводит к увеличению количества правил. Значение минимальной достоверности также не должно быть слишком маленьким, так как ценность правила с достоверностью 5 % чаще всего настолько мала, что это и правилом считать нельзя.
4) Интерпретация ассоциативных правил. Дело в том, что ассоциативные правила сами по себе, как результат работы некоторого алгоритма, еще не готовы к использованию. Их нужно проинтерпретировать, т.е. понять, какие из ассоциативных правил представляют интерес, действительно ли правила отражают закономерности или, наоборот, являются артефактом.

Это требует тщательной работы аналитика и понимания предметной области, в которой решается задача ассоциации.

Все множество ассоциативных правил можно разделить на три вида:

1) ***Полезные правила*** содержат действительную информацию, которая ранее была неизвестна, но имеет логичное объяснение. Такие правила могут быть использованы для принятия решений, приносящих выгоду.
2) ***Тривиальные правила*** содержат действительную и легко объяснимую информацию, которая уже известна. Такие правила, хотя и объяснимы, но не могут принести какой-либо пользы, т.к. отражают известные законы в исследуемой области или результаты прошлой деятельности. При анализе рыночных корзин в правилах с самой высокой поддержкой и достоверностью окажутся товары-лидеры продаж. Практическая ценность таких правил крайне низка.
3) ***Непонятные правила*** содержат информацию, которая не может быть объяснена. Такие правила могут быть получены или на основе аномальных значений, или глубоко скрытых знаний. Напрямую такие правила нельзя использовать для принятия решений, т.к. их необъяснимость может привести к непредсказуемым результатам. Для лучшего понимания требуется дополнительный анализ.

Варьируя верхним и нижним пределами поддержки и достоверности, можно избавиться от очевидных и неинтересных закономерностей, можно найти действительно интересные и полезные правила.

В рассматриваемом примере, исходя из характера имеющихся данных, укажем границы поддержки – 13 % и 80 %, и достоверности – 60 % и 90 %. В результате количество популярных наборов и количество правил увеличится (рис. 49).

Рис. 49. Процесс выявления ассоциаций

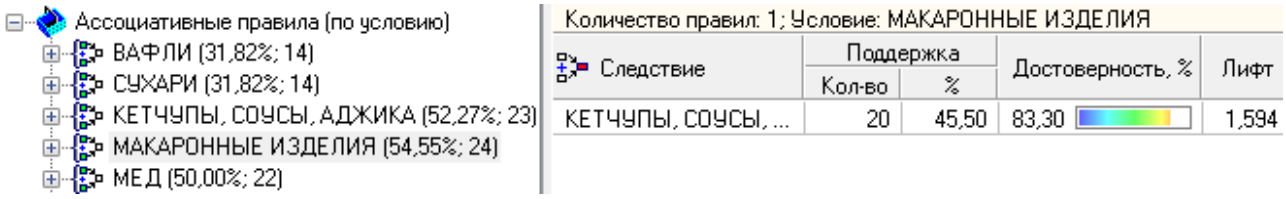


Рис.50. Визуализатор «Дерево правил»

Появились новые правила. Например, если покупатель приобрел макаронные изделия, то он с вероятностью 83,3 % также приобретет кетчупы и соусы. Это правило можно отнести к тривиальным, так как товары, присутствующие в нем, являются лидерами продаж (см. рис. 51), они имеют высокую поддержку.

Множеств: 7 из 30 | Фильтр: Минимальная мощность = 1; Максимальная мощность = 1

№	Элементы	Поддержка		Мощность
		Кол-во	%	
1	ВАФЛИ	14	31,82	1
2	КЕТЧУПЫ, СОУСЫ, АДЖИКА	23	52,27	1
3	МАКАРОННЫЕ ИЗДЕЛИЯ	24	54,55	1
4	МЕД	22	50,00	1
5	СУХАРИ	14	31,82	1
6	СЫРЫ	19	43,18	1
7	ЧАЙ	33	75,00	1

Рис. 51. Визуализатор «Популярные наборы» с фильтрацией по мощности

Наиболее удобным и оперативным инструментом использования ассоциативных правил является анализ «Что если». Внешний вид формы для проведения такого анализа представлен на рис. 52.

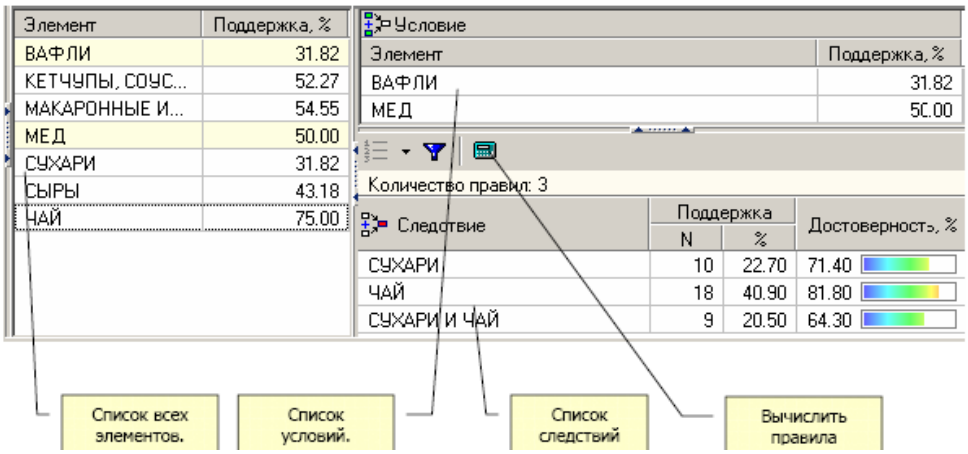


Рис. 52. Визуализатор «Что если»

Анализ «Что если» в ассоциативных правилах позволяет ответить на вопрос: «Что получим в качестве следствия, если выберем данные условия? Например, какие товары приобретаются совместно с выбранными товарами?» В окне слева расположен список всех элементов транзакций. Справа от каждого элемента указана поддержка «Сколько раз данный элемент встречается в транзакциях?» В правом верхнем углу расположен список элементов, входящих в условие. Это, например, список товаров, которые приобрел покупатель. Для них нужно найти следствие. Например, товары, приобретаемые совместно с ними, чтобы предложить человеку то, что он, возможно, забыл купить. В правом нижнем углу расположен список следствий. Справа от элементов списка отображается поддержка и достоверность.

Пусть необходимо проанализировать, что, возможно, забыл покупатель приобрести, если он уже взял вафли и мед? Для этого необходимо добавить в список условий эти товары (например, с помощью двойного щелчка мыши) и затем нажать на кнопку «Вычислить правила». При этом в списке следствий появятся товары, совместно приобретаемые с данными. В данном случае появятся «сухари», «чай», «сухари и чай». Возможно, покупатель забыл приобрести сухари или чай, или и то и другое.

Существующий в АП Deductor набор визуализаторов позволяет эксперту найти интересные, необычные закономерности, понять, почему так происходит и применить их на практике. Результаты анализа можно применить и для сегментации покупателей по поведению при покупках, и для анализа предпочтений клиентов, и для планирования расположения товаров в супермаркетах, кросс-маркетинге.

В данном примере найденные правила можно использовать для сегментации клиентов на два сегмента: клиенты, покупающие макаронные изделия и соусы к ним, и клиенты, покупающие все к чаю. В разрезе анализа предпочтений можно узнать, что наибольшей популярностью в данном магазине пользуются чай, мед, макаронные изделия, кетчупы, соусы и аджика. В разрезе размещения товаров в супермаркете можно применить результаты предыдущих двух анализов – располагать чай рядом с медом, а кетчупы, соусы и аджику – рядом с макаронными изделиями и т.д.

3.3. Применение искусственных нейронных сетей в СППР

Нейронная сеть – это самообучающаяся система, способная анализировать вновь поступающую информацию, находить в ней закономерности, производить прогнозирование и пр. Под нейронными сетями подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга (рис. 53).

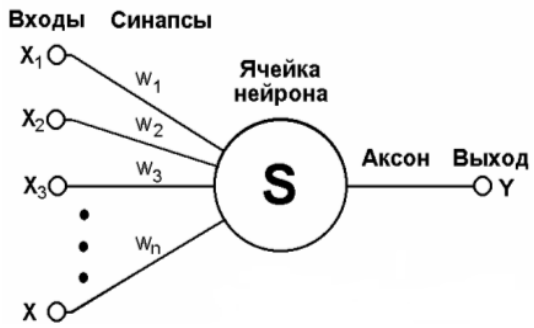


Рис. 53. Схема нейрона

$x_1 \ldots x_n$ – входные сигналы нейрона, приходящие от других нейронов. $W_1 \ldots W_n$ – синапсические веса.

Умножители (синапсы) осуществляют связь между нейронами, умножают входной сигнал на число, характеризующее силу связи.

Сумматор – сложение сигналов, поступающих по синапсическим связям от других нейронов. Состояние нейрона определяется по формуле: $S = \sum_{i=1}^{n} x_i w_i$,

где n – число входов нейрона; x_i – значение i-го входа нейрона; w_i – вес i-го синапса.

Работа нейронной сети состоит в преобразовании входного вектора в выходной вектор, причем это преобразование задается весами нейронной сети. Процесс обучения заключается в подстройке весов нейронов. Целью обучения является поиск состояния весов, которые минимизируют выходную ошибку сети в обучающем и тестовых множествах.

Модель нейрона:

1) вычисляет взвешенную сумму своих входов от других нейронов;

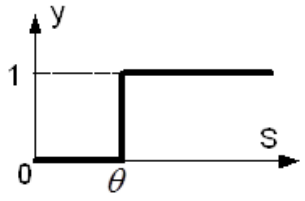
2) на входах нейрона имеются возбуждающие и тормозящие синапсы;

3) при превышении суммы входов порога нейрона вырабатывается выходной сигнал.

Нелинейный преобразователь реализует нелинейную функцию одного аргумента – выхода сумматора. Значение аксона нейрона определяется по формуле $y = f(S)$. Эта функция называется функцией активации или передаточной функцией нейрона.

Виды активационных функций:

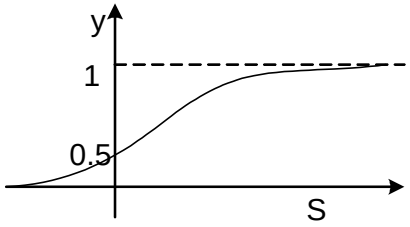
1) пороговая функция: область значения (0;1)



$$f(s) = \begin{cases} 1, s \geq Q \\ 0, \text{иначе} \end{cases}$$

«+»: простота реализации и высокая скорость вычисления

2) Сигмоидальная (логистическая функция)



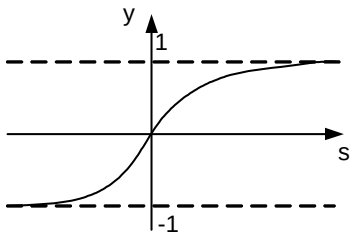
$$f(s) = \frac{1}{1+e^{-as}}$$

При уменьшении α сегмент становится более пологим, при α=0 – прямая линия.

Эта функция наиболее часто используется в качестве функции активации, так как ее достоинством является простое выражение ее производной, а также способность усиливать сигналы слабые лучше, чем большие, и предотвращать насыщения от больших сигналов.

«–»: область значения малая (0,1).

3) Гиперболический тангенс: область значений (–1,1)



$$f(s) = th(bs) = \frac{e^{as} - e^{-as}}{e^{as} + e^{-as}}\ .$$

Сильная зависимость нейронных сетей от особенностей входных данных и характера искомых закономерностей делает процесс построения моделей на основе нейронных сетей *неоднозначным*.

ИНС – это множество нейронов, соединенных определенным образом между собой. При этом:

1) выделяют 3 типа нейронов: входные, выходные, промежуточные;

2) функции активации считаются неизмененными в работе сети;

3) веса являются параметрами сети и корректируются, т.е. изменяются для достижения сетью некоторой поставленной цели.

Классификация ИНС. Задачи, решаемые с помощью нейронных сетей

1) Выделяют 3 основных типа сетей (по топологии):

а) Полносвязные сети. Каждый нейрон передает свой входной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Сеть Хопфильда;

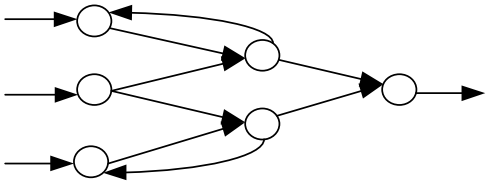
б) Многослойные сети – нейроны объединяются в слои.

Выделяют сети:

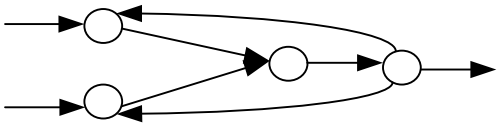
– без обратной связи – нейрон входного слоя получает входной сигнал, преобразует его и передает нейрону одного скрытого слоя, вплоть до выходного. Нейроны могут связываться через один или несколько слоев. Среди многослойных сетей различают полносвязные сети и частично – полносвязные. Классический вариант: полносвязные сети прямого распространения;

– с обратной связью – информация с последующих слоев передается на предыдущий.

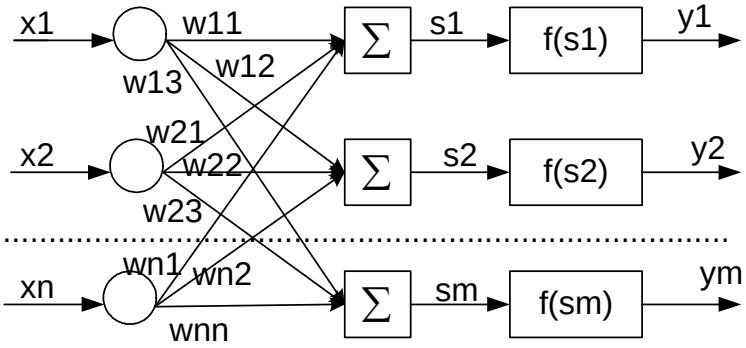
Сеть Элмана – частично рекуррентная сеть с обратными связями



Сеть Жордана – частично рекуррентная, слоистая

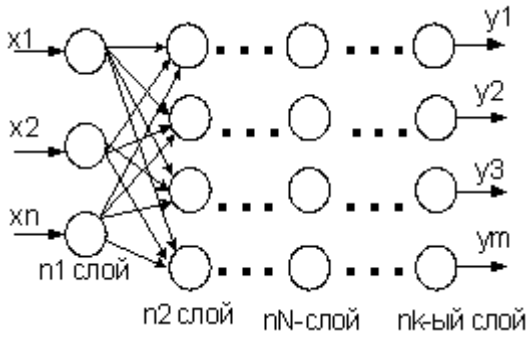


Многослойный персептрон Ф. Розенблатта – многослойная искусственная нейронная сеть прямого распространения



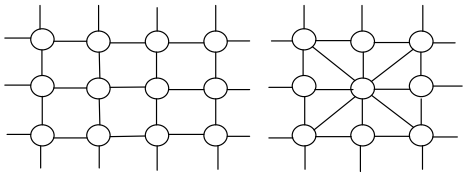
Многослойный персептрон состоит из совокупности узлов (нейронов), соединенных между собой связями. Существуют три типа узлов: входной, скрытый и выходной.

Каждый узел является своеобразным обрабатывающим модулем. Соединяющие их связи имеют веса (вес – это числовой параметр).



В каждом слое может быть различное количество элементов. Нейроны выходного слоя принимают синаптические сигналы от нейронов предыдущего слоя. Нейроны в пределах одного слоя не соединяются между собой.

в) Слабосвязные НС. Нейроны располагаются в узлах прямоугольной (гексогональной) решетки (сети Кохонена, самоорганизующиеся карты).



2) Классификация по типу структур нейронов:

– гомогенные (однородные) сети состоят из нейронов одного типа с единой функцией активности;

– гетерогенные нейросети состоят из нейронов с различными функциями активации.

3) Классификация по видам сигналов:

– бинарные сети;

– аналоговые сети.

4) По методу обучения:

– обучение с учителем;

– обучение без учителя;

– смешанное.

Обучение с учителем – это один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций) и множество возможных ответов (откликов, реакций). Существует некоторая зависимость между ответами и объектами, но она неизвестна. Известна только конечная совокупность прецедентов – пар «объект, ответ», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость,

то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ. Под учителем в данном случае понимается сама выборка или тот, кто указал на заданных объектах правильные ответы.

Существует также обучение без учителя: когда на объектах выборки ответы не задаются. Модель путем самоорганизации делает необходимые изменения. Применяются в задачах кластеризации и сегментации для поддержки принятия решений.

Классы задач, решаемых НС:

1) Классификация (распознавание образов).
2) Кластеризация.
3) Аппроксимация функции.

Формируется набор пар $D = \{(x_1, y_1); (x_2, y_2); ...(x_n, y_n);\}$, которые генерируются функцией $y = f(x_1, x_2, \ldots, x_n)$. Требуется найти вид функции, удовлетворяющей некоторым критериям. ИНС – универсальный аппроксиматор.

4) Прогнозирование:

Дано $\{y(t_1), y(t_2), ..., y(t_n)\}$. Требуется найти: $y(t_{n+1})$.

5) Оптимизация применяется в задачах с большим пространством поиска.
6) Ассоциативная память позволяет восстановить содержание по частично искаженному представлению занесенных данных.

Классификация известных нейросетевых структур по типу связей и типу обучения и их применение

Тип обучения / *Тип связей*	*Обучение с учителем*	*Обучение без учителя*
Без обратной связи	Многослойный персептрон (задачи классификации и аппроксимации)	Сети Кохонена (задача кластеризации и сжатия данных)
С обратной связью	Рекуррентные аппроксиматоры (задача прогнозирования)	Сети Хопфильда (задачи кластеризации, оптимизации, ассоциативная память)

Основные этапы нейросетевого анализа:

Этап 1. Подготовка исходных данных. Формирование обучающей выборки. (Данные не должны быть противоречивыми.)

Этап 2. Необходим выбор типа архитектуры НС:

– Выбор типа нейрона со своей функцией активации.

– Выбор количества входов и выходов, что связано с постановкой задачи.

– Выбор количества слоев и нейронов в каждом слое.

Этап 3. Подготовка данных.

– Кодирование входов/выходов.

– Нормировка данных (если необходимо).

– Обеспечение независимости между входами нейросети – предобработка данных.

Этап 4. Процесс обучения сети.

Перед тем, как приступить к обучению искусственной нейронной сети, необходимо задать ее начальное состояние. От того, насколько удачно будут выбраны начальные значения весовых коэффициентов, зависит время обучения.

Этап 5. Проверка модели на адекватность реальным данным.

Необходимо оценить, как искусственная нейронная сеть будет обрабатывать примеры, которые не входили в обучающую выборку. Проверка работоспособности модели на контрольных примерах осуществляется за счет использования разнообразных статистических критериев согласия.

В случае, если нас не устраивает результат, возвращаемся к этапу 2.

Этап 6. Выбор нейросети, которая наилучшим образом подходит по результатам обучения для решения задачи.

Алгоритмы обучения нейронных сетей. Алгоритм обратного распространения ошибки

Под обучением понимается целенаправленное изменение весовых коэффициентов синоптических связей нейронов сети из условий достижения требуемых характеристик сети, т.е. желаемая реакция на входные воздействия. В

основе лежит базовый принцип обучения – минимизация эмпирической ошибки между желаемым выходом сети и фактической реакции сети.

В процессе обучения нейронная сеть выявляет сложные зависимости между входными данными и выходными, а также выполняет обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке.

Теорема Розенблатта: Для любого данного набора входных векторов и любой требуемой их классификации алгоритм обучения через конечное число шагов приведет к вычислению требуемого набора весов, если таковой существует.

Алгоритм обратного распространения ошибки **(**BackProp) представляет собой градиентный алгоритм обучения многослойного персептрона, основанный на минимизации среднеквадратической ошибки выходов сети. Алгоритм был предложен в 1974 г. П. Дж. Вербосом, а в 1986 г. его развили Д. Румельхарт и Р. Вильямс.

Обучение искусственной нейронной сети включает 3 этапа:

1) прямое распространение входного обучающего образа;

2) вычисление ошибки и ее обратного распространения;

3) регулирование весов.

Основная идея алгоритмов обучения с учителем, к которым относится и BackProp, заключается в том, что на основе разности между желаемым и целевым выходами сети можно вычислить выходную ошибку сети. Цель определения выходной ошибки – управление процессом обучения нейронной сети, то есть корректировки весов ее межнейронных связей для минимизации функции ошибки.

Корректировка весов сети делается по правилу Видроу – Хоффа. Каждый нейрон в сети получает возбуждение от вектора входных значений, производит их взвешенное суммирование и преобразует полученную сумму с помощью активационной функции. Выходная ошибка сети формируется на нейронах вы-

ходного слоя. Но это не означает, что погрешность работы сети обусловлена только выходными нейронами. Свой вклад в результирующую ошибку вносит каждый скрытый нейрон. Тогда для него может быть указана ошибка $\delta = d - y$, где d – желаемое выходное значение, а y – реальное выходное значение.

Правило Видроу-Хоффа, называемое также δ-правилом, подразумевает корректировку весов в соответствии с выражением $\Delta w_i = \eta \delta x_i$, (1)

где η–коэффициент скорости обучения $0<\eta<1$; x_i – значения, поступающие по входным связям.

Зная величину коррекции веса i-связи, мы можем вычислить ее новый вес по формуле $w_i(k+1) = w_i(k) + \Delta w_i$, (2)

где k – номер итерации обучения.

Выходная ошибка сети определяется по формуле: $E = \frac{1}{2}\sum_j \left(y_j - d_j\right)^2$. (3)

С учетом того, что вес должен изменяться в направлении, противоположном знаку производной функции ошибки, получим: $\Delta w_{ij} = \delta_j x_i$. (4)

Ошибка $\delta_j = d_j - y_j$, где d_j – желаемое выходное значение, а y_j – реальное выходное значение, была получена для выходного нейрона. Что касается ошибки на выходах скрытых нейронов, то она не имеет непосредственной связи с выходной ошибкой. Поэтому вес скрытых нейронов в соответствии с (4) приходится корректировать по его вкладу в величину ошибки следующего слоя. В сети с одним скрытым слоем при распространении сигнала ошибки в обратном направлении ошибка выходного нейрона также вносит вклад в ошибку каждого нейрона скрытого слоя. Этот вклад зависит от значения ошибки выходного нейрона и веса связи, соединяющей его со скрытым нейроном, ошибку которого требуется определить. Чем больше ошибка на выходном нейроне и больше вес связи, соединяющей его со скрытым нейроном, тем больше ошибка на выходе скрытого нейрона.

Вычислить ошибку для нейронов выходного δ_j^{out} и скрытого слоев можно по формулам: $\delta_j^{out} = y_j(1-y_j)\cdot(d_j - y_j)$ (5)

$$\delta_j^{hit} = y_j(1-y_j)\cdot\sum_k \delta_j w_{kj} \quad (6)$$

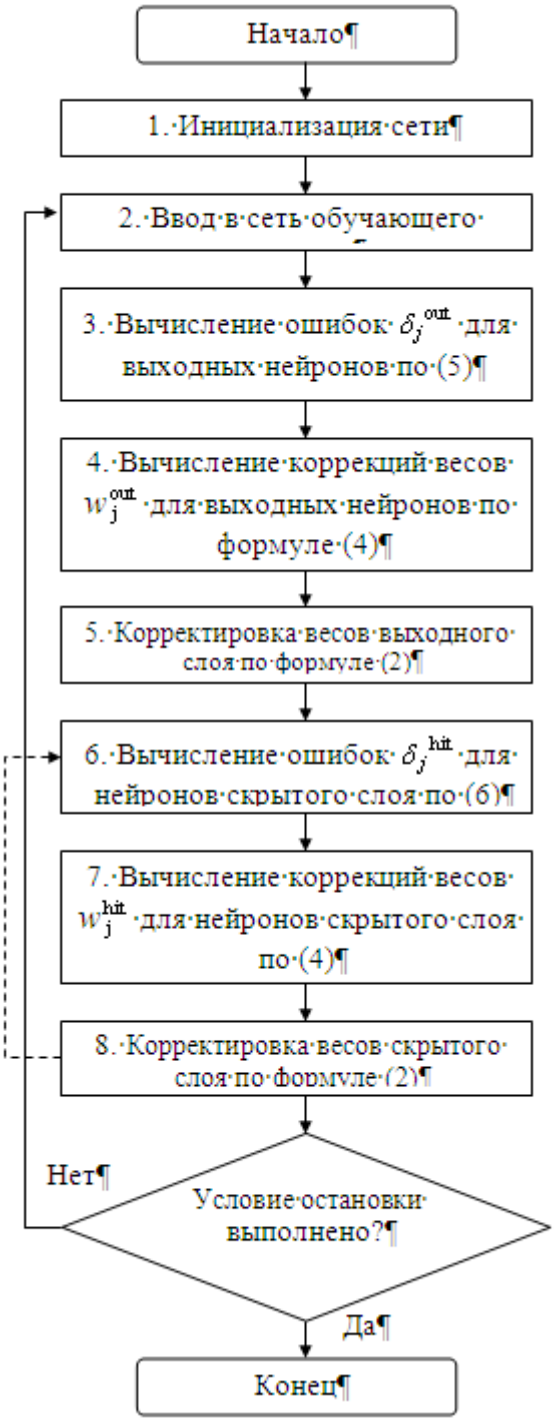


Рис. 54. Блок-схема алгоритма обратного распространения ошибки для нейронной сети с одним скрытым слоем

Пример. Выполним 3 цикла прямого и обратного прохода ИНС, используя алгоритм обратного распространения ошибок для входного образца (0,1; 0,9) и целевого выходного значения 0,9 в предположении, что сеть имеет архитектуру 2 – 3 – 1 с весовыми коэффициентами:

для первого слоя:

$$\begin{pmatrix} 0,1 & 0,1 \\ -0,2 & -0,1 \\ 0,1 & 0,3 \end{pmatrix}$$

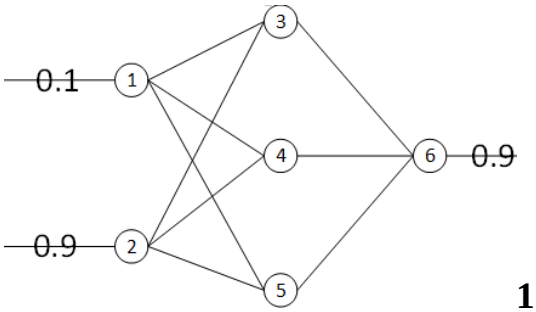
для второго слоя:

$$\begin{bmatrix} 0,2 \\ 0,2 \\ 0,3 \end{bmatrix}$$

Параметры обучения: $\alpha = 1$ и $\eta = 0,1$.

Сделать вывод: уменьшается ли ошибка E?

Решение.



1-я итерация:

Прямой проход: $S_j = \sum_{i=1}^{n} x_i w_{ij}$, $f(s) = \dfrac{1}{1+e^{-as}}$

*S*3=	0,1	*y*(*S*3)=	0,524977
*S*4=	-0,11	*y*(*S*4)=	0,472531
*S*5=	0,28	*y*(*S*5)=	0,569539
*S*6=	0,370363	*y*(*S*6)=	0,591537

E=	0,047575

Обратный проход:

δ6=	0,074531
δ3=	0,003717
δ4=	0,003715
δ5=	0,005482

Δ*w*36=	0,003913
Δ*w*46=	0,003522
Δ*w*56=	0,004245
Δ*w*13=	0,000037
Δ*w*14=	0,000037
Δ*w*15=	0,000055
Δ*w*23=	0,000335
Δ*w*24=	0,000334
Δ*w*25=	0,000493

*w*нов36=	0,203913
*w*нов46=	0,203522
*w*нов56=	0,304245
*w*нов13=	0,100037
*w*нов14=	–0,19996
*w*нов15=	0,100055
*w*нов23=	0,100335
*w*нов24=	–0,09967
*w*нов25=	0,300493

2-я итерация:

Прямой проход:

*S*3=	0,100309	y(*S*3)=	0,525054
*S*4=	–0,10969	y(*S*4)=	0,472607
*S*5=	0,280453	y(*S*5)=	0,56965
*S*6=	0,376564	y(*S*6)=	0,593035

E=	0,047114

Обратный проход:

δ6=	0,074084
δ3=	0,003767
δ4=	0,003758
δ5=	0,005526

Δ*w*36=	0,00389
Δ*w*46=	0,003501
Δ*w*56=	0,00422
Δ*w*13=	0,000038
Δ*w*14=	0,000038
Δ*w*15=	0,000055
Δ*w*23=	0,000339
Δ*w*24=	0,000338
Δ*w*25=	0,000497

*w*нов36=	0,207803
*w*нов46=	0,207023
*w*нов56=	0,308465
*w*нов13=	0,100075
*w*нов14=	–0,19992
*w*нов15=	0,100111
*w*нов23=	0,100678
*w*нов24=	–0,09932
*w*нов25=	0,300995

3-я итерация:

Прямой проход:

*S*3=	0,100622	y(S3)=	0,525132
*S*4=	–0,10938	y(S4)=	0,472685
*S*5=	0,28091	y(S5)=	0,569762
*S*6=	0,382732	y(S6)=	0,594522

E=	0,046658

Обратный проход:

δ6=	0,07364
δ3=	0,003816
δ4=	0,0038
δ5=	0,005568

Δ*w*36=	0,003867
Δ*w*46=	0,003481
Δ*w*56=	0,004196
Δ*w*13=	0,000038
Δ*w*14=	0,000038
Δ*w*15=	0,000056
Δ*w*23=	0,000343
Δ*w*24=	0,000342
Δ*w*25=	0,000501

*w*нов36=	0,21167
*w*нов46=	0,210504
*w*нов56=	0,312661
*w*нов13=	0,100114
*w*нов14=	–0,19989
*w*нов15=	0,100167
*w*нов23=	0,101026
*w*нов24=	–0,09898
*w*нов25=	0,3015

Вывод: с каждой итерацией ошибка уменьшается.

Пример построения классификатора на основе нейронной сети для оценки недвижимости в аналитической платформе Deductor Studio Academic.

Рассмотрим построение модели классификации, относящей объекты недвижимости на основе их признаков к одному из трех классов: «дорогие», «средние», «дешевые квартиры».

В аналитической платформе Deductor существует специальный обработчик «Нейросеть», который реализует модель многослойного персептрона.

Выполнив предварительную обработку данных, используя обработчики «Квантование» и «Замена значений» (аналогично тому, как это было сделано в классификаторе «Дерево решений»), выберем узел «Нейросеть».

На следующем шаге установим назначения полей и нажмем кнопку «Настройка нормализации» (рис. 55). Здесь задаются способы кодирования для непрерывных и категориальных признаков, а также диапазон изменения входных сигналов (по умолчанию от –1 до 1) и выходных (от 0 до 1).

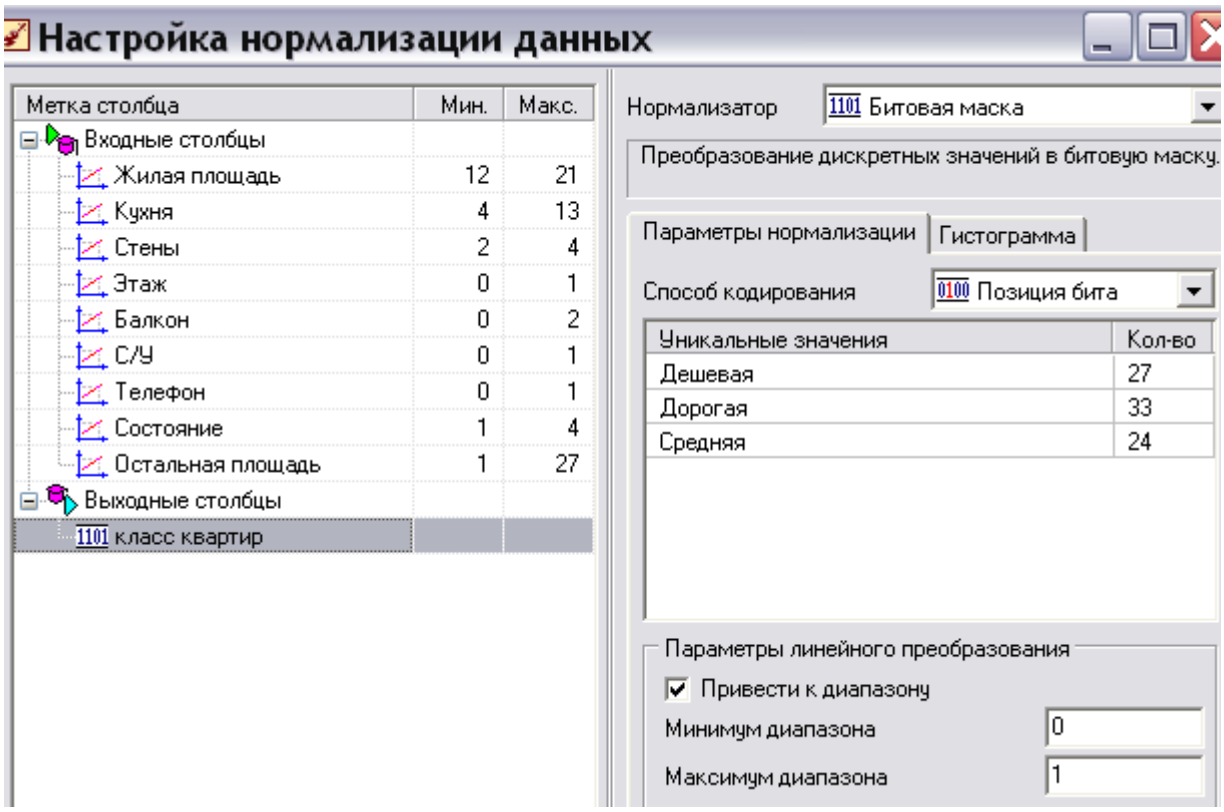


Рис. 55. Настройка нормализации полей

Следующим этапом будет разбиение исходного множества (случайным образом) на 2 подмножества: возьмем обучающее (95 %) и тестовое (5 %).

В следующем окне мастера задается архитектура многослойного персептрона и параметры активационной функции (рис. 56).

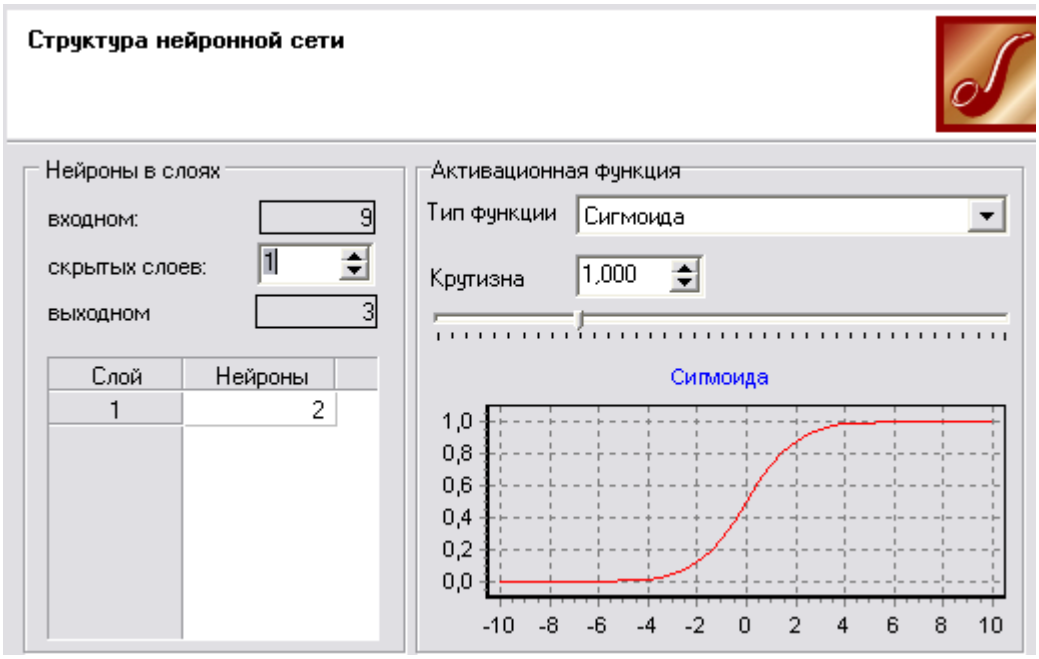


Рис. 56. Настройка структуры нейронной сети

На следующем шаге выбирается алгоритм обучения многослойного персептрона и обучения. Выберем алгоритм Back-Propagation, а коэффициенты, отвечающие за скорость и момент обучения, оставим без изменений (рис. 57).

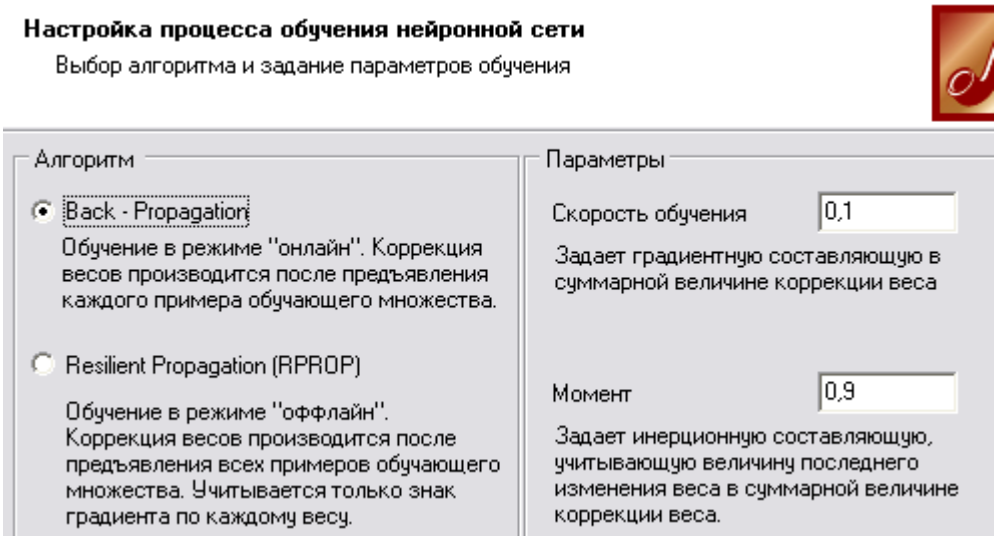


Рис. 57. Выбор алгоритма обучения многослойного персептрона

Далее необходимо задать условия, при выполнении которых обучение будет прекращено (рис. 58). Остановка обучения происходит по достижению любого из заданных условий остановки:

- *считать пример распознанным, если ошибка* (рассогласование между эталонным и реальным выходом сети) становится меньше заданного значения;
- *по достижении эпохи* – установка данного режима позволяет задать число эпох (циклов обучения), по достижении которого обучение останавливается независимо от величины ошибки;
- *обучающее множество* – остановка обучения производится по достижении на обучающем множестве заданной средней ошибки, максимальной ошибки или процента распознанных примеров;
- *тестовое множество* – остановка обучения производится по достижении на тестовом множестве заданной средней ошибки, максимальной ошибки или процента распознанных примеров.

Примем, что пример следует считать распознанным, если ошибка станет менее 0,05, и укажем в поле *Эпоха* 10 000.

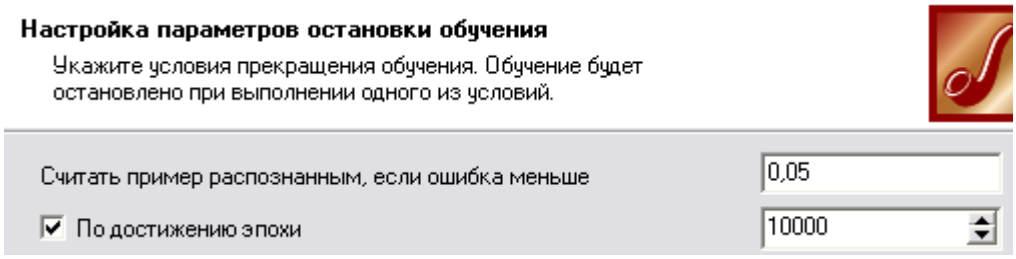


Рис. 58. Параметры остановки обучения нейросети

Теперь все готово к процессу обучения сети. В зависимости от объема обрабатываемых данных и быстродействия компьютера процесс обучения ИНС может занять определенное время (часто достаточно большое).

После запуска процесса обучения строится нейронная сеть, на выходе которой получаем три класса объектов недвижимости («дешевая», «средняя», «дорогая») (рис. 59).

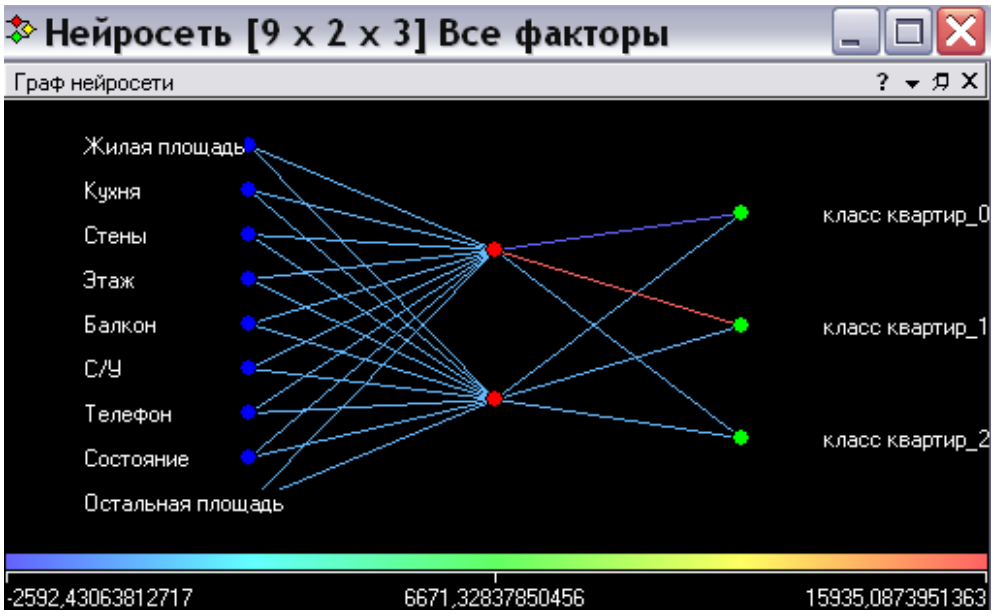


Рис. 59. Граф нейросети задачи классификации объектов недвижимости

После того как процесс обучения сети завершится, выберем визуализаторы: *Граф нейросети*, *Таблица сопряженности*, *Что если*.

Визуализатор *Граф нейросети* позволяет представить ИНС со всеми нейронами и синоптическими связями. При этом можно увидеть не только структуру НС, но и значения весов всех связей. В зависимости от веса их цвет меняется, а соответствующее числовое значение можно определить на цветовой шкале, расположенной в нижней части окна.

Таблица сопряженности (рис. 60) позволяет анализировать согласование значений, полученных в результате обработки исходной выборки с реальными результатами.

Таблица сопряженности

	Классифицировано			
Фактически	Дешевая	Дорогая	Средняя	Итого
Дешевая	25		2	27
Дорогая		32	1	33
Средняя	8	1	15	24
Итого	33	33	18	84

Рис. 60. Таблица сопряженности для ИНС

В таблице сопряженности ячейки с числом правильно распознанных примеров отображаются в зеленых ячейках, а неправильно распознанных – в красных. Чем большее число примеров попали в зеленные ячейки, тем лучше ре-

зультаты классификации. Кроме этого, в таблице сопряженности хорошо видно, по каким значения выходного поля было допущено наибольшее число ошибок классификации.

Нажатие кнопки ∑ . Суммарная информация (F4) открывает окно «Качество классификации» (рис.61), где в виде круговой диаграммы отображается общее соотношение правильно и неправильно классифицированных примеров. Над диаграммой указывается множество исходной выборки, на основе которой построена диаграмма, а все вместе – обучающее или тестовое.

Рис. 61. Диаграмма «Качество классификации ИНС»

Рис. 62. Применение визуализатора «Что если»

При помощи визуализатора «Что если» (рис. 62) имеется возможность проверить, как работает построенный нейросетевой классификатор. А именно: с помощью данного визуализатора можно определить класс, к которому относится квартира с новыми заданными параметрами.

Например, если клиент задает следующие параметры: Жилая площадь – 13 кв.м, Кухня – 5кв.м, Стены – кирпичные, Этаж 9, Балкон имеется, Санузел разделенный, Телефон имеется, Состояние квартиры – отличное, то на выходе определяем класс: «Дешевая квартира».

Построение нейросетевой модели прогнозирования стоимости недвижимости

В результате изучения предметной области должна быть разработана модель прогнозирования, составляющими которой должны быть:

- набор входных переменных;
- метод формирования входных признаков *x*;
- метод формирования обучающего правила *y*;
- архитектура нейронной сети;
- метод обучения нейронной сети;
- анализ адекватности и точности построенного прогноза.

Работа нейронной сети аналогична работе эксперта, который может оценить стоимость объекта недвижимости только на основе его свойств (признаков). Объекты недвижимости описываются определенным набором стандартных признаков, рассматриваемых экспертом и формирующих цену, и поэтому оценка недвижимости хорошо формализуется для решения методами регрессии, в том числе нейросетевыми. На вход сети подаются значения признаков определенного объекта недвижимости, а на выходе формируется оценка его стоимости. С получением входных данных обычно проблем не возникает, поскольку исчерпывающую информацию о рынке недвижимости можно получить с помощью различных агентств. Желаемый выход также хорошо определен – цена. Кроме того, имеется богатый опыт в виде предыдущих продаж для обучения нейронной сети.

Для решения задачи будем использовать те же атрибуты объектов недвижимости, что и в примере нейросетевой классификации объектов недвижимости. Архитектура многослойного персептрона будет отличаться только тем, что выходной слой теперь имеет один нейрон, где будем получать прогнозируемое значение цены (рис. 63).

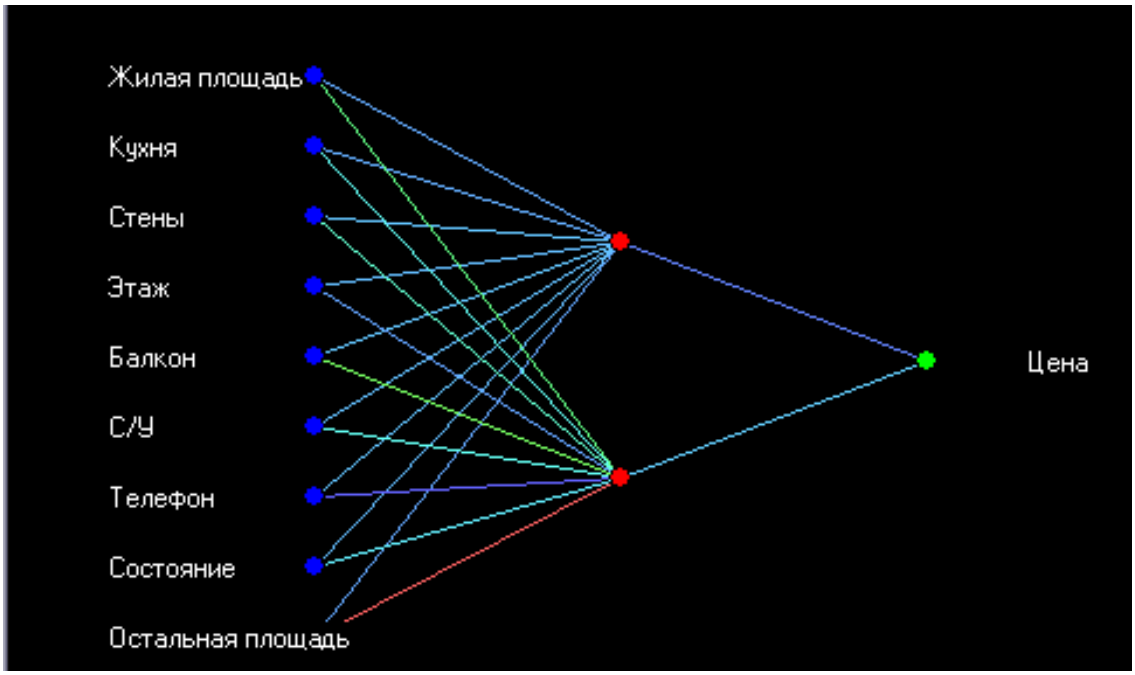


Рис. 63. Граф нейросети для решения задачи прогнозирования

Качество построенной нейронной сети подтверждает диаграмма рассеяния, показывающая хорошие прогностические результаты построенной нейросетевой модели (рис. 64).

Рис. 64. Диаграмма рассеяния

Диаграмма рассеяния служит для наглядной оценки качества построенной модели с помощью результатов сравнения непрерывных значений выходного поля и непрерывных значений того же поля, но рассчитанных моделью. На диаграмме рассеяния отображаются выходные значения для каждого из примеров обучающей выборки, координаты которых по оси X – это значение выхода на обучающей выборке (эталон), а по оси Y – значение выхода, рассчитанное обученной моделью на том же примере. Прямая диагональная линия представляет собой ориентир (линию идеальных значений). Чем ближе точка к этой линии, тем меньше ошибка модели. Также на диаграмме рассеяния отображаются две пунктирные линии – верхняя и нижняя границы доверительного интервала. Ширина доверительного интервала определяется допустимой ошибкой, которая вводится в поле «Ошибка». Если ошибка модели (величина в столбце <Имя_поля>_ERR) меньше допустимой, то точка попадает в доверительный интервал. С помощью доверительного интервала можно оценить, в каких точках отклонение рассчитанного моделью выхода от эталона является недопустимым, и в дальнейшем исследовать эти записи детальней.

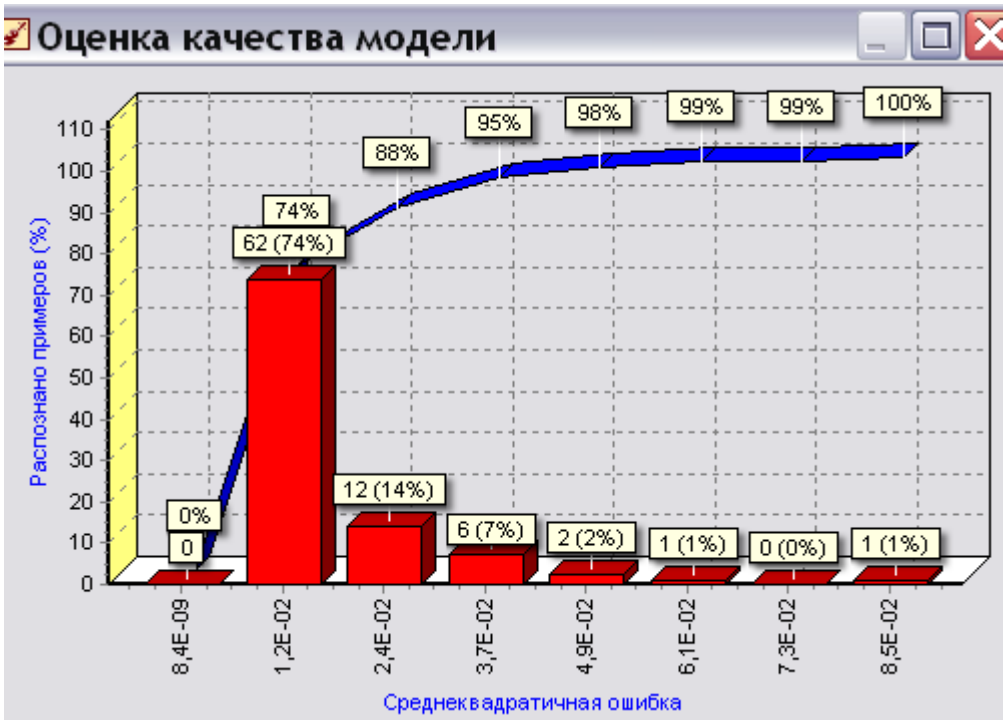


Рис. 65. Оценка качества построенной модели нейросетевого прогнозирования

3.4. Применение карт Кохонена в СППР

Одной из задач, решаемой Data Mining, является кластеризация данных.

Кластеризация – это группировка объектов на основе данных, описывающих свойства объектов.

В Data Mining для кластеризации используются алгоритмы ***k-means*** и **сети Кохонена**.

k-means (иногда называемый алгоритмом *k*-средних) – наиболее популярный метод кластеризации.

Он разбивает множество элементов векторного пространства на заранее известное число кластеров *k*. Затем случайным образом выбираются начальные центры («семена») кластеров. Для каждой записи исходной выборки определяется ближайший к ней центр кластера. Производится вычисление центроидов (центров масс векторов) как среднего для значений каждого признака. Центроид становится центром кластера.

Действие алгоритма таково, что он стремится минимизировать среднеквадратичное отклонение на точках каждого кластера от центроида:

$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} \left(x_j - \mu_i\right)^2$, где k – число кластеров, S_i – полученные кластеры;

μ_i – центры масс векторов S_i, $x_j \in S_i$.

Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на какой-то итерации не происходит изменения кластеров.

Сети Кохонена предназначены для решения задач кластеризации. Постановка задачи кластеризации. Дано: $X[n \times m]$, n – количество объектов, m – количество признаков; или Дано: множество векторов $X = \left\{X^1, X^2, ..., X^p, ..., X^n\right\}$, где $X^p = \left\{X_1^p, ..., X_m^p\right\}$.

Заранее известно количество k будущих кластеров, но само разбиение на кластеры не известно.

Необходимо: 1) найти k-ядер (центроидов) кластеров, т.е. вектор $C^k = \{C^1, C^2, ..., C^l, ..., C^k\}$, $C^l = \{C_1^l, ..., C_m^l\}$;

2) разбить множество векторов X на k кластеров: $X^p = \{X_1^p, ..., X_m^p\}$ на $K\{C^l\}$. Иными словами, задача заключается в том, чтобы найти некоторую функцию $l(p)$, которая позволяет определить номер кластера l по номеру входного объекта P, что и является решением задачи кластеризации. При этом эта функция должна удовлетворять следующему критерию: минимизации расстояний между объектами и ядрами кластеров. $D = \sum_{p=1}^{n} \sum_{l=1}^{k} d(X^p, C^l) \to \min$.

Алгоритм кластеризации:

1) задаются начальные значения ядер кластеров $\{C^l\}$.

Способы задания: а) случайными числами; б) некоторыми равными числами; в) эвристическими правилами (метод главных компонент), основанными на некоторой закономерности.

2) Фиксируются ядра кластеров $\{C^l\} = \text{const}$. Находим $l(p)$ – разбиение $\{X^p\}$ с целью $D \to \min$.

3) Корректируется $\{C^l\}$ таким образом, чтобы в пределах каждого кластера суммарное расстояние между объектом этого кластера и ядром было минимальным.

$$D_l = \sum_{p \in S_l} d(X^p, C^l) \to \min,$$

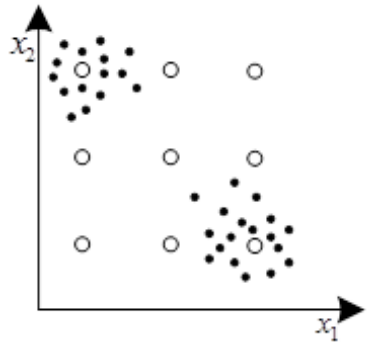
результат $\{C^l\}$.

Алгоритм обучения без учителя для сетей Кохонена

I. Инициализация весов:

1) малые случайные числа;

2) равномерная инициализация.



Для случая $n=k$ обучение не проводится. Для нормальной работы $n>>k$.

3) специальные методы инициализации.

II. Обучающая выборка: $X = \{X^p\} = \{X^1, X^2, ..., X^n\}$

III. Рассчитывается l_0 – прямой проход по сети.

IV. Корректировка весов сети Кохонена.

1) традиционный способ: корректируются веса только нейрона-победителя: $W_{l_0}^{\text{нов}} = W_{l_0}^{\text{стар}} + \alpha(X^p - W_{l_0}^{\text{стар}})$, где α – скорость обучения ($0<\alpha<1$) монотонно убывает.

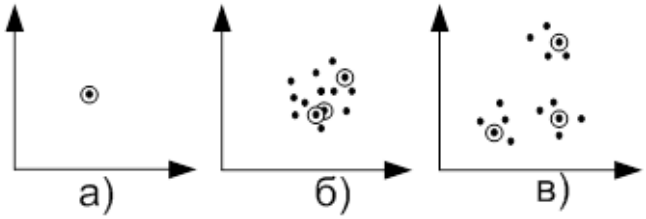
Смысл корректировки весов заключается в попытке приблизить координаты нейрона-победителя к входному образу вектора. Обучение продолжается до тех пор, пока координаты весов не перестанут изменяться. По окончанию обучения каждый нейрон отвечает за определение кластера.

2) метод выпуклой комбинации;

3) модифицированные алгоритмы.

В процессе обучения вектора расходятся от начальных точек к своим истинным значениям. При этом особенность обучения заключается в следующем: в процессе движения к своим истинным значениям обучающие вектора захватывают ядра кластеров пропорционально своей плотности распределения.

Графическая интерпретация:



Трудности при использовании алгоритма это выбор исходных центров кластеров и необходимость знать число кластеров заранее. Также алгоритм плохо работает в случае, когда один кластер значительно больше остальных и они находятся близко друг от друга. Возникает эффект «расщепления» большого кластера.

Самоорганизующаяся карта Кохонена строится на основе нейросети Кохонена.

Сеть Кохонена является одной из разновидностей соревновательной (конкурентной) нейронной сети с обучением без учителя. Ее основной целью является преобразование сложных многомерных данных в более простую структуру малой размерности. Сеть состоит из узлов, которые объединяются в кластеры.

Сигнал в сеть Кохонена поступает сразу на все нейроны, веса соответствующих синапсов интерпретируются как координаты положения узла, и выходной сигнал формируется по принципу «победитель забирает все» — то есть ненулевой выходной сигнал имеет нейрон, ближайший (в смысле весов синапсов) к подаваемому на вход объекту. В процессе обучения веса синапсов настраиваются таким образом, чтобы узлы решетки «располагались» в местах локальных сгущений данных, то есть описывали кластерную структуру облака данных; с другой стороны – связи между нейронами соответствуют отношениям соседства между соответствующими кластерами в пространстве признаков.

Карта Кохонена состоит из ячеек, центром каждой из которых является нейрон выходного слоя. В ячейку попадает один или несколько объектов, векторы весов которых оказываются ближе к вектору весов данного нейрона.

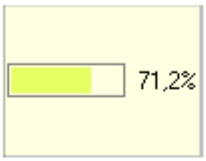
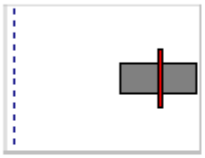
Изначально самоорганизующаяся карта представляет сетку из узлов, соединенных между собой. Кохонен рассматривал два варианта соединения узлов – в прямоугольную и гексагональную сетку. Отличие состоит в том, что в прямоугольной сетке каждый узел соединен с четырьмя соседними, а в гексагональной – с шестью ближайшими узлами. Шестиугольные ячейки более корректно отражают расстояния между объектами на карте, т.к. у них равны расстояния между центрами смежных ячеек.

Самоорганизующаяся карта Кохонена является методом проецирования многомерного пространства в пространство с более низкой размерностью (чаще всего двухмерное). Двухмерная карта имеет цветную раскраску. Интенсивность цвета в определенной точке зависит от данных, которые туда попали. Ячейки, в которые попали элементы с минимальными значениями или не попало вообще ни одной записи, окрашиваются синим цветом, а ячейки с максимальными значениями окрашиваются красным. Раскраска карты позволяет оценить и результаты кластеризации. Если ячейки с одинаковой расцветкой образуют обособленные области, то результаты кластеризации хорошие. Если ячейки разных цветов разбросаны вперемешку по всей карте, то результаты плохие.

Недостатком карты Кохонена является эвристический характер метода. Начальная инициализация карты, т.е. задание весов нейронов, является произвольной, что может привести к потере однозначности результата. Если обучать карту Кохонена несколько раз, то получаются непохожие итоги.

Кластеры отображают группы векторов, расстояние между которыми меньше, чем расстояние до соседних групп. Иными словами, все элементы карты, попавшие в область одного цвета (кластер), имеют сходные признаки.

Результаты кластеризации алгоритмом Кохонена можно увидеть не только на карте, но и на специальном визуализаторе *Профили кластеров*. Это кросс-таблица с двумя измерениями – к*ластеры* и *поля*. На их пересечении отображаются следующие показатели:

Показатель	Пример	Описание
Значимость		1 минус вероятность нулевой гипотезы. Значимость выражается в процентах. Для непрерывных полей используется *t*-критерий Стьюдента, а для дискретных полей – критерий хи-квадрат. Общая значимость поля определяется по F-критерию Фишера.
Доверительный интервал		Графическое изображение 95% доверительного интервала для среднего значения кластера (темно-серая область). Кроме этого, показываются: ▪ среднее значение по кластеру – красной линией; ▪ среднее значение по всей выборке – синей штрихпунктирной линией.
Среднее	–	Среднее значение по полю, рассчитанное для объектов, попавших в кластер.
Стандартное отклонение	–	Стандартное отклонение по полю, рассчитанное для объектов, попавших в кластер.
Стандартная ошибка	–	Стандартная ошибка по полю, рассчитанная для объектов, попавших в кластер.

Пример использования карт Кохонена для решения задачи сегментации данных по продажам некоторой компании на кластеры

В Deductor Studio сети и карты Кохонена реализованы в обработчике *Карта Кохонена*, где содержатся сам алгоритм Кохонена и специальный *визуализатор Карта Кохонена*.

Алгоритм Кохонена применяется к сети Кохонена, состоящей из ячеек, упорядоченных на плоскости. По умолчанию размер карты равен 16 х 12, что соответствует 192 ячейкам (нейронам).

Не существует строгих правил для выбора числа нейронов в карте Кохонена. Каждый раз аналитик должен сам определять (возможно, методом проб и ошибок) оптимальное количество нейронов, исходя из решаемой задачи и особенностей данных. Например, если разброс значений признаков в обучающей выборке сильный (то есть векторы объектов в пространстве признаков разрежены), то, возможно, следует уменьшить число нейронов карты, чтобы избе-

жать большого количества пустых ячеек. И наоборот, если векторы объектов расположены в пространстве признаков плотно, то для получения лучших результатов можно увеличить число нейронов.

Ограничения использования карт Кохонена в Deductor: в Deductor Studio алгоритм Кохонена ориентирован на работу преимущественно с числовыми типами данных, а также с упорядоченными (ординальными) типами. Обработка данных в полях, значения которых нельзя упорядочить, будет приводить к некорректным результатам. Упорядочивание ординальных типов осуществляется на вкладке *Настройка нормализации*.

Импортируем в Deductor набор данных из файла *Продажи 2*.

Запустим мастер обработки и выберем узел *Карта Кохонена*..

В качестве входных параметров используются:

- Товарная группа;
- Сумма;
- Количество.

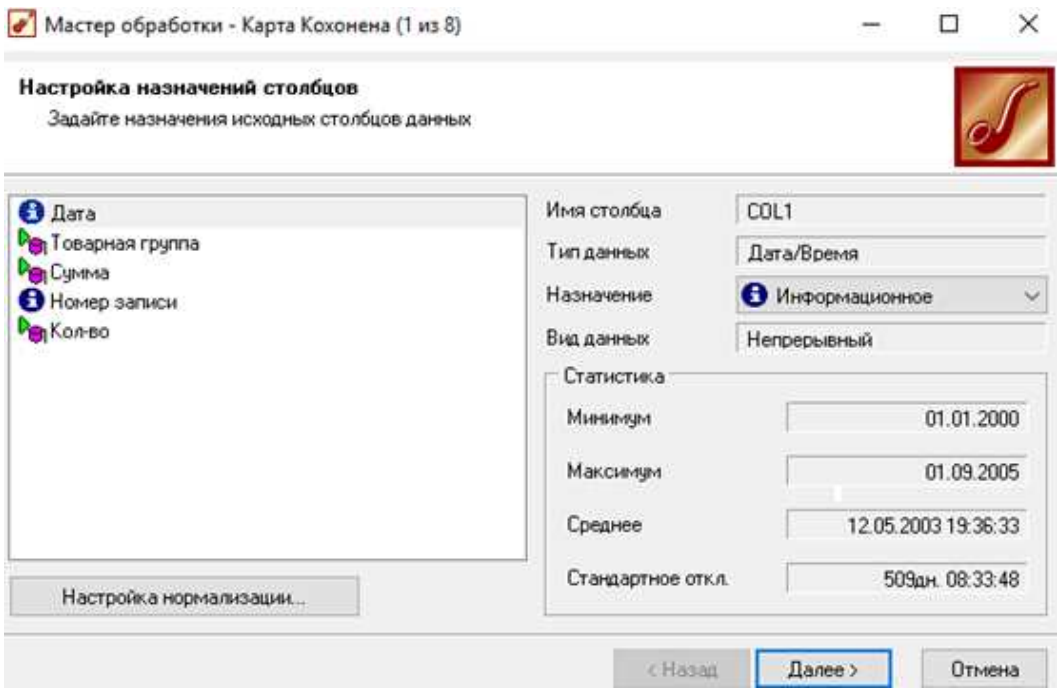


Рис. 66. Настройка назначений полей

На этой же вкладке при нажатии кнопки «Настройка нормализации» откроется окно, где можно задать значимость каждого входного поля. Оставим значимость одинаковой для всех полей без изменений.

Замечание. В обработчике «карта Кохонена» допускается задавать и выходное поле. Несмотря на такое название, это поле не будет принимать участие в алгоритме Кохонена, однако после построения по этому полю будет собрана статистика. Это открывает возможности для решения картой Кохонена задачи классификации или регрессии.

Затем разбиваем входящее множество на обучающее и тестовое. Поскольку любой метод кластеризации, в том числе и алгоритм Кохонена, субъективен, смысл в выделении отдельного, тестового множества, как правило, отсутствует. Оставим в обучающем множестве 100 % записей.

На третьей вкладке задаются размер и форма карты Кохонена (рис. 67). Согласимся с настройками по умолчанию – шестиугольные ячейки, размер 16x12.

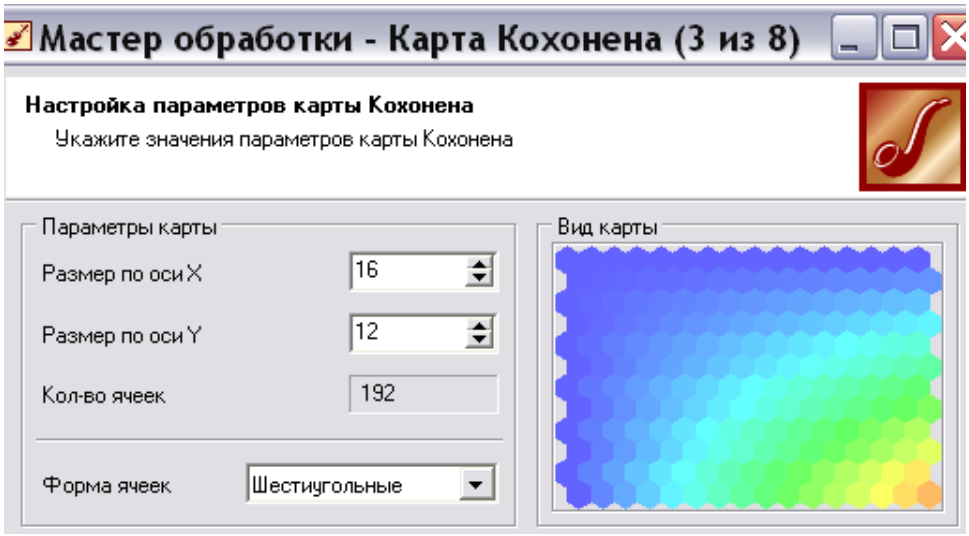


Рис. 67. Параметры будущей карты Кохонена

Замечание: Обычно строят несколько моделей с разными настройками. Например, карта с увеличенным масштабом 24x18 может оказаться лучше, так как позволит «разглядеть» кластер, который не удавалось обнаружить при размере карты 16x12. Поэтому здесь универсальных рецептов нет. Понять, лучше

или хуже карта Кохонена, можно, только сравнив ее с картами при других настройках, сравнив матрицы ошибок квантования и матрицы плотности попадания.

Важным этапом является настройка параметров остановки обучения. Устанавливаем параметры остановки обучения и устанавливаем значение эпохи, по достижению которой обучение будет прекращено.

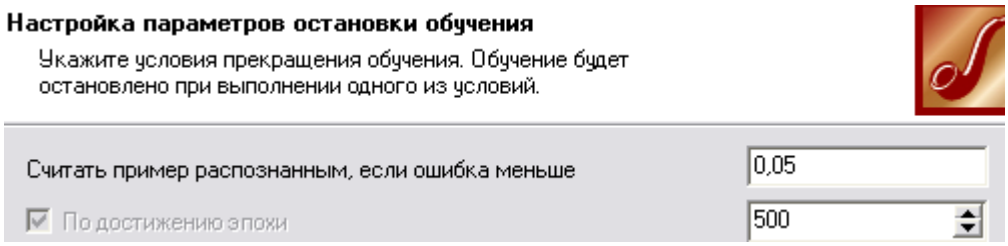


Рис. 68. Настройка параметров остановки обучения

Наконец, на последнем шаге, предшествующем обучению, настраиваются параметры обучения карты Кохонена (рис. 69).

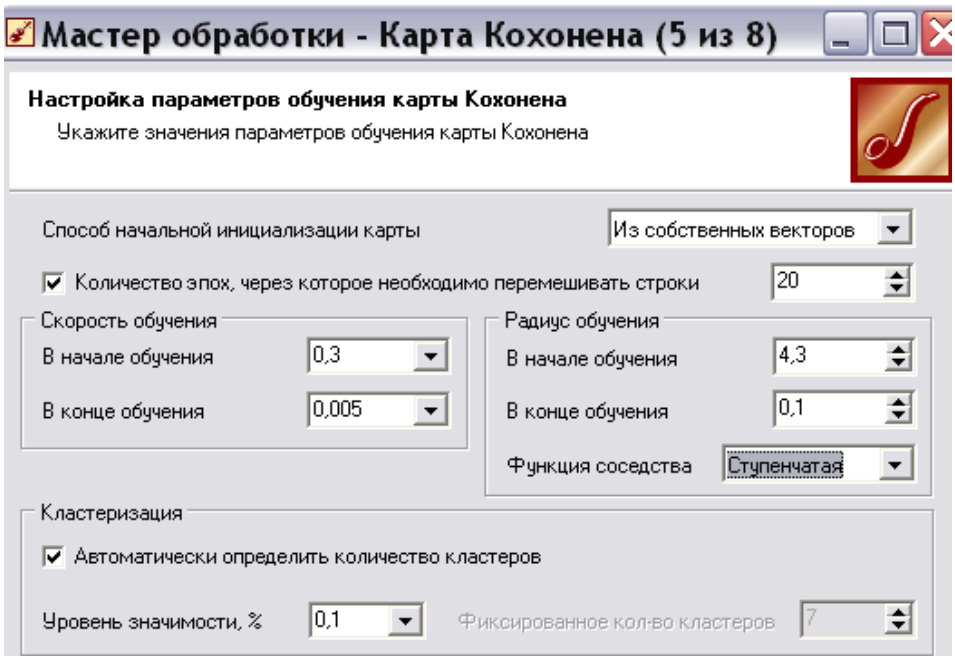


Рис. 69. Параметры обучения карты Кохонена

Здесь задаются следующие опции:

Способ начальной инициализации карты определяет, как будут установлены начальные веса нейронов карты. Удачно выбранный способ инициализации может существенно ускорить обучение и привести к получению более качественных результатов.

Доступны три варианта:

• *Случайными значениями* – начальные веса нейронов будут инициированы случайными значениями.

• *Из обучающего множества* – в качестве начальных весов будут использоваться случайные примеры из обучающего множества.

• *Из собственных векторов* – начальные веса нейронов карты будут проинициализированы значениями подмножества гиперплоскости, через которую проходят два главных собственных вектора матрицы ковариации входных значений обучающей выборки.

При выборе способа начальной инициализации (рис. 69) следует руководствоваться следующей информацией:

• объемом обучающей выборки;

• количеством эпох, отведенных для обучения;

• размером карты.

Между указанными параметрами и способом начальной инициализации существует много зависимостей. Выделим несколько главных.

1. Если объем обучающей выборки значительно (в 100 и более) превышает число ячеек карты и время обучения не играет первоочередной роли, то лучше выбрать *инициализацию случайными значениями*, так как это даст меньшую вероятность попадания в локальный минимум ошибки кластеризации.

2. Если объем обучающей выборки не очень велик, время обучения ограничено или необходимо уменьшить вероятность появления после обучения пустых ячеек, в которые не попало ни одного экземпляра обучающей выборки, то следует использовать *инициализацию примерами из обучающего множества*.

3. *Инициализацию из собственных векторов* можно использовать при любом стечении обстоятельств. Единственное замечание: вероятность появления пустых ячеек после обучения выше, чем при инициализации примерами из обучающего множества. Именно этот способ лучше выбирать при первом ознакомлении с данными.

Скорость обучения – задается скорость обучения в начале и в конце обучения карты Кохонена. Рекомендуемые значения: 0,1–0,3 в начале и 0,05–0,005 в конце обучения.

Радиус обучения – задается радиус обучения в начале и в конце обучения карты Кохонена. Радиус в начале должен быть достаточно большой – примерно половина или меньше размера карты (максимальное линейное расстояние от любого нейрона до другого любого нейрона), а в конце – достаточно малым, примерно 1 или меньше. Начальный радиус в Deductor подбирается автоматически в зависимости от размера карты.

В этом же блоке задается **Функция соседства**: Гауссова или Ступенчатая. Если функция соседства *Ступенчатая*, то «соседями» для нейрона-победителя будут считаться все нейроны, линейное расстояние до которых не больше текущего радиуса обучения. Если используется Гауссова функция соседства, то «соседями» для нейрона-победителя будут считаться все нейроны карты, но в разной степени полноты. При использовании Гауссовой функции соседства обучение проходит более плавно и равномерно, так как одновременно изменяются веса всех нейронов, что может дать немного лучший результат, чем если бы использовалась ступенчатая функция. Однако времени, необходимого на обучение, требуется немного большее по причине того, что на каждой эпохе корректируются все нейроны.

Кластеризация – в этой области указываются параметры алгоритма *k-means* (*G-means*), который запускается после алгоритма Кохонена для группировки ячеек карты. Здесь нужно только определить, позволить алгоритму автоматически определить число кластеров (*G-means*) или сразу зафиксировать его (*k-means*). Следует знать, что автоматически подбираемое число кластеров не всегда приводит к желаемому результату – число кластеров может предлагаться слишком большим, поэтому рассчитывать на эту опцию можно только на этапе исследования данных.

После окончания ввода параметров запускаем процесс обучения– необходимо нажать на кнопку **Пуск** и дождаться окончания процесса обучения. В открывшемся окне можно будет увидеть динамику процесса обучения карты Кохонена (рис. 70). По умолчанию алгоритм делает 500 итераций (эпох). Если предварительно установить флаг **Рестарт**, то веса нейронов будут проинициализированы согласно выбранному на предыдущем шаге способу инициализации, иначе обучение начнется с текущих весовых коэффициентов (это справедливо только при повторной настройке узла).

Рис. 70. Обучение карты Кохонена

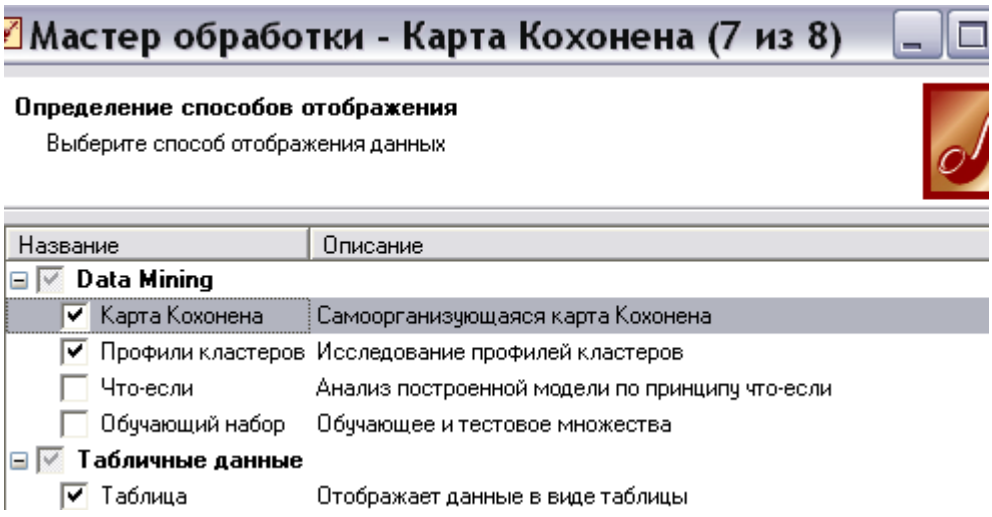


Рис. 71. Выбор способа отображения результатов кластеризации

К обученной сети Кохонена предлагаются специализированные визуализаторы – **Карта Кохонена** и **Профили кластеров** (рис. 71). Параметры карты задаются на специальной вкладке мастера (рис. 72).

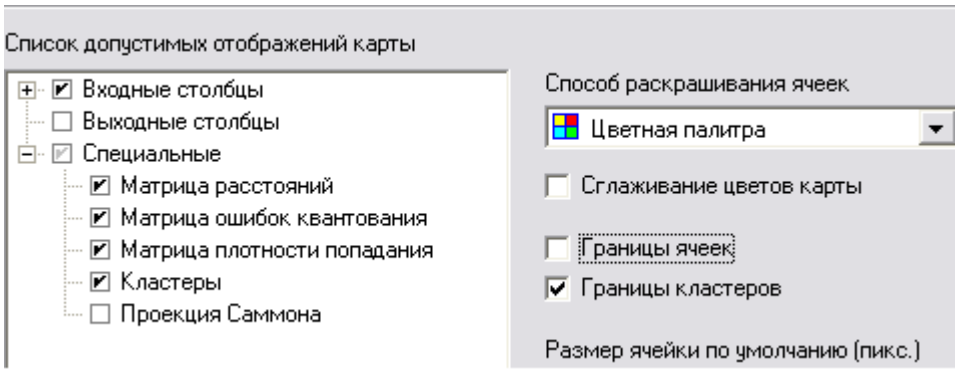


Рис. 72. Настройки визуализатора карты Кохонена

Список допустимых отображений карты содержит три группы – входные поля, выходные поля и специальные. Последние не связаны с каким-либо полем набора данных, а служат для анализа всей карты.

- *Матрица расстояний* применяется для визуализации структуры кластеров, полученных в результате обучения карты. Большое значение говорит о том, что данный нейрон сильно отличается от окружающих и относится к другому классу.
- *Матрица ошибок квантования* отображает среднее расстояние от расположения примеров до центра ячейки. Расстояние считается как евклидово расстояние. Матрица ошибок квантования показывает, насколько хорошо обучена сеть Кохонена. Чем меньше среднее расстояние до центра ячейки, тем ближе к ней расположены примеры и тем лучше модель.
- *Матрица плотности попадания* отображает количество объектов, попавших в ячейку.
- *Кластеры* – ячейки карты Кохонена, объединенные в кластеры алгоритмом *k-means*.
- *Проекция Саммона* – матрица, являющаяся результатом проецирования многомерных данных на плоскость.

Выбрав по окончании обучения в списке визуализаторов *карту Кохонена*, увидим, что в результате кластеризации получилось четыре кластера (рис. 73).

При анализе карт входов используем сразу несколько карт (это зарплата, расход, доход). Например, на одной из карт выделяем область с наибольшими значениями показателя (выделена красным цветом) и изучаем эти же нейроны на других картах. При работе с картой доступны операции, выполняемые с помощью кнопок на панели инструментов визуализатора или контекстного меню, вызываемого правой кнопкой мыши в любом окне карты.

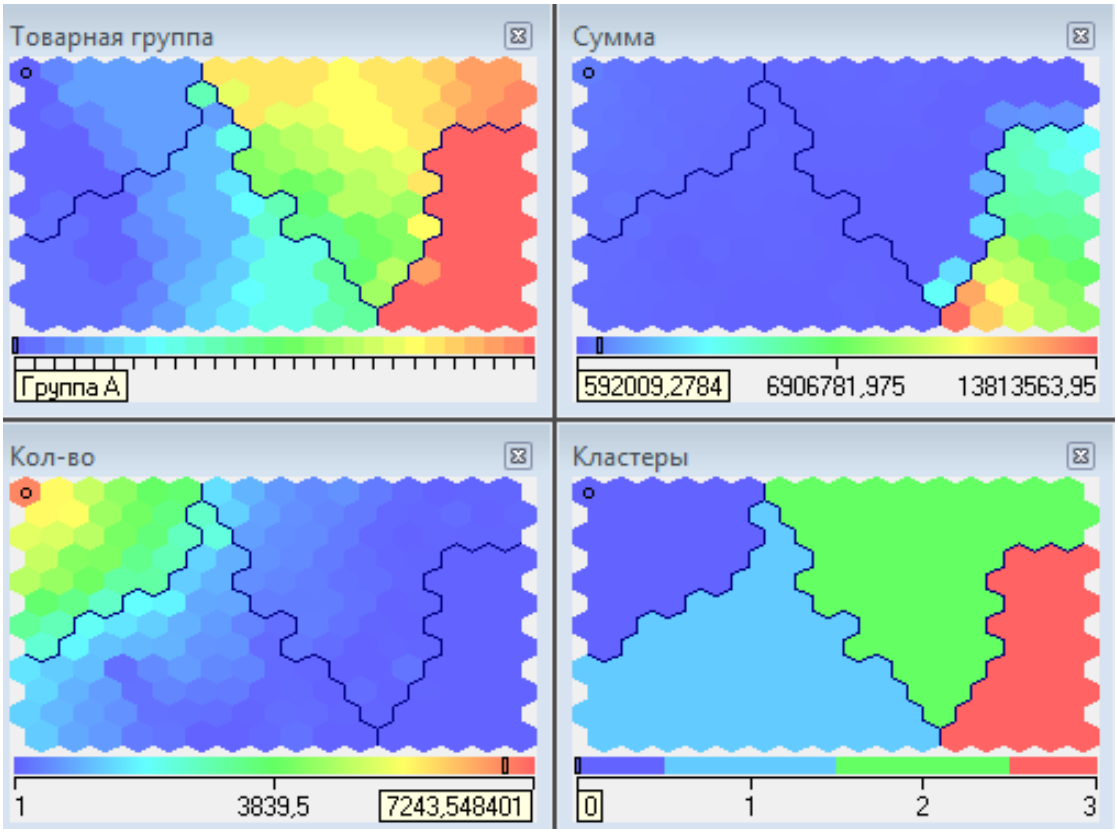


Рис.73. Карта Кохонена для сегментации продаж по товарным группам

Результат анализа раскраски карт соответствующих показателей и их статистических характеристик, используя визуализатор *Профили кластеров* (рис. 74) позволил дать каждому кластеру описание.

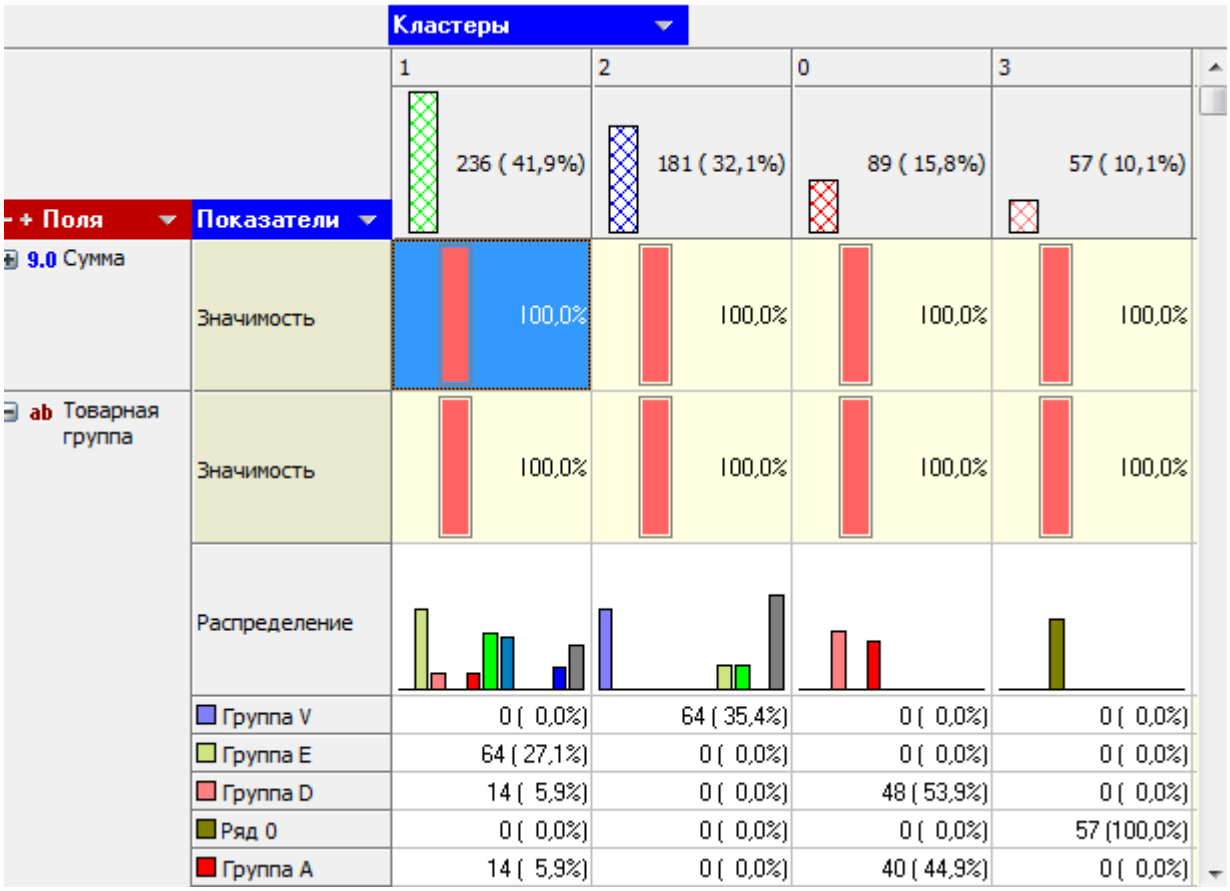


Рис. 74. Визуализатор *Профили кластеров*

Кластер	Средняя прибыль	Кол-во	Кол-во, %	Прибыль	Процент от общей прибыли
0	319573,2	89	15,8	28442014,8	6
1	79673	236	41,9	18802828	4
2	30329,5	181	32,1	5489639,5	2
3	6900086	57	10,1	393304924,8	88
Итого		563		446039407,1	100

Кластер 0. (Мощность 15,8 %). Содержит в основном товары товарных групп A и D, дают компании 6 % прибыли.

Кластер 1. (Мощность 41,9 %). Кластер, который содержит товары почти всех товарных групп, но преобладают товары групп B, E, I. Продажа товаров этого кластера приносит компании 4 % прибыли.

Кластер 2. (Мощность 32,1 %). В этом кластере присутствуют товары разных товарных групп, но преобладают товары группыV. Товары, входящие в данный кластер, приносят самую меньшую прибыль компании (1 %).

Кластер 3. (Мощность 10,1 %). Товарная группа, по которой заключаются самые крупные сделки в компании. Это чистый сегмент – нет товаров других групп. По кластеру было совершено немного продаж, но зато сделки были самые крупные, которые приносили максимальную прибыль компании (88 %).

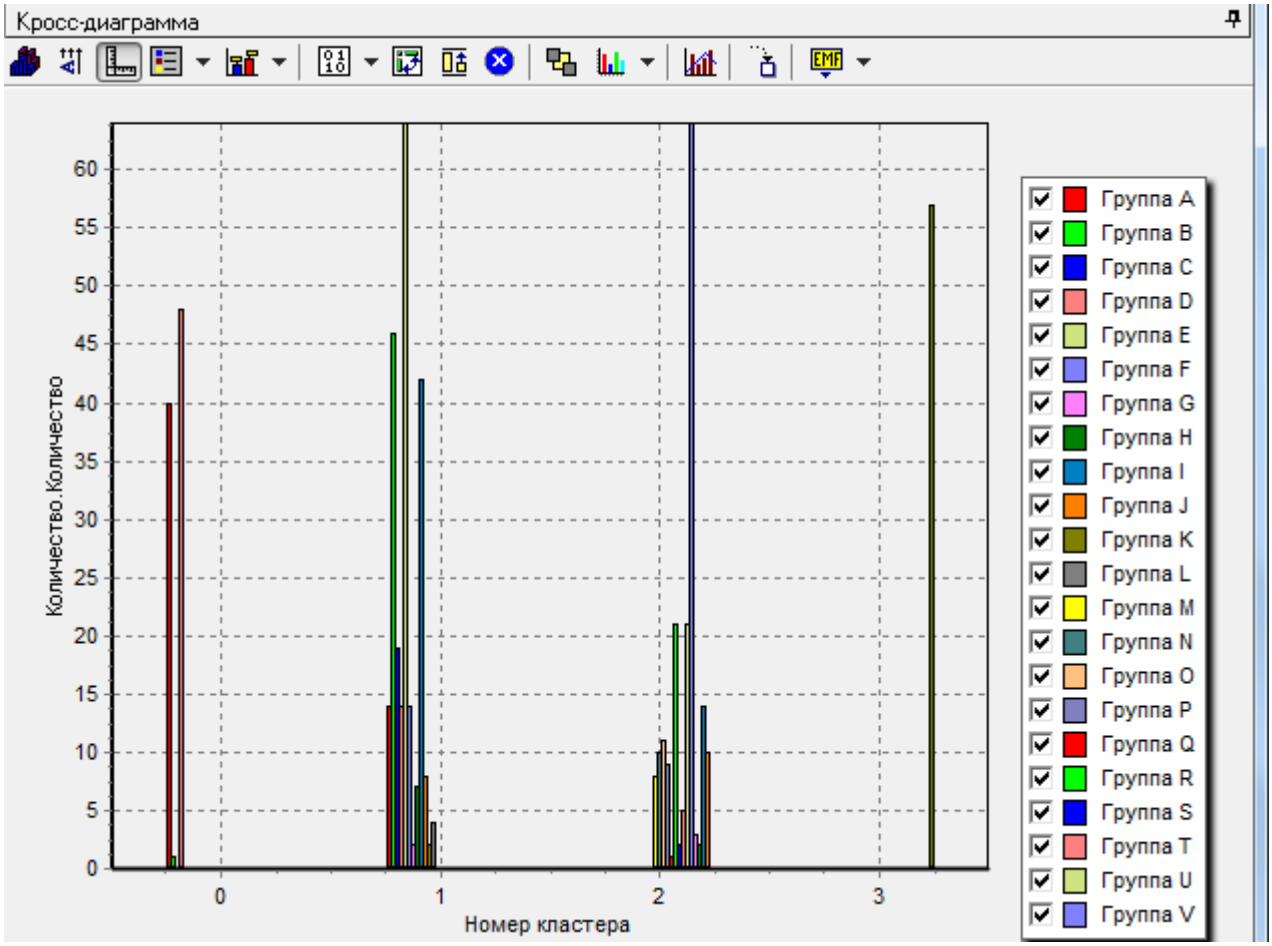


Рис. 75. Кросс-диаграмма (количество товаров по каждому кластеру)

Вопросы для самопроверки

1. Дайте определения ИИС и Data Mining.
2. Дайте определения задачам Data Mining (классификация, регрессия, кластеризация, ассоциативные правила).
3. Какие существуют алгоритмы Data Mining?
4. В чем суть модели Дерево решений?
5. Каким свойством деревьев решений обусловлена их высокая объясняющая способность?

6. Опишите алгоритм ID3.
7. Основные показатели качества модели ДР.
8. Основные показатели значимости правил модели ДР.
9. Что такое ассоциативные правила?
10. Как создаются ассоциативные правила?
11. Для чего используются ассоциативные правила при анализе данных?
12. Что такое достоверность правила?
13. Что такое поддержка правила?
14. Какие инструменты для построения ассоциативных правил имеются в системе Deductor?
15. Что такое Дерево правил?
16. Какие варианты создания Дерева правил существуют в Deductore?
17. Приведите пример полученных результатов анализа данных с помощью ассоциативных правил.
18. Способы машинного обучения: обучение с учителем и без учителя. Методы формирования тестовой и обучающей выборки.
19. Что представляет искусственная нейронная сеть?
20. Дайте определение искусственного нейрона.
21. Какая операция выполняется в теле нейрона над сигналами, поступающими по входным связям?
22. Перечислите и поясните применяемые виды активационных функций.
23. В чем заключается процесс обучения нейронной сети?
24. Что называют многослойным персептроном?
25. Для каких моделей используются таблица сопряженности и диаграмма рассеяния и как с их помощью оценить точность модели?
26. Data Mining: задача кластеризации. Методы кластерного анализа. (метод *k -средних*).
27. Data Mining: описание модели Карта Кохонена.

Задания для самостоятельной работы

Задание 1.

Выберите лучший атрибут для разбиения по алгоритму ID3.

Номер записи	Количество комнат	Тип дома	Минут ходьбы до остановки	Балкон / лод-жия	Категория
1	2	кирпич	далеко	0	дешево
2	2	кирпич	далеко	1	дешево
3	3	панель	далеко	1	дешево
4	4	кирпич	близко	1	дорого
5	3	панель	близко	0	дорого
6	3	кирпич	далеко	1	дорого
7	3	кирпич	далеко	1	дорого
8	2	панель	близко	0	дешево
9	2	панель	близко	1	дешево
10	4	панель	далеко	0	дорого

Задание 2. Классификация на основе Дерева решений

Разделить все районы Нижегородского региона на различные классы по уровню дохода бюджета при помощи инструментов Квантование и Дерево решений (данные взять из файла *показатели.txt* или из созданного ранее ХД *Регион*).

Для этого:

а) Нужно найти средние значения показателей по каждому району за весь исследуемый период;

б) Значения поля «доход бюджета» при помощи обработчика «Квантование» нужно разбить на три диапазона «низкий доход», «средний доход», «высокий доход».

в) С помощью обработчика «Дерево решений» получить правила, применяя которые можно определить к какому их трех возможных уровней дохода будет относиться произвольный район.

г) Оценить качество построенной классификационной модели по таблице сопряженности и соответствующей ей диаграмме.

Задание 3. Классификация на основе Дерева решений

1) Построить классифицирующее *Дерево решений* для отнесения водных объектов на основе показателя ИЗВ (индекс загрязнения воды) к определенному классу вод, используя критерии, описанные в таблице.

Классы качества вод в зависимости от значения ИЗВ

Значение ИЗВ	Воды
до 0,2	Очень чистые
0,2 – 1,0	Чистые
1,0 – 2,0	Умеренно загрязненные
2,0 – 4,0	Загрязненные
4,0 – 6,0	Грязные
6,0 – 10,0	Очень грязные

2) Результаты классификации отобразить на диаграмме «Процентное соотношение качества вод региона» (рис. 77). Ответить на вопрос: какой процент водных объектов Нижегородской области относится к классу Загрязненных вод?

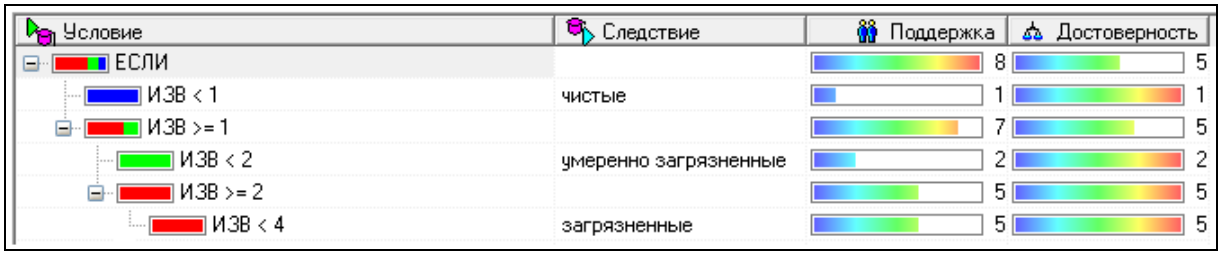


Рис. 76. Дерево решений

Рис. 77. Диаграмма «Процентное соотношение качества вод региона»

Задание 4

Дана небольшая база

T01	Сливы, салат, помидоры
T02	Сельдерей, конфеты
T03	Конфеты
T04	Яблоки, морковь, помидоры, картофель, конфеты
T05	Яблоки, апельсины, салат, конфеты, помидоры
T06	Персики, апельсины, сельдерей, помидоры
T07	Фасоль, салат, помидоры
T08	Апельсины, салат, морковь, помидоры, конфеты
T09	Яблоки, бананы, сливы, морковь, помидоры, лук, конфеты
T010	Яблоки, картофель

1) Приняв пороговое значение поддержки, равное 35 %, найдите популярные трехпредметные наборы.

2) Рассчитать показатели: поддержка (*S*), достоверность (*C*), лифт (*L*), леверидж (Рычаг) (*T*), улучшение (*I*) для правил: а) салат → помидоры; б) конфеты → помидоры.

Задание 5

Дана небольшая база

1	*a, b, c, d, e*
2	*a, b, c*
3	*a, c, d, e*
4	*b, c, d, e*
5	*b, c*
6	*b, d, e*
7	*c, d, e*

1) Найти ассоциативные правила, используя метод a priori (порог=4). Выявить значимые правила (поддержка ≥ 20%, достоверность ≥ 80%).

2) Рассчитать показатели: поддержка (*S*), достоверность (*C*), лифт (*L*), леверидж (Рычаг) (*T*), улучшение (*I*) для всех наборов.

3) Построить FP – дерево.

Задание 6. Генерация ассоциативных правил в АП Deductor. Интерпретация правил

1. Загрузить данные по вариантам.

2. Оставить настройки параметров построения ассоциативных правил по умолчанию:

Поддержка: 1 %< *S* <20 %

Достоверность: 40 %< *S* <90 %

Записать:

Количество популярных наборов =

Количество популярных наборов, удовлетворяющих поддержке >6 % =

Количество правил =

Товары – лидеры продаж (имеющие поддержку в нашей задаче >10 %):

Указание: используйте фильтр в визуализаторе «Популярные наборы»

Тривиальные правила (включающие лидеры продаж):

Тривиальные правила (экспертное мнение):

Полезные правила:

Непонятные правила:

3. Изменить настройки параметров построения ассоциативных правил

Поддержка: 1 %< *S* <20 %

Допустимая достоверность: 25 %< *S* <40 %

Записать:

Количество популярных наборов =

Количество правил =

Тривиальные правила (включающие лидеры продаж):

Тривиальные правила (экспертное мнение):

Полезные правила:

Непонятные правила:

Для полезных правил найти и проанализировать показатели значимости: *S, C, L, T, I.*

4. Изменить настройки параметров построения ассоциативных правил

Поддержка: 1 %< *S* < 20%

Допустимая достоверность: 1 %< *S* < 30%

Записать:

Количество популярных наборов =

Количество правил =

Тривиальные правила (включающие лидеры продаж):

Тривиальные правила (экспертное мнение):

Полезные правила:

Непонятные правила:

Для полезных правил найти и проанализировать показатели значимости: *S, C, L, T, I.*

5. Самостоятельно изменить настройки параметров построения ассоциативных правил, чтобы уменьшить (или увеличить) количество правил. Выписать интересные полезные правила.

6. Сделать вывод: как полезные правила применять на практике.

Задание 7

Построить нейронную сеть, позволяющую аппроксимировать заданную многомерную нелинейную функцию:

- Подготовить обучающую выборку средствами приложения Microsoft Excel и оформить ее в виде текстового файла с разделителями.

Рекомендации: Чтобы создать набор случайных чисел, нужно использовать функцию Excel СЛЧИС(). Затем случайные числа следует перевести в нужный диапазон и рассчитать значение заданной функции в соседнем столбце.

- Провести обучение нескольких нейронных сетей (с различной архитектурой) с помощью Deductor по алгоритму обратного распространения;
- Проверить качество каждой обученной сети с помощью диаграммы рассеяния.

Выбрать наилучшую модель и оценить точность аппроксимации.

Общее задание: $f = x_1 \cdot x_2$ (использовать готовый файл *multi.txt*).

Индивидуальные задания:

1. $f = \frac{x_1 + x_2}{x_3} + x_4 \cdot x_5$

2. $f = x_1 - 20\sin(x_2) + 5x_3 + \frac{x_4}{e^{x_5}}$

3. $f = \frac{x_1 \cdot x_2^2}{\sqrt{x_3}} + \sin(x_4 \cdot x_5)$

4. $f = x_1 - x_2 - x_3 + x_4 \cdot x_5^2$

5. $f = 0{,}5\cos(x_1 + x_2)^2 + \frac{1}{(x_3 + x_4)^2} + x_5$

6. $f = 5x_1 + \cos(x_2 + \sqrt{x_3}) + \sin(x_4 + \frac{x_5}{2})$

7. $f = 3\cos(x_1 \cdot x_2) + 2\sin(x_3) + \ln x_4 + 10x_5^2$

8. $f = \sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2}$

9. $f = x_1 + 2\cos(x_2) + x_3^2 + \sqrt{x_4} + \sin(x_5)$

10. $f = \frac{x_1 \cdot x_2^2}{\sqrt{x_3}} + \cos(x_4 \cdot x_5)$

11. $f = x_1 - 20\cos(x_2) + 5x_3 + \frac{x_4}{e^{x_5}}$

12. $f = 5x_1 + \sin(x_2 + \sqrt{x_3}) + \cos(x_4 + \frac{x_5}{2})$

13. $f = \sqrt{x_1^3 + x_2^3 + x_3^3 + x_4^3 + x_5^3}$

Задание 8. Решение задачи классификации и прогнозирования с помощью нейросетевого моделирования

1. Импортировать данные из файла *Риэлтор.txt*.

2. В графическом экране отсортировать по убыванию поле *Цена*.

3. Используя OLAP, определить характеристики 10 самых дорогих квартир.

4. Выполнить предобработку данных с помощью обработчика *Парциальная обработка* (по данным поля *Цена* – удалить аномалии).

5. Выполнить *корреляционный анализ* (для определения незначащих факторов): на входе: *все факторы, кроме номера примера*; На выходе –*цена*.

- Какие из признаков более значимые, какие – нет?

6. Выполнить *квантование* данных по полю *Цена* по трем интервалам.

7. Применяя обработчик *Замена данных*, замените названия интервалов на *дешевые, средние, дорогие.*

8. Построить классификатор на основе нейронной сети для оценки недвижимости, относящий объекты недвижимости на основе их признаков к одному из трех классов «дорогие», «средние», «дешевые» квартиры.

9. Для оценки качества модели постройте *таблицу сопряженности* и диаграмму «Качество классификации».

10. Используя обработчик «Нейросеть», постройте несколько моделей прогнозирования цены с разной конфигурацией (поле «Номер примера» – информационное, все – на вход, а цена – на выход). Теперь поле *Цена* обязательно должно быть непрерывным.

11. Сколько весовых связей имеет нейронная сеть? Рассчитайте максимально возможное число нейронов скрытого слоя.

12. В ходе проектирования нейросетевой модели выявите оптимальную конфигурацию нейронной сети, используя визуализаторы оценки качества модели.

13. Определить оптимальную структуру нейронной сети с точки зрения минимизации среднеквадратической ошибки обучения.

14. Изучить графические зависимости среднеквадратической ошибки обучения от количества нейронов, используемых в скрытых слоях, и от количества итераций, используемых для обучения.

15. При помощи визуализатора «Что если» проверьте, как работает построенный нейросетевой классификатор и лучшая нейросеть прогноза цены.

Задание 9. По прогнозированию отклика клиентов на массовую рассылку с помощью нейросетей

Торговая компания, осуществляющая продажу товаров, располагает информацией о своих клиентах и их покупках. Компания провела рекламную рассылку 13 504 клиентам и получила отклик в 14,5 % случаев. Необходимо построить модели отклика и проанализировать результаты, чтобы предложить способы минимизации издержек на новые почтовые рассылки. Данные находятся в файлах *responses1.txt* и *responses2.txt*, представляют собой таблицу со следующими полями:

Поля наборов данных «Отклики» (*responses1.txt и responses2.txt*)

N	Поле	Описание	Тип
1	Код клиента	Уникальный идентификатор	целый
2	Пол	Пол клиента	строковый
3	Сколько лет клиенту	Число лет с момента первой покупки. Если менее года, то в поле стоит 0	целый
4	Количество позиций товаров	Сколько уникальных товаров приобретал клиент	целый
5	Доход с клиента, тыс. ед.	Суммарная стоимость всех заказов клиента	вещест.
6	Число покупок в тек. году	Сколько раз клиент делал заказ в текущем году	целый
7	Обращений в службу поддержки	Сколько раз клиент обращался в службу поддержки	целый
8	Задержки платежей	Задержки клиента фиксируются, когда длительное время после заказа оплата не поступает	целый
9	Дисконтная карта	Является ли клиент участником дисконтных программ, дающих право на скидки	целый
10	Возраст	Возраст клиента	целый
11	Отклик	Отклик клиента на последнюю рассылку.	целый

		Значение «1» означает, что клиент совершил покупку после прямой адресной рассылки.	
12	Дата отклика	Информационное поле. Пустое, если отклика не было.	дата

Имеется также следующая информация:

- расходы на одну рассылку CM = 1,0 ед.;
- издержки на обслуживание клиента CR = 9,0 ед.;
- ожидаемая выручка с 1 заказа R = 20,0 ед.

Доход-издержки в массовой рассылке

Исход	Предсказано моделью	Фактически	Доход, ед.	Комментарий
TN	Нет отклика	Нет отклика	0	Нет контакта, нет издержек.
TP	Есть отклик	Есть отклик	(R –CM–CR) = 10,0	Ожидаемая выручка минус расходы на обслуживание и рассылку
FN	Нет отклика	Есть отклик	0	Нет контакта, нет издержек
FP	Есть отклик	Нет отклика	CM = –1,0	Это издержки: печать, упаковка и доставка по почте

Доход модели «разослать всем»: (781×10,0 – 4621×1,0) = 3189 ед.

Создать следующий сценарий в Deductor:

Указание: 1) Архитектура НС: входной слой – 9 нейронов (код клиента – информационное поле, дата отклика – неиспользуемое поле, отклик – выход модели, все остальные поля – входные; 1-й внутренний слой с 4 нейронами, выходной слой с 1 нейроном (Поле Отклик).

2) При расчете издержек в узле Калькулятор нужно ввести формулы:

TP=(RESPONSE=1 AND RESPONSE_OUT=1)*10

FP=(RESPONSE=0 AND RESPONSE_OUT=1)*1

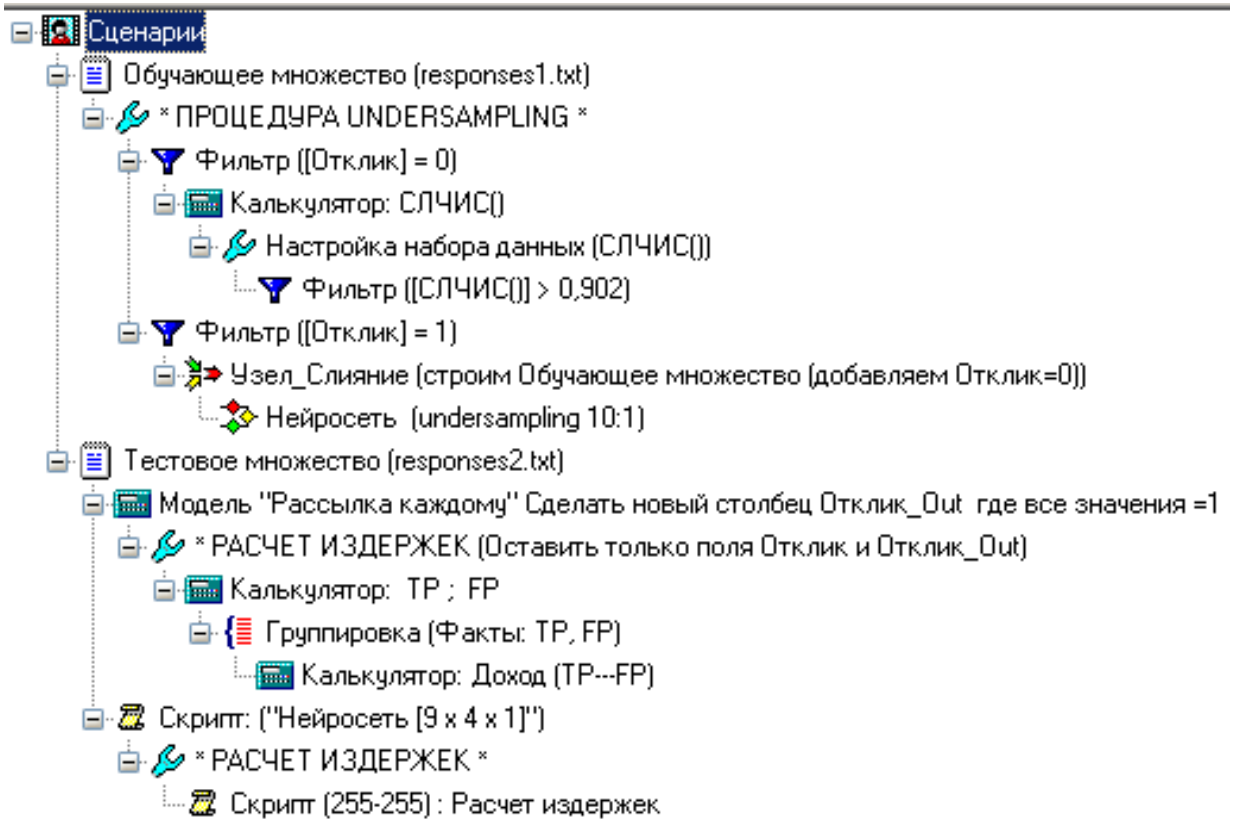


Рис. 78. Образец сценария в АП Deductor для задания 9

Сделайте вывод о качестве модели НС (используя таблицы сопряженности и прибыльность).

Граф нейросети | Что-если | Таблица сопряженности

Отклик

	Классифицировано		
Фактически	0	1	Итого
0	434	225	659
1		1171	1171
Итого	434	1396	1830

Рис. 79. Таблица сопряженности для оценки качества НС на обучающем множестве

Таблица сопряженности

Отклик

	Классифицировано		
Фактически	0	1	Итого
0	3120	1501	4621
1	4	777	781
Итого	3124	2278	5402

Рис. 80. Таблица сопряженности для оценки качества НС на тестовом множестве

Задание 10

1. Загрузите данные по кредитованию.
2. Постройте многомерный отчет и кросс-диаграмму распределения заемщиков: а) по возрастным группам; б) по целям кредитования;
   в) по целям кредитования и полу заемщика;
   г) по целям кредитования и должностям.
3. Постройте многомерный отчет и кросс-диаграмму возрастных групп, на которые приходится 50 % выдаваемых кредитов.
4. С помощью самоорганизующихся карт произвести сегментацию заемщиков на кластеры. Дайте каждому сегменту заемщиков название и проинтерпретируйте результаты сегментации.
5. Постройте несколько моделей деревьев решений при различных настройках, а также отдельные деревья решений для заемщиков, состоящих и не состоящий в браке. Выберите модель, которая чаще отказывает в выдаче кредита.
6. Сравните качество полученных моделей и сформулируйте рекомендации по их выбору при различной кредитной политике.

4. Адаптивные системы с генетическими алгоритмами

Генети́ческий алгори́тм (англ. *genetic algorithm*) – эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путем случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как наследование, мутации, отбор и кроссинговер. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

Искусственная эволюция стала общепризнанным методом оптимизации после работы Инго Рехенсберга и Ханса-Пауля Швефеля в 1960-х и начале 1970 -х годов двадцатого века – группа Рехенсберга смогла решить сложные инженерные проблемы согласно стратегиям эволюции. Другим подходом была техника эволюционного программирования Лоренса Дж. Фогеля, которая была предложена для создания искусственного интеллекта. Генетические алгоритмы стали особенно популярны благодаря работе Джона Холланда (Holland, 1975) и его книге «Адаптация в естественных и искусственных системах». Его исследование основывалось на экспериментах с клеточными автоматами, проводившимися Холландом и на его трудах, написанных в университете Мичигана. Холланд ввел формализованный подход для предсказывания качества следующего поколения, известный как теорема схем. Исследования в области генетических алгоритмов оставались в основном теоретическими до середины 80-х годов, когда была, наконец, проведена Первая международная конференция по генетическим алгоритмам в Питтсбурге, Пенсильвания (США). С ростом исследовательского интереса существенно выросла и вычислительная мощь настольных компьютеров, это позволило использовать новую вычислительную технику на практике. В конце 80-х компания General Electric начала продажу первого в мире продукта, работавшего с использованием генетического алгоритма. Им стал набор промышленных вычислительных средств. В 1989 г. другая компания Axcelis, Inc. выпустила Evolver – первый в мире коммерческий продукт на генетическом алгоритме для настольных компьютеров.

Основные понятия генетических алгоритмов

1) Особь (генотип, индивидуум, структура) – вариант решения задачи в закодированном виде. Иначе, это точка в многомерном пространстве для оптимизируемой функции $f(x_1,x_2,\ldots,x_n) \rightarrow \max$.

2) Популяция – множество особей, образованное на i-м шаге выполнения.

3) Поколение – очередная популяция на i-м шаге.

4) Хромосома – вектор из нулей и единиц (или других значений).

5) Ген – одна позиция (бит) хромосомы.

6) Аллель – значение гена.

7) Локус – позиция (номер гена) в хромосоме

8) Фенотип – множество декодированных значений, соответствующих генотипу.

9) Кроссовер – операция, при которой хромосомы обмениваются частями.

10) Мутация – случайное изменение одной (нескольких) позиций в хромосоме.

11) Функция приспособленности (fitness function) – функция оценки, которая определяет меру приспособленности данной особи в популяции и позволяет выбрать из множества особей наиболее приспособленную в соответствии с эволюционным принципом выживания сильнейших.

Виды функций приспособленности:

- В задачах оптимизации ФП – сама целевая функция. При этом если отыскивается максимум, то функция не модифицируется. Если же отыскивается минимум, то функция приспособления модифицируется так, чтобы опять свести задачу к максимизации.
- В задачах теории управления ФП – это функция ошибки ε.
- В теории игр ФП – это стоимостная функция.

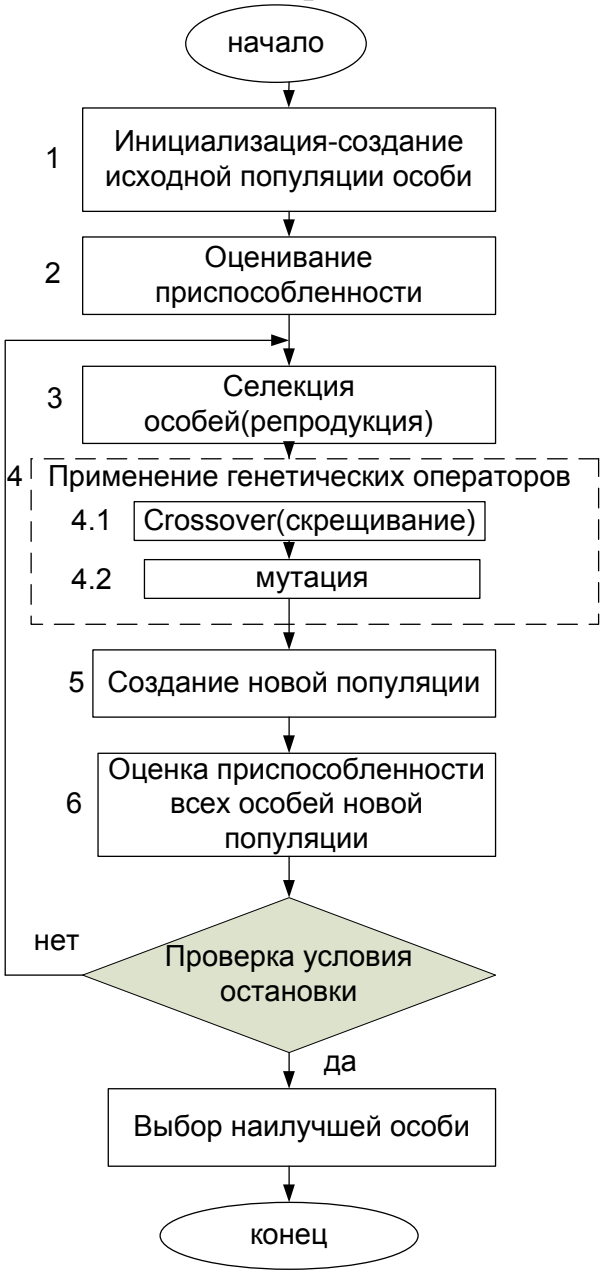
Генетические Алгоритмы (ГА) – это адаптивные методы функциональной оптимизации, основанные на компьютерном имитационном моделировании биологической эволюции по теории Дарвина.

В основе модели эволюции Дарвина лежат случайные изменения отдельных материальных элементов живого организма при переходе от поколения к поколению. Целесообразные изменения, которые облегчают выживание и производство потомков в данной конкретной внешней среде, сохраняются и передаются потомству, т.е. наследуются. Особи, не имеющие соответствующих приспособлений, погибают, не оставив потомства или оставив его меньше, чем приспособленные. Поэтому в результате естественного отбора возникает попу-

ляция из наиболее приспособленных особей, которая может стать основой нового вида.

Каждый конкретный генетический алгоритм представляет имитационную модель некоторой определенной теории биологической эволюции или ее варианта. Работа ГА представляет собой итерационный процесс, который продолжается до тех пор, пока поколения не перестанут существенно отличаться друг от друга, или не пройдет заданное количество поколений или заданное время. Для каждого поколения реализуются отбор, скрещивание и мутация.

Блок-схема классического генетического алгоритма



Пример (Этап инициализации)

Пусть нужно максимизировать следующую функцию $f(x)= -x^2+14x+1$ на отрезке [0;15].

Проведем инициализацию. Выберем размер популяции N=4. Произвольно берем 4 точки на отрезке. Вычисляем значение функции в них и переводим в двоичный вид: Ch1*=f(3)=34 Ch1= [100010]; Ch2*=f(7)=50 Ch2= [110010]

Ch3*=f(14)=1 Ch3= [000001] Ch4*=f(5)=46 Ch4= [101110].

Таким образом, мы получили первоначальную популяцию.

Селекция хромосом. Метод «рулетки»

Как видно из блок-схемы на 3-м этапе генетического алгоритма, происходит селекция особей. Наиболее распространенным методом селекции является метод «рулетки»:

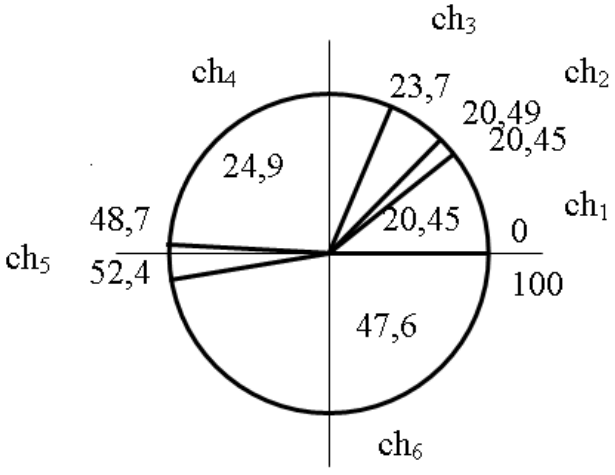
Формируется круг, площадь которого равна сумме всех значений приспособленности особей. Этот круг делится на сегменты, где каждая часть (вероятность селекции *i*-й хромосомы) определяется по формуле:

$$S_{\text{сегм}} = \frac{F(ch_i^*)}{\sum F(ch_i^*)} * 100\,\% .$$

Очевидно, что чем больше сектор, тем больше вероятность победы соответствующей хромосомы, и соответственно в среднем функция приспособленности от поколения к поколению будет возрастать.

Пример

Вращаем колесо рулетки 6 раз. Выпадают числа от 0 до 100. Пусть выпали следующие числа: 97, 26, 54, 13, 31, 88.
Идентифицируем, в какой сектор попали эти числа, то есть какие хромосомы участвуют в скрещивании: ch_6, ch_4, ch_6, ch_1, ch_4, ch_6.



Применение генетических операторов

К генетическим операторам относятся Crossover и мутация.

Crossover – операция скрещивания хромосом, при котором хромосомы обмениваются своими частями.

1. Генерируются пары случайным образом.

2. Для каждой пары хромосом подбирается локус случайным образом.

3. Производится обмен частями хромосом между двумя родителями.

Пример. Скрещивание

Родитель 1	**100**11
Родитель 2	101**01**
Потомок 1	10001
Потомок 2	10111

Мутация – случайное изменение одной или нескольких позиций в хромосоме. Оператор мутации применяется с определенной вероятностью p_m к особям популяции. 101**1**1 $\rightarrow$ 101**0**1

Проверка условия остановки ГА

Итерации повторяются до тех пор, пока не будет выполняться условие остановки.

Некоторые из возможных условий остановки

1. По времени.

2. По количеству итераций.

3. По отсутствию улучшения функции приспособленности.

4. По достижению максимума (если он известен).

Простой генетический алгоритм

Согласно репродуктивному плану Холланда генетические схемы поиска оптимальных решений включают следующие этапы процесса эволюции:

1. Конструируется начальная популяция. Вводится начальная точка отсчета поколений $t = 0$. Вычисляются приспособленность хромосом популяции (целевая функция) и средняя приспособленность всей популяции.

2. Устанавливается значение $t := t+1$. выбираются два родителя (хромосо-

мы) для кроссинговера. Выбор осуществляется случайным образом пропорционально жизнеспособности хромосом, которая характеризуется значениями целевой функции.

3. Формируется генотип потомка. Для этого с заданной вероятностью над генотипами выбранных хромосом производится операция кроссинговера. Случайным образом выбирается один из потомков *A(t)*, который сохраняется как член новой популяции. Далее к потомку *A(t)* последовательно с заданными вероятностями применяются операторы инверсии и мутации. Полученный в результате генотип потомка сохраняется как *A(t)*.

4. Обновление текущей популяции путём замены случайно выбранной хромосомы на *A(t)*.

5. Определение приспособленности *A(t)* и пересчет средней приспособленности популяции.

6. Если $t=T$, где T – заданное число шагов, то переход к этапу 7, в противном случае – переход к этапу 2.

7. Конец работы.

Основная идея эволюции, заложенная в различные конструкции генетических алгоритмов, проявляется в способности «лучших» хромосом оказывать большее влияние на состав новой популяции за счет длительного выживания и более многочисленного потомства.

Простой генетический алгоритм включает операцию случайной генерации начальной популяции хромосом и ряд операторов, обеспечивающих генерацию новых популяций на основе начальной. Этими операторами являются репродукция, кроссинговер и мутация.

Репродукцией называется процесс копирования хромосом с учетом значений целевой функции, т.е. хромосомы с «лучшими» значениями целевой функции имеют большую вероятность попадания в следующую популяцию. Этот процесс является аналогией митозного деления клеток (хромосом), для репродукции проводится в соответствии принципом «выживания сильнейшего».

Простейшим способом представления операции репродукции в алгоритмической форме является колесо «рулетки», в котором каждая хромосома имеет поле, пропорциональное значению целевой функции.

Генетический алгоритм Девиса включает следующие шаги:

1. Инициализация популяции хромосом.

2. Оценка каждой хромосомы в популяции.

3. Создание новых хромосом посредством изменения и скрещивания текущих хромосом (применение операторов мутации и кроссинговера).

4. Устранение хромосом из популяции для замены их новыми.

5. Оценка новых хромосом и включение их в популяцию.

6. Проверка условия исчерпания ресурса времени, отведенного на поиск оптимального решения (если время исчерпано, то работа алгоритма завершается и производится возврат к наилучшей хромосоме, в противном случае – переход к шагу 3).

Холланд предложил для генетического алгоритма оператор инверсии, который реализуется по схеме:

1. Стринг (хромосома) $B=(b_1,b_2,...,b_n)$ выбирается случайным образом из текущей популяции.

2. Из множества $Y= (0,1,2,\ldots, L+1)$ случайным образом выбираются два числа y_1 и y_2 и определяются значения $x_1=\min(y_1, y_2)$.

3. Из хромосомы B формируется новая хромосома путем инверсии (обратного порядка) сегмента, лежащего справа от позиции x_2 в хромосоме B. После применения оператора инверсии строка B примет вид $B = (b_1,...,bx_1, bx_2\text{-}1, bx_2\text{-}2, \ldots,bx_{1+1}, bx_2, \ldots, b_L)$.

Например, для строки $B=(1,2,3,4,5,6)$ при выборе $y_1=6$ и $y_2=2$ и соответственно $x_1=2$, $x_2=6$ результатом инверсии будет $B= (1,2,6,5,4,3)$.

Операции кроссинговера и мутации, используемые в простом ГА, изменяют структуру хромосом, в том числе разрушают удачные фрагменты найденных решений, что уменьшает вероятность нахождения глобального оптимума.

Для устранения этого недостатка в генетических алгоритмах используют схемы (схематы или шаблоны), представляющие собой фрагменты решений или хромосом, которые желательно сохранить в процессе эволюции. При использовании схем в генетическом алгоритме вводится новый алфавит (0,1,*), где * интерпретируется как «имеет значение 1 или 0».

Пример:

Схема(*0000) соответствует двум стрингам (10000 и 00000);

Схема (*111*) соответствует четырём строкам (01110, 11110, 01111, 11111);

Схема (0*1**) может соответствовать восьми пятизначным стрингам.

В общем случая хромосома длиной L максимально может иметь $3L$ (шаблонов), но только $2L$ различных альтернативных стрингов. Это следует из факта, что схеме (**) в общем случае могут соответствовать $3^2=9$ стрингов, а именно (**, *1, *0, 1*, 0*, 00, 01, 10, 11), и только $2^2=4$ альтернативные строки (00, 01, 10, 11), т.е. одной и той же строке может соответствовать несколько схем. Если в результате работы генетического алгоритма удалось найти схемы типа (11***) и (**111), то, применив к ним оператор кроссинговера, можно получить хромосому (11111), обладающую наилучшим значением целевой функции.

Схемы небольшой длины называются строительными блоками. Размер строительных блоков заметно влияет на качество и скорость нахождения результата. Вид строительного блока выбирается с учетом специфики решаемой задачи, а их разрыв в генетических алгоритмах допускается только в исключительных случаях, определяемых пользователем. Например, в схеме (****1) строительным блоком является элемент 1, а в схеме (10***) – составной элемент 10.

При использовании большого числа строительных блоков генетические алгоритмы, основанные на случайной генерации популяций и хромосом, переходят в разряд беспорядочных.

Стационарные генетические алгоритмы отличаются от поколенческих тем, что у первых размер популяции является заданным постоянным параметром,

который определяется пользователем, а у вторых размер популяции в последующих генерациях может увеличиваться или уменьшаться.

Процедура удаления лишних хромосом в стационарных и поколенческих генетических алгоритмах основана на эвристических правилах, примерами которых являются следующие:

• случайное равновероятное удаление хромосом;

• удаление хромосом, имеющих худшие значения целевой функции;

• удаление хромосом на основе обратного значения целевой функции;

• удаление хромосом на основе турнирной стратегии.

Следует иметь в виду, что использование в генетических алгоритмах тех или иных эвристик удаления хромосом может повлечь за собой негативные последствия. Например, удаление худших хромосом приводит к поврежденной утрате разнообразия и, как следствие, к попаданию целевой функции в локальный оптимум, а при наличии большого числа хромосом с плохими значениями целевой функции утрачивается направленность поиска и он превращается в «слепой» поиск.

Основные отличия ГА от других алгоритмов оптимизации:

• используются не параметры, а закодированные множества параметров;

• поиск осуществляется не из единственной точки, а из популяции точек;

• в процессе поиска используются значения целевой функции, а не ее приращения;

• применяются вероятные, а не детерминированные правила поиска и генерации решений;

• выполняется одновременный анализ различных областей пространства решений, в связи с чем возможно нахождение новых областей с лучшими значениями целевой функции за счёт объединения квазиоптимальных решений из разных популяций.

Достоинства генетических алгоритмов

1) ГА не имеет значительных математических требований к видам целевых функций и ограничений (отсутствует ограничение на дифференцируемость функций). Исследователь не должен упрощать модель объекта, теряя ее адекватность и искусственно добиваясь возможности применения доступных математических методов. При этом могут использоваться самые разнообразные целевые функции и виды ограничений (линейные и нелинейные), определенные на дискретных, непрерывных и смешанных универсальных множествах.

2) Гибкость. ГА хорошо работает при минимуме информации об окружающей среде (при высокой степени априорной неограниченности).

3) При использовании классических пошаговых методик глобальный оптимум может быть найден только в том случае, когда проблема обладает свойством выпуклости. В то же время эволюционные операции генетических алгоритмов позволяют эффективно отыскивать глобальный оптимум.

4) В ряде случаев ГА может находить только локальный минимум (максимум). Несмотря на это, дает быстрое нахождение приемлемого решения.

5) Комбинируется с другими методами искусственного интеллекта, и его эффективность может повышаться.

6) Применяется для решения поисковых задач, которые имеют большое пространство в поисках решения с целью уменьшения этого пространства поиска. Наиболее распространенное применение – решение задач оптимизации.

7) Минимизация ошибок. Высокая скорость поиска решений. Возможность распараллеливания вычислений.

Пример 1. Найти хромосому с максимальным количеством единиц.

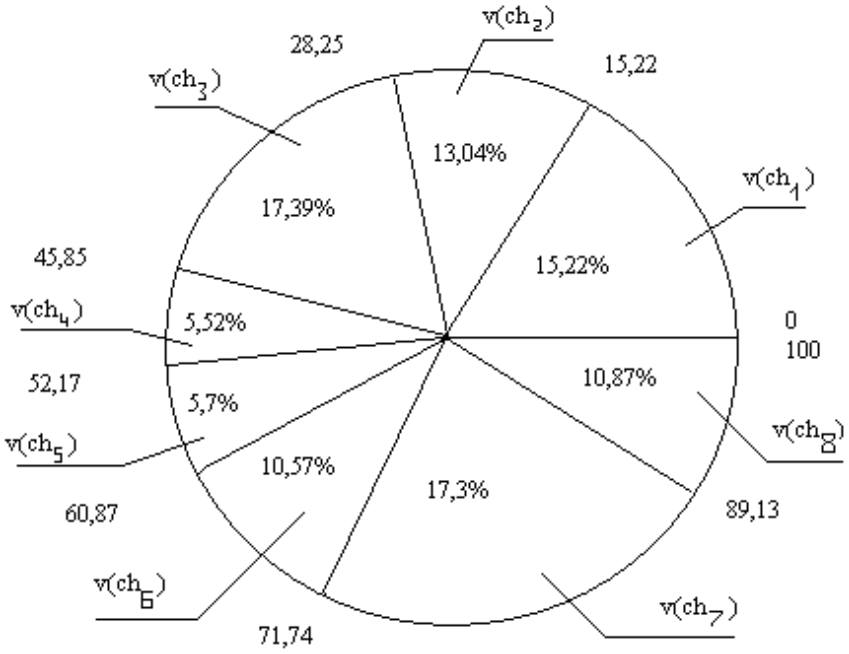
Допустим, что хромосомы состоят из 12 генов, а популяция насчитывает 8 хромосом. Понятно, что наилучшей будет хромосома, состоящая из 12 единиц.

Выбор исходной популяции хромосом. Необходимо случайным образом сгенерировать 8 двоичных последовательностей длиной 12 битов. Этого можно

достигнуть, например, подбрасыванием монеты (96 раз, при выпадании «орла» приписывается значение 1, а в случае «решки» – 0). Таким образом, можно сформировать исходную популяцию: *ch*1 = [111001100101] *ch*5 = [010001100100] *ch*2 = [001100111010] *ch*6 = [010011000101] *ch*3 = [011101110011] *ch*7 = [101011011011] *ch*4 = [001000101000] *ch*8 = [000010111100].

Оценка приспособленности хромосом к популяции. Функция приспособленности определяет количество единиц в хромосоме. Ее значения для каждой хромосомы из исходной популяции: *F*(*ch*1)=7, *F*(*ch*5)=4, *F*(*ch*2)=6, *F*(*ch*6)=5, *F*(*ch*3)=8, *F*(*ch*7)=8, *F*(*ch*4)=3, *F*(*ch*8)=5. Хромосомы *ch*3 и *ch*7 характеризуются наибольшими значениями функции принадлежности. Они считаются наилучшими кандидатами на решение задачи. Если условие остановки алгоритма не выполняется, то на следующем шаге производится селекция хромосом из текущей популяции.

Селекция хромосом. Селекция производится по методу рулетки. Для каждой из 8 хромосом текущей популяции получаем сектора рулетки, выраженные в процентах:



Принцип работы рулетки. $V(ch1)=15,22$; $V(ch5)=8,7$; $V(ch2)=13,04$; $V(ch6)=10,87$; $V(ch3)=17,39$; $V(ch7)=17,39$; $V(ch4)=6,52$; $V(ch5)=10,87$.

Розыгрыш с помощью колеса рулетки сводится к случайному выбору числа из интервала [0, 100], указывающего на соответствующий сектор на колесе, т.е. на конкретную хромосому. Допустим, что разыграны следующие 8 чисел: 70, 44, 9, 74, 44, 86, 48, 23. Это означает выбор хромосом *ch*7, *ch*3, *ch*1, *ch*7, *ch*3, *ch*7, *ch*4, *ch*2. Как видно, хромосома *ch*7 была выбрана трижды, а хромосома *ch*3 – дважды. Именно эти хромосомы имеют наибольшее значение функции приспособленности. Однако выбрана хромосома *ch*4 с наименьшим значением функции приспособленности. Все выбранные таким образом хромосомы включаются в так называемый родительский пул.

Применение генетических операторов. Допустим, что ни одна из отобранных в процессе селекции хромосом не подвергается мутации и все они составляют популяцию хромосом, предназначенных для скрещивания. Это означает, что вероятность скрещивания p_c =1, а вероятность мутации $p_м$ = 0. Допустим, что из этих хромосом случайным образом сформированы пары родителей: *ch*2 и *ch*7, *ch*1 и *ch*7, *ch*3 и *ch*4, *ch*3 и *ch*7. Для первой пары случайным образом выбрана точка скрещивания l_k = 4, для второй l_k = 3, для третьей l_k = 11, для четвертой l_k = 5. При этом процесс скрещивания протекает так, как показано на рис.81.

ПЕРВАЯ ПАРА РОДИТЕЛЕЙ		ПЕРВАЯ ПАРА ПОТОМКОВ
[001100111010]		[001111011011]
	Скрещивание	
[101011011011]	⟶	[101000111010]
ВТОРАЯ ПАРА РОДИТЕЛЕЙ		ВТОРАЯ ПАРА ПОТОМКОВ
[111001100101]		[111011011011]
	Скрещивание	
[101011011011]	⟶	[101001100101]
ТРЕТЬЯ ПАРА РОДИТЕЛЕЙ		ТРЕТЬЯ ПАРА ПОТОМКОВ
[011101110011]		[011101110010]
	Скрещивание	
[001000101000]	⟶	[001000101001]
ЧЕТВЕРТАЯ ПАРА РОДИТЕЛЕЙ		ЧЕТВЕРТАЯ ПАРА ПОТОМКОВ
[011101110011]		[011101011011]
	Скрещивание	
[101011011011]	⟶	[101011110011]

Рис.81. Работа генетического алгоритма

В результате выполнения оператора скрещивания получается 4 пары потомков. Если бы при случайном подборе хромосом для скрещивания были объединены, например, *ch*3 с *ch*3 и *ch*4 с *ch*7 вместо *ch*3 с *ch*4 и *ch*3 с *ch*7, а другие пары остались без изменения, то скрещивание *ch*3 с *ch*3 дало бы две такие же хромосомы независимо от разыгранной точки скрещивания. Это означало бы получение двух потомков, идентичных своим родителям. Заметим, что такая ситуация вероятна для хромосом с наибольшим значением функции приспособленности, т.е. именно такие хромосомы получают наибольшие шансы на переход в новую популяцию.

Работа ГА. Формирование новой популяции. После выполнения операции скрещивания мы получаем следующую популяцию потомков: *Ch*1 = [001111011011] *Ch*5 = [011101110010] *Ch*2 = [101000111010] *Ch*6 = [001000101001] *Ch*3 = [111011011011] *Ch*7 = [011101011011] *Ch*4 = [101001101101] *Ch*8 = [101011110011]. Для отличия от хромосом предыдущей популяции обозначения вновь сформированных хромосом начинаются с большой буквы *C*.

Согласно блок-схеме генетического алгоритма производится возврат ко второму этапу, т.е. к оценке приспособленности хромосом из вновь сформированной популяции, которая становится текущей. Значения функций приспособленности этой популяции составляют *F*(*Ch*1)=8, *F*(*Ch*5)=7, *F*(*Ch*2)=6, *F*(*Ch*6)=4, *F*(*Ch*3)=9, *F*(*Ch*7)=8, *F*(*Ch*4)=6, *F*(*Ch*8)=8. Заметно, что популяция характеризуется более высоким средним значением функции приспособленности, чем популяция родителей. В результате скрещивания получена хромосома *Ch*3 с наибольшим значением функции приспособленности, которым не обладала ни одна хромосома из родительской популяции. Однако могло произойти и обратное, поскольку после скрещивания на первой итерации хромосома, которая в родительской популяции характеризовалась наибольшим значением функции приспособленности, могла просто «потеряться». Помимо этого, «средняя» приспособленность новой популяции все равно оказалась бы выше предыдущей, а

хромосомы с большими значениями функции приспособленности имели бы шансы появиться в следующих поколениях.

Пример 2. Найти максимум функции $f(x) = 2x^2+1$ для $x = 0,\ldots,31$.

Для применения ГА необходимо прежде всего закодировать в виде двоичных последовательностей значения x. Очевидно, что числа 0,…,31 можно представить в двоичной системе счисления. При этом 0 кодируется как 00000, а 31 – 11111, т.е. это цепочка из 5 битов.

Выберем случайным образом популяцию, состоящую из 6 кодовых последовательностей N=6 (30 раз подбрасываем монету). Допустим, что выбраны хромосомы: *ch*1 = 10011, *ch*4= 10101, *ch*2= 00011, *ch*5= 01000, *ch*3= 00111, *ch*6 = 1110. Соответствующие им фенотипы – это числа из интервала (0,…,31). *ch*1 = 19, *ch*4= 21, *ch2*= 3, *ch*5= 8, *ch*3= 7, *ch*6 = 29. По вышеприведенной формуле рассчитываем значения функции приспособленности *Fch*1 = 723, *Fch*4= 883, *Fch*2= 19, *Fch*5= 129, *Fch*3= 99, *Fch*6 = 1683. Среднее значение *F* =589, сумма – 3536.

Рассчитываем *Pch*1 = 20,45; *Pch*4= 24,97; *Pch*2= 0,54; *Pch*5= 3,65; *Pch3*=2,8; *Pch*6 = 47,6.

Методом рулетки выбираем 6 хромосом для репродукции (вращаем рулетку 6 раз). Допустим, что выбраны числа 97, 26, 54, 13, 31, 88. Это означает выбор хромосом *ch*6, *ch*4, ch6, *ch*1, *ch*4, *ch*6. Пусть скрещивание выполняется с вероятностью 1. Для скрещивания случайным образом формируем пары *ch*1 и *ch*4, *ch*4 и *ch*6, *ch*6 и *ch*6. Допустим, что случайным образом выбрана точка скрещивания, равная 3, для первой пары, а также точка скрещивания, равная 2, для второй пары. Операция скрещивания для третьей пары бесполезна, т.к. дает тот же результат. При условии, что вероятность мутации равна 0, в новую популяцию включаются хромосомы: *Ch*1 = 10001, *Ch*4= 11101, *Ch*2= 10111, *Ch*5= 11101, *Ch*3= 10101, *Ch*6 = 11101.

В результате декодирования получаем числа *Ch*1 = 17, *Ch*4= 29, *Ch*2= 23, *Ch*5= 29, *Ch*3= 21, *Ch*6 = 29. Функции приспособленности *Fch*1 = 579,

*Fch*4= 1683, *Fch*2= 1059, *Fch*5= 1683, *Fch*3= 883, *Fch*6 = 1683. Среднее значение целевой функции *F* =1262, что значительно больше предыдущего значения.

Вопросы для самопроверки

1. Машинная эволюция. Генетические алгоритмы. В чем состоит идея генетических алгоритмов?
2. Опишите в общих чертах работу генетического алгоритма. Приведите блок-схему генетического алгоритма.
3. Что такое функция приспособленности, нормализованная и средняя приспособленность?
4. Опишите процесс селекции. Метод рулетки.
5. Перечислите операторы репродукции и приведете примеры их применения.
6. Проиллюстрируйте работу генетического алгоритма на примере поиска максимума квадратичной функции.
7. Проиллюстрируйте работу генетического алгоритма на примере обучения нейронной сети.

Задания для самостоятельной работы

Задание 1. Нужно найти максимум функции $f(x)= -2x^2+15x+50$ на отрезке [0;7] с точностью ε=0,5 с помощью генетического алгоритма.

1. Найти максимум аналитически.
2. С шагом 0,5 определить фенотипы (x_o=0).
3. Для каждого фенотипа найти генотип.
4. Размер популяции *N*=6.
5. Случайный выбор аллелей для особей первоначальной популяции (всего 4x6 случайных чисел).
6. Найти функцию приспособленности для первоначальной популяции.

7. Составить родительский пул (мощности 6). (Выбор более приспособленных хромосом методом рулетки).
8. Сформировать родительские пары 6x6=36.
9. Случайным образом выбрать 6 пар.
10. Применить к ним кроссовер (локус= 2).
11. Для новой популяции посчитать значение функции приспособленности.
12. Проверить критерий останова. Если точность не достигнута, то переход на шаг 5.

Указание: Если ошибка перестала уменьшаться на некотором шаге, то нужно применить ***элитизм*** (заменить худшую особь на лучшую из первоначальной популяции).

Задание 2. Определение оптимального набора весов модели ИНС с помощью генетического алгоритма.

Дана искусственная нейронная сеть, которая имеет архитектуру 2 – 3 – 1 с весовыми коэффициентами:

для первого слоя: для второго слоя:

$$\begin{pmatrix} 0,1 & 0,1 \\ -0,2 & -0,1 \\ 0,1 & 0,3 \end{pmatrix} \qquad \begin{bmatrix} 0,2 \\ 0,2 \\ 0,3 \end{bmatrix}$$

Параметры обучения: $\alpha = 1$ и $\eta = 0,1$; входной образец (0,1; 0,9) и целевое выходное значение 0,9.

1. Сформировать случайным образом начальную популяцию весов из векторов $\overline{\omega_i} = (\omega_{13}, \omega_{14}, \omega_{15}, \omega_{23}, \omega_{24}, \omega_{25}, \omega_{36}, \omega_{46}, \omega_{56})$.

   Пусть начальная популяция состоит из 10 хромосом, где каждый ген – это случ. число из $[-0,2; 0,3]$.
2. Рассчитать значение функции приспособленности для каждой хромосомы $F(\overline{\omega_i}) = -(d - y_i)^2$, где y_i – это выходное значение нейронной сети.
3. Рассчитать среднее значение функции приспособленности.

4. Провести операцию селекции методом рулетки (для этого определить вероятность для каждой хромосомы):

$p_i = \frac{F_i^{-1}}{S}$, где $F_i^{-1} = \frac{1}{|F(\overline{\omega}_i)|}$, а $S = \sum_{i=1}^{10} F_i^{-1}$ и определить пару родительских хромосом.

5. Выполнить шаг скрещивания, используя одноточечный кроссовер. В результате получить два потомка.
6. Заменить худшую хромосому начальной популяции (у нее наименьшая вероятность) на лучшую из дочерних хромосом.
7. Если не выполнено условие останова (ошибка ИНС существенна или не исчерпан лимит итераций работы генетического алгоритма), то перейти на шаг 2.

Задание 3. Оптимизация многоэкстремальных функций с помощью генетических алгоритмов.

1. Параметры программы, реализующей генетический алгоритм:

DIM – размерность задачи

POPSIZE – размер популяции

NUMGEN – число поколений (время эволюции)

PROB_MUTATION – вероятность мутаций (слишком большая портит решение, слишком малая не может найти дополнительные варианты для поиска)

PROB_CROSS – вероятность скрещивания

MN – массив ограничений «снизу»

MX – массив ограничений «сверху»

CROSS_METHOD – метод скрещивания

ONE_POINT – одноточечный «кроссовер»

TWO_POINT – двухточечный «кроссовер»

UNIFORM – равномерный «кроссовер»

SELECT_METHOD – метод отбора

ROULETTE – по правилу рулетки

2. Осуществить вывод данных программы ГА в файлы (тип *txt* или *Excel*). Из файла.*txt* импортировать данные в *Excel*. Изменить тип данных – перевести их в числовой формат (заменой · на ,).
3. Используя исходные параметры программы «по умолчанию», построить график зависимости среднего значения функции приспосабливаемости и максимального значения функции приспосабливаемости.
4. Выясните, на какой итерации достигается максимум. Автоматизируйте поиск номера итерации с помощью функций *Excel*.
5. Написать программу для функции своего варианта.
6. Выполнить шаги 3 и 4 для своей функции.
7. Для своей функции, отменив элитизм, построить график зависимости среднего значения функции приспосабливаемости и максимального значения функции приспосабливаемости.
8. Для одноточечного кроссовера построить графики и проанализировать, как они будут меняться при увеличении/уменьшении вероятности кроссовера.
9. Для двухточечного кроссовера построить графики и проанализировать, как они будут меняться при увеличении/уменьшении вероятности кроссовера.
10. Для своей функции, изменив вероятность мутации, построить график зависимости среднего значения функции приспосабливаемости и максимального значения функции приспосабливаемости. Проанализировать, как они будут меняться при увеличении/уменьшении вероятности мутации.
11. Оформить отчет: «**Влияние параметров генетического алгоритма на эффективность поиска экстремума многоэкстремальных функций».**

Параметр и его значение	Экстремум функции X^*, $F(X^*)$	На какой итерации найден экстремум

12.Сделать вывод: при каких параметрах эффективнее работает алгоритм.

Тестовая функция №1

$$F(x_1, x_2) = \frac{100}{100(x_1^2 - x_2) + (1 - x_1)^2 + 1}, \qquad -1{,}28 \le x_{1,2} \le 1{,}28,$$

$$F^* = F(1{,}00,\ 1{,}00) = 100{,}00.$$

Тестовая функция №2

$$F(x_1, x_2, \ldots, x_5) = \sum_{i=1}^{5} \lceil x_i \rceil, \quad 5{,}12 \le x_{1,2,3,4,5} \le 5{,}12,$$

$$F^* = 30_{\text{на гиперплоскости}}\ 5{,}00 \le x_{1,2,3,4,5} \le 5{,}12.$$

Тестовая функция №3

$$F(x_1, x_2) = 0{,}002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}, \quad \text{где } a_{1j} = 16[(j \bmod 5) - 2],$$

$$a_{2j} = 16[(j\,\%5) - 2], \quad -65{,}536 \le x_{1,2} \le 65{,}536\ .$$

$$F^* = F(-16, -32) = 1{,}002.$$

Тестовая функция №4

$$F(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10\cos(2\pi x_1) - 10\cos(2\pi x_2), \qquad -5{,}12 \le x_{1,2} \le 5{,}12$$

$$F^* = F(4{,}52299,\ 4{,}52299) = F(-4{,}52299,\ 4{,}52299) = F(-4{,}52299,\ -4{,}52299) =$$

$$= F(4{,}52299,\ -4{,}52299) = 80{,}7065.$$

Тестовая функция №5

$$F(x_1, x_2) = \frac{1}{\frac{x_1^2 + x_2^2}{200} - \cos(x_1)\cos\left(\frac{x_2}{\sqrt{2}}\right) + 2} \qquad -20 \le x_{1,2} \le 20$$

$$F^* = F(0{,}00,\ 0{,}00) = 1{,}0.$$

Тестовая функция №6

$$F(x_1, x_2, \ldots, x_{10}) = \sum_{i=1}^{10}\left(10\cos(2\pi x_i) - x_i^2\right) - 100, \quad -5{,}12 \le x_{1,2,\ldots,10} \le 5{,}12,$$

$$F^* = F(0.00, 0.00, \ldots, 0.00) = 0.00.$$

Тестовая функция №7

$$F(x_1, x_2, \ldots, x_{10}) = 10 - \left(\sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right), \quad -512{,}0 \le x_{1,2,\ldots,10} \le 512{,}0,$$

$$F^* = F(0.00, 0.00, \ldots, 0.00) = 10.00.$$

5. Экспертные системы. Модели знаний

Экспертная система – это информационная система, в которую включены знания специалистов о некоторой проблемной области и которая в пределах этой области способна принимать экспертные решения.

К разработке экспертной системы необходимо привлекать инженеров по знаниям, экспертов в данной предметной области и конечных пользователей.

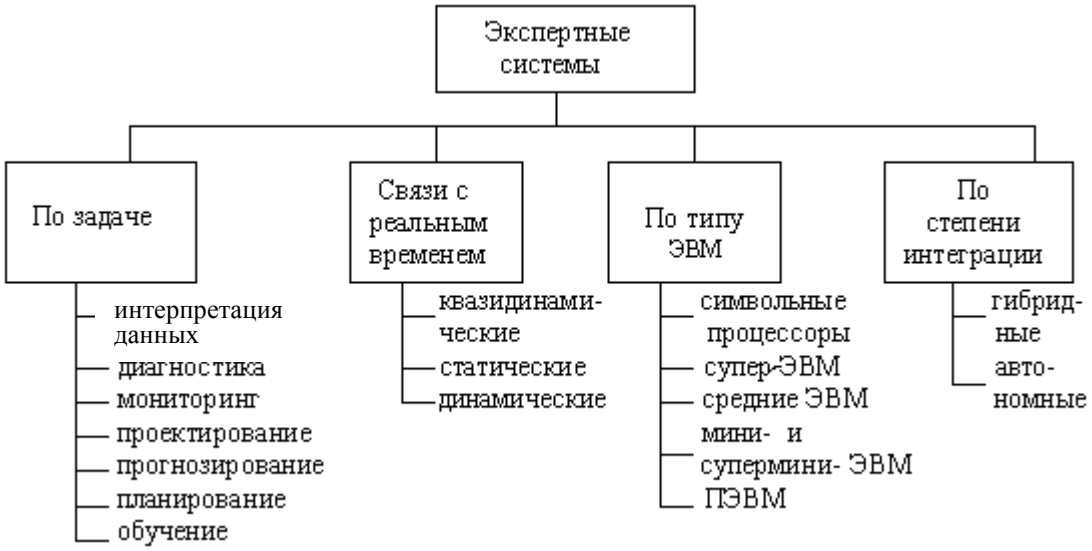
Инженер по знаниям (когнитолог) является экспертом по языку искусственного интеллекта и представлению знаний. Его главная задача – выбрать программный и аппаратный инструментарий для проекта, помочь эксперту в данной области членораздельно сформулировать необходимую информацию, а также реализовать ее в корректной и эффективной базе знаний. Часто инженер по знаниям изначально не знаком с предметной областью.

Знания о предметной области обеспечивает эксперт. Экспертом предметной области обычно является тот человек, который работал в этой области и понимает принципы решения ее задач, знает приемы решения, может обеспечить управление неточными данными, оценку частичных решений. Эксперт в предметной области отвечает за передачу этих навыков инженеру по знаниям.

В большинстве приложений основные проектные ограничения определяет конечный пользователь. Обычно разработка продолжается до тех пор, пока

пользователь не будет удовлетворен. Навыки и потребности пользователя учитываются в течение всего цикла разработки.

Цель исследований в области экспертных систем состоит в разработке программ (устройств), которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом. В большинстве случаев экспертные системы решают трудноформализуемые задачи или задачи, не имеющие алгоритмического решения.



Интерпретация – анализ наблюдаемых данных и ситуаций с целью определения их смысла и описания.

Пример: обнаружение и идентификация различных типов океанских судов – *SIAP*.

Диагностика – классификация и поиск неисправностей в живых и неживых системах, основанные на результатах интерпретации.

Пример: диагностика и терапия сужения коронарных сосудов – *ANGY*; диагностика ошибок в аппаратуре и математическом обеспечении ЭВМ – система *GRIP* и др.

Мониторинг – сравнение наблюдений с критическими точками плана и выдача сообщений при отклонении от плана, либо – непрерывный процесс интерпретации сигналов и выдача сообщений в ситуациях, требующих использо-

вание экспертной системы более высокого уровня или вмешательства человека.

Пример: помощь диспетчерам атомного реактора *REACTOR*; контроль аварийных датчиков на химическом заводе *FALCON* и др.

Проектирование – нахождение конфигурации компонентов системы, которая удовлетворяет целевым условиям и множеству проектных ограничений.

Пример: проектирование БИС – *CADHELP*; синтез электрических цепей *SYN* и др.

Прогнозирование – проектирование возможных последствий данной ситуации.

Пример: предсказание погоды – система *WILLARD*; прогнозы в экономике – *ECON* и др.

Планирование – разработка последовательности действий для достижения множества целей при данных начальных условиях и временных ограничениях.

Пример: планирование промышленных заказов – *ISIS* планирование эксперимента – *MOLGEN* и др.

Инструктирование (обучение) – помощь в образовательном процессе по изучению технической области. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Кроме того, они планируют акт общения с учеником в зависимости от успехов ученика с целью передачи знаний.

Пример: система *PROUST* – обучение языку Паскаль и др.

Управление – управление поведением сложной среды или системы.

Тестирование – проверка качества работы с помощью специальных тестов.

Ремонт – выполнение плана организации исправления некоторого обнаруженного дефекта.

Классификация по связи с реальным временем:

Квазидинамические ЭС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени.

Пример: микробиологические ЭС, в которых снимаются лабораторные измерения с технологического процесса один в 4 – 5 ч (например, производство лизина) и анализируется динамика полученных показателей по отношению к предыдущему измерению.

Статические ЭС разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени. Они полностью стабильны.

Пример: диагностика неисправностей в автомобилях.

Динамические ЭС работают в сопряжении с датчиками объектов в режиме времени с непрерывной интерпретацией поступаемых данных.

Пример: управление гибкими производственными комплексами, мониторинг в реанимационных палатах и т.д.

Классификация по степени интеграции с другими программами:

Автономные ЭС работают непосредственно в режиме консультаций с пользователем для специфически «экспертных» задач, для решения которых не требуется привлекать традиционные методы обработки данных (расчеты, моделирование и т.д.)

Гибридные ЭС представляют программный комплекс, агрегирующий стандартные пакеты прикладных программ (например, математическую статистику, линейное программирование или системы управления базами данных) и средства манипулирования знаниями.

Несмотря на большую выгодность гибридных ЭС, их создание и разработка являются более сложной задачей, нежели автономных ЭС. Некоторые инструментальные средства построения экспертных систем.

Классификация по типу ЭВМ:

На сегодняшний день существуют:

ЭС для уникальных стратегически важных задач на супер ЭВМ (Эльбрус, *CRAY*, *CONVEX* и др.);

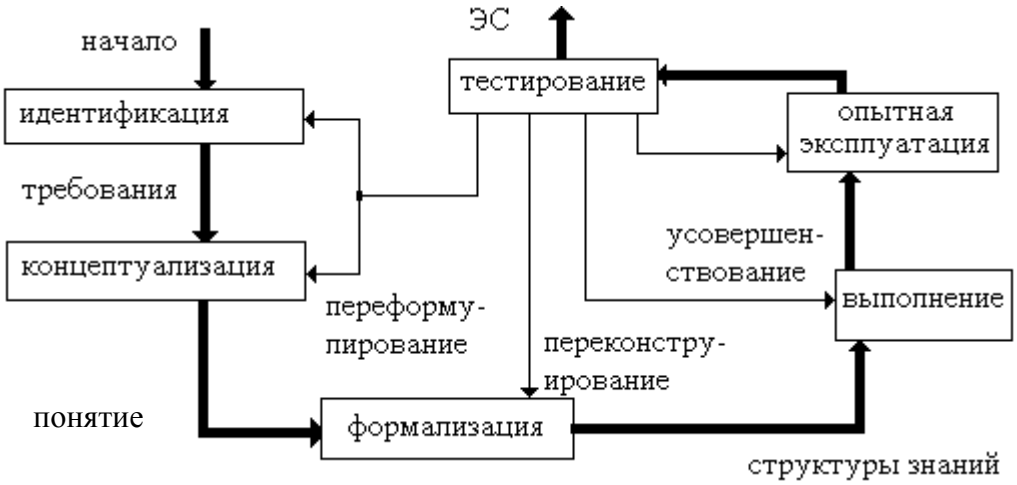
ЭС на ЭВМ средней производительности (типа ЕС ЭВМ, *mainframe*);

ЭС на символьных процессорах и рабочих станциях (*SUN*, *APOLLO*);

ЭС на мини- и супермини- ЭВМ (VAX, *micro-VAX* и др.);

ЭС на персональных компьютерах (*IBM PC*, *MAC* II и подобные).

Технология разработки ЭС включает в себя шесть этапов



1) На этапе ***идентификации*** необходимо выполнить следующие действия:

- определить задачи, подлежащие решению и цели разработки;
- определить экспертов и тип пользователей.

2) На этапе ***концептуализации*:**

- проводится содержательный анализ предметной области;
- выделяются основные понятия и их взаимосвязи;
- определяются методы решения задач.

3) На этапе ***формализации***:

- выбираются программные средства разработки ЭС;
- определяются способы представления всех видов знаний;
- формализуются основные понятия.

4) На этапе ***выполнения*** (наиболее важном и трудоемком) осуществляется наполнение экспертом БЗ, при котором процесс приобретения знаний разделяют:

- на «извлечение» знаний из эксперта;
- на организацию знаний, обеспечивающую эффективную работу ЭС;

- на представление знаний в виде, понятном для ЭС.

Процесс приобретения знаний осуществляется инженером по знаниям на основе деятельности эксперта.

5) На этапе ***тестирования*** эксперт и инженер по знаниям с использованием диалоговых и объяснительных средств проверяют компетентность ЭС. Процесс тестирования продолжается до тех пор, пока эксперт не решит, что система достигла требуемого уровня компетентности.

6) На этапе ***опытной эксплуатации*** проверяется пригодность ЭС для конечных пользователей. По результатам этого этапа возможна существенная модернизация ЭС.

Процесс создания ЭС не сводится к строгой последовательности этих этапов, так как в ходе разработки приходится неоднократно возвращаться на более ранние этапы и пересматривать принятые там решения.

Выделяют три **стратегии получения знаний** при разработке систем, основанных на знаниях:

1. Приобретение знаний. Это означает получение знаний с использованием компьютера при наличии подходящего программного инструментария.

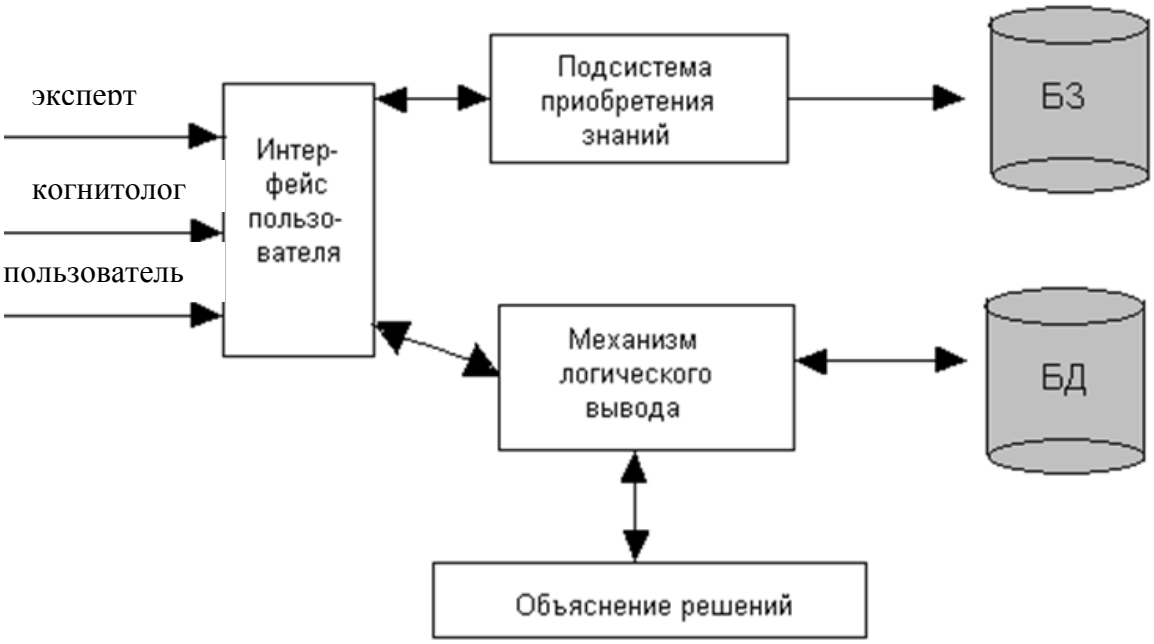
2. Формирование знаний. Под этим понимают получение знаний с использованием программ обучения при наличии репрезентативной (т.е. достаточно представительной) выборки примеров принятия решений в предметной области и соответствующих пакетов прикладных программ.

3. Извлечение (выявление) знаний. Сюда относят получение знаний без использования вычислительной техники путем непосредственного контакта инженера по знаниям с источником знаний, в результате которого становятся явными структура его представлений о предметной области, а также процесс рассуждений специалистов при принятии решения.

Потенциальные источники знаний включают в себя экспертов, специальную литературу (учебники, справочники, отчеты, истории болезни и т.п.), справочно-нормативные сведения, набор данных (базы данных), личный опыт и др.

АРХИТЕКТУРА ЭКСПЕРТНОЙ СИСТЕМЫ

Схема обобщенной экспертной системы имеет вид:



1. Лингвистический процессор (интерфейс с пользователем) осуществляет диалоговое взаимодействие с пользователем (экспертом) на естественном для него языке (естественный язык, профессиональный язык, язык графики и т. д.). В экспертных системах применяются различные варианты реализации интерфейса: меню-ориентированный (графический, командный, речевой).

2. База знаний обеспечивает хранение знаний, представленных с помощью одной из моделей: логической, продукционной, фреймовой, сетевой.

3. Рабочая память (база данных) хранит данные, имеющие отношения к анализируемой системой ситуации.

4. Интерпретатор (машина вывода) на основе входных данных, продукционных правил и общих фактов о проблемной области формирует решение задачи.

5. Компонента приобретения знаний используется как с целью автоматизации процесса наполнения ЭС знаниями, так и при корректировке базы знаний, при ее обновлении, пополнении или исключении элементов знаний.

6. Объяснительная компонента, дающая объяснение действий системы и отвечающая на вопросы о том, почему некоторые заключения были сделаны или отвергнуты.

Экспертная система работает в двух режимах: в режиме приобретения знаний и в режиме решения задач.

В режиме ***приобретения знаний*** общение с ЭС осуществляет эксперт через посредничество инженера по знаниям. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования данными, характерные для рассматриваемой проблемной области. Эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области.

Режиму приобретения знаний при традиционном подходе к разработке программ соответствуют этапы алгоритмизации, программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода разработку программ осуществляет эксперт (с помощью ЭС), не владеющий программированием.

В ***режиме консультации*** общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ получения решения. Пользователь в зависимости от назначения ЭС может не быть специалистом в данной проблемной области. Термин «пользователь» является многозначным, так как, кроме конечного пользователя, применять ЭС может и эксперт, и инженер по знаниям, и программист. Поэтому, когда хотят подчеркнуть, что речь идет о том, для кого делалась ЭС, используют термин «конечный пользователь».

В режиме консультации данные о задаче пользователя обрабатываются диалоговой компонентой, которая выполняет следующие действия:

распределяет роли участников (пользователя и ЭС) и организует их взаимодействие в процессе кооперативного решения задачи; преобразует данные пользователя о задаче, представленные на первичном для пользователя языке, во внутренний язык системы; преобразует сообщения системы, представленные на внутреннем языке, в сообщения на языке, привычном для пользователя (обычно это ограниченный естественный язык или язык графики).

После обработки данные поступают в рабочую память. На основе входных данных из рабочей памяти, общих данных о проблемной области и правил из базы знаний решатель (интерпретатор) формирует решение задачи.

В отличие от традиционных программ ЭС в режиме задачи не только исполняет предписанную последовательность операций, но и предварительно формирует ее. Если ответ ЭС не понятен пользователю, он может потребовать объяснения, как ответ получен.

В состав экспертной системы входят знания специалистов о некоторой проблемной области.

Знания – это воспринятая живым существом (субъектом) информация из внешнего мира и в отличие от «информации» «знание» субъективно. Оно зависит от особенностей жизненного опыта субъекта, его истории взаимоотношения с внешней средой, т.е. от особенностей процесса его обучения или самообучения. На этом уровне абстракции знание уникально и обмен знанием между индивидуумами не может происходить без потерь в отличие от данных, в которых закодирована информация (неоднородность) и которые могут передаваться от передатчика к приемнику без потерь (не учитывая возможность искажения вследствие помех). Знание передается между субъектами посредством какого-либо языка представления знаний, наиболее типичным представителем которого является естественный язык.

Данные – это совокупность сведений, зафиксированных на определенном носителе в форме, пригодной для постоянного хранения, передачи и обработки. Преобразование и обработка данных позволяют получить информацию.

Информация – это результат преобразования и анализа данных. Например, в базах данных хранятся различные данные, а по определенному запросу система управления базой данных выдает требуемую информацию.

Знания – это зафиксированная и проверенная практикой обработанная информация, которая использовалась и может многократно использоваться для принятия решений. Знания – это вид информации, которая хранится в базе знаний и отображает знания специалиста в конкретной предметной области.

Основные характеристики знаний:

– *Внутренняя интерпретируемость*. Данные, хранимые в памяти ЭВМ, могут интерпретироваться только соответствующей программой. Данные без программы не несут никакой информации, в то время как знания имеют интерпретацию, поскольку они содержат одновременно и данные, и соответствующие им имена, описания.

– *Структурированность*. Рекурсивная вложимость одних информационных единиц в другие – возможность произвольного установления между отдельными информационными единицами отношений типа «часть – целое», «род – вид» или «элемент – класс».

– *Связность*. Между информационными единицами должна быть предусмотрена возможность установления связей различного типа, характеризующих отношения между информационными единицами (например, «одновременно», «причина – следствие», «аргумент – функция»). Все отношения могут быть разделены на 4 категории: отношения структуризации (задают иерархию информационных единиц), функциональные отношения (несут процедурную информацию, позволяющую вычислять одни информационные единицы через другие), каузальные отношения (задают причинно-следственные связи) и семантические отношения (все остальные отношения).

– *Семантическая метрика*. Между информационными единицами задают отношения релевантности, которые характеризуют ситуационную близость

информационных единиц. Отношение релевантности позволяет находить знания, близкие к уже найденным.

– *Активность*. Изменение состояния информационной базы приводит к выполнению некоторых действий. Например, добавление в базу знаний новых фактов или описаний событий приводит к запуску программы, проверяющей непротиворечивость новых и старых знаний.

Перечисленные характеристики определяют разницу между данными и знаниями, при этом базы данных перерастают в базы знаний.

Центральным вопросом построения систем, основанных на знаниях, является выбор формы представления знаний. Представление знаний – это способ формального выражения знаний о предметной области в компьютерно-интерпретируемой форме. Можно предложить следующий перечень критериев оценки моделей представления знаний:

– уровень сложности (абстрактности) элемента знаний, с которыми работает модель;

– универсальность представления знаний – возможность описания знаний из различных предметных областей;

– естественность и наглядность представления знаний при использовании;

– способность модели к обучению и формированию новых, непротиворечивых знаний;

– размерность модели по объему памяти, необходимому для хранения элемента модели;

– удобство разработки системы на основе модели.

Проведем анализ современных моделей представления знаний по указанным критериям.

Модель представления знаний с помощью фактов и правил (продукционная модель) построена на использовании выражений вида: ЕСЛИ (условие) – ТО (действие). Если текущая ситуация (факты) в задаче удовлетворяет или согласуется с частью правила ЕСЛИ, то выполняется действие, определяемое ча-

стью ТО. Это действие может оказаться воздействием на окружающий мир или же повлиять на управление программой (например, вызвать проверку и запуск некоторого набора других правил), или может сводиться к указанию системе добавить новый факт или гипотезу в базу данных.

Сопоставление частей ЕСЛИ правил с фактами может породить так называемую цепочку выводов – дерево решений. Один из главных недостатков метода представления знаний с помощью правил – значительные затраты времени на построение цепочки вывода. При частом использовании какого-либо дерева решений система редуцирует («сжимает») дерево решений до нового правила и вводит его в базу знаний. Это действие называют продукцией правил. Правило такого рода имеет значительно большую размерность, чем исходные правила. Системы, построенные на основе продукционных моделей, более эффективны по затратам памяти и по быстродействию, чем системы, основанные просто на правилах.

По уровню абстрактности элемента знаний модель работает с простейшими составляющими знания – фактами и правилами. Модель направлена на решение простых, однородных задач и приводит к резкому падению эффективности решения таких проблем, которые состоят из нескольких разнородных задач. Серьезнейшим недостатком является невозможность эффективно описать правила с исключениями. Объем памяти, необходимый для хранения элемента знаний модели – конструкции ЕСЛИ – ТО, мал в силу его простоты. Однако база знаний, описывающая реальную, даже не очень сложную задачу, должна содержать сотни и тысячи правил.

Одной из первых была система *DENDRAL*, созданная для формирования заключения о структурах химических соединений на основании масс-спектрометрии. Не менее известными экспертными системами, основанными на описанной модели, являются: *MYCIN* – система для диагностирования бактериальных инфекций; *INTERNIST* (позже – *CADUCEUS*) – система-консультант в области общей терапии.

Эффективность этих ЭС объясняется довольно просто. ***Продукционная модель*** накладывает ряд ограничений на решаемую задачу, объем знаний и некоторые другие параметры ИС. Создатели первых подобных систем строили их в рамках этих ограничений и получили эффективные и удобные решения. Каждая из перечисленных ЭС охватывает узкую и сравнительно хорошо определенную предметную область. Разработанные системы настолько понравились пользователям, что были предприняты попытки применения данного подхода к другим областям знаний с аналогичными свойствами. Из системы *MYCIN* извлечена проблемно-независимая часть в форме «пустой» системы *MYCIN*, названной *EMYCIN*. Последняя использована, например, при создании систем *PUFF* (обеспечение активной медицинской помощи при респираторных заболеваниях), *SACOM* (расчеты механических структур), *CLOT* (измерение состава крови) и т.д.

Модель представления знаний с помощью логики предикатов использует в своей основе математический аппарат символьной логики. Основными формализмами представления предикатов являются «терм», устанавливающий соответствие знаковых символов описываемому объекту, и предикат для описания отношения сущностей в виде реляционной формулы, содержащей в себе термы. Когда говорится «предикат», то обычно имеется в виду, что в него входит терм-переменная. Например, таким предикатом является Начальник (*X*,*Y*). Пусть «Петров», «Иванов» – это термы. Если между ними имеется отношение *подчинения*, то это отношение описывается как Начальник (Петров, Иванов).

Предикат, все термы которого являются термами-константами, называется высказыванием. По уровню абстрактности элемента знаний эта модель, как и предыдущая, работает с простейшими составляющими знания – фактами и правилами. Модель универсальна, однако, так же, как и модель представления знаний с помощью фактов и правил, не может быть использована для создания ИС, которые должны одновременно манипулировать специальными знаниями из разных предметных областей.

С помощью логики предикатов можно, определяя произвольным образом знания, выяснить, имеются или отсутствуют противоречия между новыми и уже существующими знаниями. Объем памяти, необходимый для хранения элемента знаний – предиката или предикатной формулы, мал в силу его простоты. Однако база знаний, описывающая реальную, даже не очень сложную предметную область, должна содержать значительное количество указанных элементов.

Модель привлекает разработчиков высокой модульностью, легкостью внесения в систему дополнений и изменений, простотой механизма логического вывода и часто применяется в промышленных ИС.

Семантические сети

Модель представления знаний ***с помощью семантических сетей*** состоит из вершин, называемых узлами, соответствующих объектам, концепциям или событиям, и связывающих их дуг, описывающих отношения между рассматриваемыми объектами. Дуги могут быть определены разными методами. Обычно для представления иерархии используются дуги типа *IS-A* (отношение «является») и *HAS-PART* (отношение «имеет часть»). Они также устанавливают иерархию наследования в сети, т.е. элементы более низкого уровня в сети могут наследовать свойства элементов более высокого уровня, что экономит память, поскольку информацию о наследуемых свойствах не нужно повторять в каждом узле сети.

Выводы на семантических сетях реализуются через отношения между элементами, однако они таят в себе угрозу возникновения противоречий.

Модель универсальна и легко настраивается. Характерная особенность семантической сети – наглядность знаний как системы.

Семантические сети применены в системе *CASNET* (*Caysal Associational NETwork*). Целью разработки были исследования стратегий медицинской диагностики, в основу которых положены психологические и функциональные модели болезней. На основе семантических сетей также разработана известная

система *PROSPECTOR*, предназначенная для оказания помощи геологам-изыскателям и способная давать три типа «советов»: оценку местности на предмет существования определенных залежей, оценку геологических ресурсов региона и выбор участков местности, наиболее благоприятных для бурения. Программа создана компанией *SRI International* (совместно с консультантами по геологии) и организацией *U.S. Geological Survey*. Серьезным недостатком систем *CASNET* и *PROSPECTOR* является их неудовлетворительная способность объяснить свои решения.

Фреймы

Модель представления знаний с помощью фреймов предложена Марвином Минским (Минский Марвин Ли (родился в 1927 г., Нью-Йорк). Американский математик, кибернетик, специалист по компьютерным наукам в 1951 г. создал первую случайносвязанную нейросетевую обучаемую машину). Он описывает фреймы следующим образом: «Фрейм – это структура данных, представляющая стереотипную ситуацию вроде нахождения внутри некоторого рода жилой комнаты или сбора на вечеринку по поводу рождения ребенка. К каждому фрейму присоединяются несколько видов информации. Часть этой информации – о том, как использовать фрейм. Часть о том, чего можно ожидать далее. Часть о том, что следует делать, если эти ожидания не подтвердятся».

Фреймовая модель по своей организации во многом похожа на семантическую сеть. Она является сетью узлов и отношений, организованных иерархически: верхние узлы представляют общие понятия, а подчиненные им узлы – частные случаи этих понятий. В системе, основанной на фреймах, понятие в каждом узле определяется набором атрибутов-слотов (например, имя, цвет, размер) и значениями этих атрибутов (например, «Запорожец» красный, маленький). Каждый слот может быть связан со специальными процедурами, которые выполняются, когда информация в слотах (значения атрибутов) меняется. С каждым слотом можно связать любое число процедур.

Описание некоторой предметной области в виде фреймов обладает высо-

ким уровнем абстрактности. Фреймовая система не только описывает знания, но и позволяет человеку описывать метазнания, т.е. правила и процедуры обработки знаний, выбора стратегий, приобретения и формирования новых знаний. Модель является универсальной, поскольку существуют не только фреймы для обозначения объектов и понятий, но и фреймы-роли (отец, начальник, пешеход), фреймы-ситуации (тревога, рабочий режим устройства) и др.

Обучение фреймовых систем затруднено. Приобретение новых знаний возможно только в системах со сложной структурой фреймов. Создание таких систем требует серьезных затрат времени и средств, но они позволяют формировать новые знания на уровне понятий. При этом проблема устранения противоречивых знаний должна решаться самой системой. Для хранения элемента модели требуются значительные объемы памяти, определяемые сложностью конкретного фрейма.

Одной из наиболее известных ИС, построенных на основе фреймов, является система *MOLGEN*, предназначенная для планирования экспериментов в области молекулярной генетики. Речь идет о планировании, а не о решении аналитических задач, т.е. невозможно полностью описать цель задачи перед началом ее решения. При такой постановке вопроса редуцировать пространство поиска с помощью простых методов не представляется возможным.

В настоящее время концепция фреймов быстро развивается и расширяется, благодаря развитию методов объектно-ориентированного программирования. Практически во всех современных языках программирования появились специальные структурно-функциональные единицы (объекты, классы), обладающие основными признаками фреймов.

Таким образом, каждая из известных моделей представления знаний обладает как минимум тремя недостатками из приведенного списка: недостаточный универсализм, сложность получения новых знаний, возможность получения противоречивых знаний; сложность наращивания модели, значительная размерность модели, отсутствие наглядности в представлении знаний.

Именно поэтому в последнее время значительное внимание в инженерии знаний уделяется сочетанию разных моделей.

Гибридные системы поддержки решений объединяют методологии экспертных систем, теории нечетких множеств, ИНС, генетических алгоритмов и др. Примером гибридных систем является обучение ИНС при помощи генетических алгоритмов. При этом обеспечивается высокая скорость подстройки весов сети, достаточно малые ошибки обучения.

Пример. Продукционная модель знаний. Представление знаний в виде правил.

Для предметной области ***Видеохостинг***, нужно сформировать базу знаний, соответствующую следующим требованиям:

- включить не менее 12 правил, из которых не менее 7 – сложные правила;
- для описания правил использовать, не менее 8 переменных;
- число циклов просмотра правил для прямой цепочки рассуждений должно составлять не менее 3;
- для обратной цепочки рассуждений должны быть логически выведены не менее 4 переменных, прежде чем будет определена переменная вывода.

Решение

Переменные:

НС – находится на сайте (да/нет),

ФРЕГ – форма регистрации (заполнена/не заполнена),

ОФРЕГ – отправка формы регистрации (да/нет),

РЕГ – зарегистрирован (да/нет),

ВЛОГ – возможность войти на сайт под логином и паролем (активно/неактивно),

ЛОГ – вошел на сайт под логином и паролем (да/нет),

ФПСК – форма поиска (активна/неактивна),

ФРПСК – форма расширенного (активна/неактивна),

ВМ – видеоматериал (загружен/не загружен),

ЗВМ – загрузить видеоматериал на компьютер (активно/неактивно),

ОВМ – отправить видеоматериал на сервер (доступно/не доступно),

СВМ – свои видеоматериалы на странице профиля (доступно/недоступно),

ПВМ – поделиться видеоматериалом через соцсеть (активно/неактивно),

ДВМ – добавить видеоматериал к профилю (возможно/невозможно),

РВМ – оценить видеоматериал (поставить ранг для видео, поэтому Р типа звездочки от 1 до 5) (активно/неактивно),

КВМ – комментировать видеоматериал (активно/неактивно),

ХШ – хэштеги (есть/нет),

СС – сессия (имеется в виду открытая сессия в соцсети) (открыта/закрыта).

Составлены 13 правил, 9 сложных.

1. Если НС = *да* и РЕГ = *нет*, то ВЛОГ = *неактивно*.
2. Если НС = *да* и РЕГ = *нет*, то ФПСК = *активна*. (Поиск по названию)
3. Если НС = *да* и РЕГ = *нет* И СС = *открыта*, то ПВМ = *активно.*
4. Если ФРЕГ = *заполнена* и ОФРЕГ = *да*, то РЕГ = *да*.
5. Если НС = *да* и РЕГ = *да*, то ВЛОГ = *активно*.
6. Если ЛОГ = *да*, то ЗВМ = *активно*.
7. Если ЛОГ = *да*, то КВМ = *активно*.
8. Если ЛОГ = *да*, то ФРПСК = *активно*. (Поиск по хэштегам)
9. Если ВМ = *загружен* и ХШ = *есть*, то ОВМ = *доступно*.
10. Если ЛОГ = *да* и ВМ = *загружен*, то СВМ = *активно*.
11. Если ЛОГ = *да*, то ДВМ = *возможно*.
12. Если ЛОГ = *да* и СС = *открыта*, то ПВМ = *активно*.
13. Если ЛОГ = *да*, то РВМ = *активно*.

Замечание. В чем разница между 3 и 12 правилами: в обоих случаях можно поделиться видеоматериалами через соцсеть, но, возможно, во втором есть какие–то возможности для каждого профиля на хостинге. И у них могут

быть разные функциональные возможности – с того же ютуба можно поделиться ссылкой и не регистрируясь на ютубе.

Общая схема алгоритма прямой цепочки рассуждений:

1. Определить исходную переменную, запомнить ее значение.
2. Установить признак продолжения цикла в значение «ложь».
3. Сделать первое правило текущим.
4. Если текущее правило простое, то перейти к шагу 6.
5. Если в условной части правила один факт $F1$ истинен и содержится другой факт $F2$, в котором содержится неопределенная переменная, то запросить значение переменной из факта $F2$ у пользователя.
6. Если условная часть правила истинна и переменная из заключительной части не определена, то присвоить значение переменной, исключить правило из дальнейшего рассмотрения и установить признак продолжения цикла в значение «истина».
7. Если не достигнуто последнее правило в БЗ, то сделать следующее правило текущим и вернуться к шагу 4.
8. Если все переменные определены, то перейти к шагу 10.
9. Если признак продолжения цикла имеет значение «истина», то вернуться к шагу 2.
10. Сообщить пользователю окончательный вывод.
11. Конец алгоритма.

Алгоритм прекращает работу, если выполняется одно из трех условий:

1) все значения переменных определены;
2) при переборе правил в БЗ ни одно из правил не было исключено из рассмотрения;
3) все правила исключены из рассмотрения.

Приведем пошаговый вывод, реализующий алгоритм прямой цепочки рассуждений на основе базы знаний. При этом нужно предусмотреть пошаговый логический вывод на экран следующей информации:

– факты, которые были определены пользователем;

– факты, которые выведены из правил (с указанием номеров правил);

– окончательный логический вывод, полученный экспертной системой.

Решение

Перед началом работы алгоритма все переменные неопределенны:

1	2	3	4	5	6	7	8	9
НС	ФРЕГ	ОФРЕГ	РЕГ	ВЛОГ	ЛОГ	ФПСК	ФРПСК	ВМ
NIL	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*

10	11	12	13	14	15	16	17	18
ЗВМ	ОВМ	СВМ	ПВМ	ДВМ	РВМ	КВМ	ХШ	СС
NIL	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*

Вошел на сайт под логином и паролем (ЛОГ=*да*) – регистрации в соцсети нет (РЕГ=*нет*)

1	2	3	4	5	6	7	8	9
НС	ФРЕГ	ОФРЕГ	РЕГ	ВЛОГ	ЛОГ	ФПСК	ФРПСК	ВМ
NIL	*NIL*	*NIL*	**Нет**	*NIL*	**Да**	*NIL*	*NIL*	*NIL*
			Факт1		Факт2			

10	11	12	13	14	15	16	17	18
ЗВМ	ОВМ	СВМ	ПВМ	ДВМ	РВМ	КВМ	ХШ	СС
NIL	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*

Начинаем просмотр правил из БЗ и поиск тех правил, в условной части которых встречается или факт 1 или факт 2. Таким правилом является П1. В П1, помимо Ф1 и Ф2 есть факт, содержащий переменную РЕГ.

Значение этой переменной запрашиваем у пользователя:

– *Вы находитесь на сайте?*

1-й цикл просмотра правил в БЗ.

Допустим, он отвечает:

– *Да.*

Таким образом, получаем факт 3: НС=*Да*.

Допустим, пользователь отвечает:

–Нет.

Таким образом, получаем факт 3: Н= *Нет*.

1	2	3	4	5	6	7	8	9
НС	ФРЕГ	ОФРЕГ	РЕГ	ВЛОГ	ЛОГ	ФПСК	ФРПСК	ВМ
Да	*NIL*	*NIL*	**Нет**	*NIL*	**Да**	*NIL*	*NIL*	*NIL*
Факт 3			Факт 1		Факт 2			

10	11	12	13	14	15	16	17	18
ЗВМ	ОВМ	СВМ	ПВМ	ДВМ	РВМ	КВМ	ХШ	СС
NIL	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*

Условная часть П1 признается истиной, получаем факт 4: ВЛОГ=*неактивно*. Исключаем правило из дальнейшего рассмотрения.

1	2	3	4	5	6	7	8
НС	ФРЕГ	ОФРЕГ	РЕГ	ВЛОГ	ЛОГ	ФПСК	ФРПСК
Да	*NIL*	*NIL*	**Нет**	**Неактивно**	**Да**	*NIL*	*NIL*
Факт 3			Факт 1	Факт 4	Факт 2		

9	10	11	12	13	14	15	16	17	18
ВМ	ЗВМ	ОВМ	СВМ	ПВМ	ДВМ	РВМ	КВМ	ХШ	СС
NIL	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*

Продолжаем просмотр правил.

Факт 1 и факт 4 встречаются в условной части П2. Так как переменная вывода П2 ФПСК не определена, то присваиваем ей значение ФПСК=*активна* (факт 5), исключаем П2 из дальнейшего рассмотрения.

1	2	3	4	5	6	7	8
НС	ФРЕГ	ОФРЕГ	РЕГ	ВЛОГ	ЛОГ	ФПСК	ФРПСК
Да	*NIL*	*NIL*	**Нет**	**Неактивно**	**Да**	**Активна**	*NIL*
Факт 3			Факт 1	Факт 4	Факт 2	Факт 5	

9	10	11	12	13	14	15	16	17	18
ВМ	ЗВМ	ОВМ	СВМ	ПВМ	ДВМ	РВМ	КВМ	ХШ	СС
NIL	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*

Продолжаем просмотр правил. Факт 2 встречается в П6. Это простое пра-

вило. Так как переменная вывода П6 не определена, то присваиваем ей значение ЗВМ=*активно*. Получаем факт 6.

Факт 2 также встречается в П7. Это простое правило. Так как переменная вывода П7 не определена, то присваиваем ей значение КВМ=*активно* (факт 7).

Факт 2 снова встретился нам в П8. Это простое правило. Так как переменная вывода П8 не определена, то присваиваем ей значение ФРПСК=*активно*. Получаем факт 8.

Факт 2 встречается в П11. Это простое правило. Так как переменная вывода П11 не определена, то присваиваем ей значение ДВМ=*возможно* (факт 9).

Факт 2 также встречается в П13. Это простое правило. Так как переменная вывода П13 не определена, то присваиваем ей значение РВМ=*активно* (факт 10).

Исключаем правила 6, 7, 8, 11 и 13 из дальнейшего рассмотрения.

1	2	3	4	5	6	7	8
НС	ФРЕГ	ОФРЕГ	РЕГ	ВЛОГ	ЛОГ	ФПСК	ФРПСК
Да	*NIL*	*NIL*	**Нет**	**Неактивно**	**Да**	**Активна**	**Активна**
Факт 3			Факт 1	Факт 4	Факт 2	Факт 5	Факт 8

9	10	11	12	13	14	15	16	17	18
ВМ	ЗВМ	ОВМ	СВМ	ПВМ	ДВМ	РВМ	КВМ	ХШ	СС
NIL	**Активна**	*NIL*	*NIL*	*NIL*	**Возможно**	**Активна**	**Активна**	*NIL*	*NIL*
	Факт 6				Факт 9	Факт 10	Факт 7		

2-й цикл просмотра правил в БЗ

Просмотр правил начинаем с первого, с учетом тех фактов, которые были получены на предыдущих шагах алгоритма.

Факт 1 и факт 3 встречаются в условной части П3. Помимо Ф1 и Ф1, условная часть П3 имеет переменную СС. Значение этой переменной запрашиваем у пользователя:

– *У Вас открыта сессия в социальной сети?*

Допустим, пользователь отвечает:

– *Открыта*.

Таким образом, получаем факт 11: СС=*открыта*.

1	2	3	4	5	6	7	8	9
НС	ФРЕГ	ОФРЕГ	РЕГ	ВЛОГ	ЛОГ	ФПСК	ФРПСК	ВМ
Да	*NIL*	*NIL*	**Нет**	**Неактивно**	**Да**	**Активна**	**Активна**	*NIL*
Факт 3			Факт 1	Факт 4	Факт2	Факт 5	Факт 8	

10	11	12	13	14	15	16	17	18
ЗВМ	ОВМ	СВМ	ПВМ	ДВМ	РВМ	КВМ	ХШ	СС
Активна	*NIL*	*NIL*	*NIL*	**Возможно**	**Активна**	**Активна**	*NIL*	**Открыта**
Факт 6				Факт 9	Факт 10	Факт 7		Факт 11

Условная часть П3 признается истиной. Мы получаем переменную со значением ПВМ=*активно*. Это факт 12. Исключаем П3 из дальнейшего рассмотрения.

Факт 2 встречается в П10. Но в условной части есть еще одна переменная ВМ. Запросим у пользователя значение переменной ВМ:

– *Загружен у Вас видеоматериал?*

Допустим, пользователь ответил, что загружен. В таком случае получаем ВМ=*загружен* (факт 13). Так как переменная вывода П10 не определена, присваиваем ей значение СВМ=*активно* (факт 14). Исключаем правило 10 из дальнейшего рассмотрения.

Факт 2 и факт 11 встречаются в П12. Там нам встретилась переменная, которая у нас уже определена как *активно*. Здесь она определена так же. Исключаем П12 из дальнейшего рассмотрения.

1	2	3	4	5	6	7	8	9
НС	ФРЕГ	ОФРЕГ	РЕГ	ВЛОГ	ЛОГ	ФПСК	ФРПСК	ВМ
Да	*NIL*	*NIL*	**Нет**	**Неактивно**	**Да**	**Активна**	**Активна**	**Загружен**
Факт 3			Факт 1	Факт 4	Факт 2	Факт 5	Факт 8	Факт 13

10	11	12	13	14	15	16	17	18
ЗВМ	ОВМ	СВМ	ПВМ	ДВМ	РВМ	КВМ	ХШ	СС
Активна	*NIL*	**Активно**	**Активно**	**Возможно**	**Активна**	**Активна**	*NIL*	**Открыта**
Факт 6		Факт 14	Факт 12	Факт 9	Факт 10	Факт 7		Факт 11

Не все переменные получили свои значения, но мы обратились к пользователю 3 раза, это – наш критерий-останова, следовательно, прямая цепочка рассуждений закончена.

Факты, которые определены пользователем:

Факт 3 (НС[находится на сайте]=*да*)

Факт 11 (СС[сессия]=*открыта*)

Факт 13 (ВМ[видеоматериал]=*загружен*)

Факты, которые выведены из правил:

Факт 4 (ВЛОГ[возможность войти на сайт под логином и паролем]=*неактивно*)

Факт 5 (ФПСК[форма поиска]=*активна*)

Факт 6 (ЗВМ[загрузить видеоматериал]=*активно*)

Факт 7 (КВМ[комментировать видеоматериал] =*активно*)

Факт 8 (ФРПСК[форма расширенного поиска]=*активно*)

Факт 9 (ДВМ[добавление видеоматериала]=*возможно*)

Факт 10 (РВМ[оценить видеоматериал]=*активно*)

Факт 12 (ПВМ[поделиться видеоматериалом через социальную сеть]=*активно*)

Факт 14 (СВМ[свои видеоматериалы на странице]=*активно*).

Общая схема алгоритма обратной цепочки рассуждений:

1. Определить переменную логического вывода.
2. Найти правило, заключительная часть которого содержит переменную вывода.
3. Если такое правило не найдено, то сообщить пользователю, что вывод невозможен, и перейти к шагу 14, иначе поместить правило в стек.
4. Если переменная, соответствующая номеру условия правила в вершине стека, определена, то увеличить номер условия на 1 и перейти к шагу 8.
5. Найти правило, в заключительной части которого встречается переменная, соответствующая номеру условия.

6. Если правило не найдено или предыдущий вывод неверен (см. шаг 9), то запросить значение переменной у пользователя, увеличить номер условия на 1 и перейти к шагу 8.
7. Поместить найденное правило в стек и вернуться к правилу 4.
8. Если номер условия меньше или равен числу фактов в условной части правила, то вернуться к шагу 4.
9. Если условная часть истинна, то присвоить значение переменной из заключительной части правила, иначе предыдущий вывод неверен.
10. Удалить правило из стека.
11. Если переменная вывода определена, то перейти к шагу 13.
12. Если стек пуст, то вернуться к шагу 2.
13. Сообщить пользователю окончательный вывод.
14. Конец алгоритма.

Алгоритм прекращает работу, если выполняется одно из двух условий:

1. Значение переменной вывода определено.
2. При полученных значениях переменных значение переменной вывода получить невозможно.

Приведем пошаговый вывод, реализующий алгоритм обратной цепочки рассуждений на основе базы знаний. При этом нужно предусмотреть пошаговый логический вывод на экран следующей информации:

- Факты, которые были определены пользователем;
- Факты, которые выведены из правил (с указанием номеров правил);
- Содержимое стека правил (при пошаговом выводе);
- Окончательный логический вывод, полученный экспертной системой.

Решение

Работа алгоритма начинается с задания пользователем переменной логического вывода в форме запроса:

– *Поделиться видеоматериалом?*

Следовательно, ЭС необходимо определить значение переменной ПВМ.

В базе знаний производится поиск правил, в заключительной части которых находится переменная ПВМ. Первое такое правило П3 помещается в стек правил:

П3	1(НС)

Определяем переменные из условной части правила П3. Первая переменная – НС (на сайте). В БЗ производим поиск правил, в заключительной части которых встречается переменная НС. Переменная НС не определена и нигде не встречается, значит, спрашиваем о ней у пользователя:

– *Находитесь на сайте?*

Допустим, что пользователь ответил:

– *Да*.

Имеем: НС = *да*. Номер условия правила П3 увеличиваем на 1, что соответствует переменной РЕГ (зарегистрирован).

П3	2(РЕГ)

В БЗ производим поиск правил, в заключительной части которых встречается переменная РЕГ. Находим ее в П4, помещаем это правило на вершину стека:

П4	1(ФРЕГ)
П3	2(РЕГ)

Определяем переменные из условной части правила П4. Первая переменная – ФРЕГ (форма регистрации). В БЗ производим поиск правил, в заключительной части которых встречается переменная ФРЕГ. Переменная ФРЕГ не определена и нигде не встречается, значит, спрашиваем о ней у пользователя:

– *ФРЕГ?*

Допустим, что пользователь ответил:

– *Да*.

Имеем: НС = *да*, ФРЕГ = *да*. Номер условия правила П4 увеличиваем на 1, что соответствует переменной ОФРЕГ (отправка формы регистрации):

П4	2(ОФРЕГ)
П3	2(РЕГ)

В БЗ производим поиск правил, в заключительной части которых встречается переменная ОФРЕГ. Переменная ОФРЕГ не определена и нигде не встречается, значит, спрашиваем о ней у пользователя:

– *ОФРЕГ?*

Допустим, что пользователь ответил:

– *Да*.

Имеем: НС = *да*, ФРЕГ = *да*, ОФРЕГ = *да*. Признаем условную часть П4 истинной, значение РЕГ определяем, как да. Убираем П4 из стека.

Имеем: НС = *да*, ФРЕГ = *да*, ОФРЕГ = *да*, РЕГ = *да*.

Номер условия правила П3 увеличиваем на 1, что соответствует переменной СС (сессия в соцсети).

П3	3(СС)

В БЗ производим поиск правил, в заключительной части которых встречается переменная СС. Переменная СС не определена и нигде не встречается, значит, спрашиваем о ней у пользователя:

– *СС?*

Допустим, что пользователь ответил:

– *Да*.

Имеем: НС = *да*, ФРЕГ = *да*, ОФРЕГ = *да*, РЕГ = *да*, СС = *да*. Признаем условную часть П3 истинной, значение ПВМ определяем, как *активно*.

Необходимое значение переменной было установлено, значит, обратная цепочка рассуждений завершена.

Факты, которые были определены пользователем:

НС = *да* – ФРЕГ = *заполнена*;

ОФРЕГ = *да* – СС = *открыта*.

Факты, которые были выведены:

Правило 3: РЕГ = *да*.

Вопросы для самопроверки

1. Экспертные системы. Их отличия от других ИИС. Структура ЭС.
2. Классификации экспертных систем. Информационные технологии ЭС.
3. Информация, данные, знания. Свойства знаний. Классификация знаний.
4. Модели представления знаний. Классификация моделей представления знаний. Формальные логические модели.
5. Продукционная модель знаний. Механизмы логического вывода в продукционной модели.
6. Представление знаний в виде семантической сети. Вывод на семантических сетях.
7. Фреймовые модели знаний. Организация логического вывода во фреймовой модели.

Задания для самостоятельной работы

Задание 1. Создание информационной базы

1. Используя *MS Word*, нужно создать таблицу «Термины» со следующими полями: Порядковый номер, Термин, Код термина, Предметная область, Код предметной области.

2. Заполнить столбец Порядковый номер и Термин (набрав 30 терминов из разных предметных областей).

3. Присвоить терминам коды.

Указание. Каждой из предметных областей присваивается код из двух десятичных цифр, первая из которых не должна быть нулем. После этого код термина строится из шести десятичных цифр, первые две из которых представляют собой код предметной области, а четыре оставшихся – порядковый номер данного термина в данной предметной области. Скопируйте таблицу «Термины» в соответствующий диапазон рабочего листа табличного процессора *Excel* с тем же названием. На рабочем листе *Excel* рекомендуется:

- пронумеровать строки таблицы;

- указать названия столбцов в первой строке таблицы.

4. Импортируйте диапазон данных таблицы «Термины» табличного процессора *Excel* в таблицу СУБД *Access*:

- Откройте в СУБД *Access* новую базу данных под названием *Inf.mdb*.
- Используя средства Импорт, загрузите ранее созданную таблицу «Термины».
- Создайте новую таблицу «Предметные области», содержащую столбцы Назв_Пр_Обл и Код_Пр_обл, которые заполняются вручную.
- В таблице «Термины» помещается Код_Предметной_области. Соответствующее название будет извлекаться по связи между таблицами через поле Код_Предметной_области (эту связь необходимо установить).

5. После построения в БД *Inf.mdb* двух указанных таблиц дополните таблицу «Термины» таким образом, чтобы по каждой из предметных областей имелось не менее 12 терминов. Затем необходимо построить 2 запроса:

1) по заданному термину найти соответствующую предметную область;

2) по заданной предметной области найти все принадлежащие ей термины.

Задание 2. Формальные логические модели

1. Запишите логической формулой следующие умозаключения и уточните их справедливость:

A) Увеличение первоначального капитала в условиях инфляции произойдет, если коэффициент превысит индекс цен. Коэффициент наращения меньше индекса цен. Следовательно, увеличение капитала не произойдет.

B) Увеличение первоначального капитала в условиях инфляции произойдет, если коэффициент превысит индекс цен. Произошло увеличение капитала. Следовательно, коэффициент наращения больше индекса цен.

2. Запишите логической формулой следующие умозаключения и уточните их справедливость:

Если у фирмы сломалось оборудование, то она либо купит новое, либо бывшее в употреблении. Новое оборудование фирма приобретет, если у нее есть лишние деньги, в противном случае придется покупать более дешевое, бывшее в употреблении. У фирмы нет лишних денег в день выдачи зарплаты:

А) Сегодня день выдачи зарплаты, следовательно, фирма приобретет бывшее в употреблении оборудование.

Б) Фирма приобрела новое оборудование, следовательно, сегодня не день выдачи зарплаты.

3. Запишите логической формулой следующие умозаключения и уточните их справедливость:

Семья может заплатить за свет либо по счетчику, либо по среднему тарифу. Оплата по счетчику произойдет, если представитель придет 1-го числа и кто-нибудь из семьи будет в это время дома. Оплата света в этом месяце производилась по среднему. Следовательно, 1-го числа никого не было дома.

Задание 3. Продукционная модель знаний, представление знаний в виде правил

Для выбранной самостоятельно предметной области сформировать базу знаний, соответствующую следующим требованиям:

- включить не менее 12 правил, из которых не менее 7 – сложные правила;
- для описания правил использовать не менее 8 переменных;
- число циклов просмотра правил для прямой цепочки рассуждений должно составлять не менее 3;
- для обратной цепочки рассуждений должны быть логически выведены не менее 4 переменных, прежде чем будет определена переменная вывода;
- пару последовательных правил.

Отчет о выполненной работе должен содержать:

1. Перечисление переменных, их описание и принимаемые ими возможные значения.
2. Правила, составляющие базу знаний.

Задание 4. Прямая цепочка рассуждений

Разработать экспертную систему, реализующую алгоритм прямой цепочки рассуждений на основе базы знаний, разработанной в задании 3. Предусмотреть пошаговый логический вывод на экран следующей информации:

- факты, которые были определены пользователем;
- факты, которые выведены из правил (с указанием номеров правил);
- окончательный логический вывод, полученный экспертной системой.

Отчет о выполненной работе должен содержать:

1. Правила, составляющие базу знаний;
2. Общую схему алгоритма прямой цепочки рассуждений;
3. Пошаговый вывод, полученный с помощью разработанной экспертной системы.

Задание 5. Обратная цепочка рассуждений

Разработать экспертную систему, реализующую алгоритм обратной цепочки рассуждений на основе базы знаний, разработанной в задании 3. Предусмотреть автоматический и пошаговый логический вывод. Разработанная экспертная система должна обеспечивать вывод на экран следующей информации:

– факты, которые были определены пользователем;

– факты, которые были выведены из правил (с указанием номеров правил);

– содержимое стека правил (при пошаговом выводе);

– окончательный логический вывод, полученный экспертной системой.

Отчет о выполненной работе должен содержать:

1. Правила, составляющие базу знаний;
2. Общую схему алгоритма обратной цепочки рассуждений;
3. Пошаговый вывод, полученный с помощью разработанной экспертной системы.

Список литературы

1. Барсегян А.А. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP/ А.А. Барсегян – СПб.: БХВ-Петербург, 2007. – 284 с.
2. Болотова Л.С. Системы и методы искусственного интеллекта: модели и технологии, основанные на знаниях/ Л.С. Болотова. – М.: Финансы и статистика, 2012. – 664 с.
3. Паклин Н. Б. Бизнес-аналитика: от данных к знаниям /Н.Б.Паклин, В.И. Орешков – СПб.: Питер, 2010. – 704 с.
4. Практикум по анализу данных на компьютере / И.А. Кацко, Н.Б. Паклин /Под ред. Г.В. Гореловой – М.: КолосС, 2009. – 278 с.
5. Матвеев М.Г. Модели и методы искусственного интеллекта/ М.Г. Матвеев, А.С. Спиридонов, Н.А. Алейникова. – М.: Финансы и статистика; ИНФРА-М, 2008. – 448 с.
6. Data Mining – добыча данных/ BaseGroup Labs. Режим доступа.– http://www.basegroup.ru/library/methodology/data_mining/

Прокопенко Наталья Юрьевна

СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

на базе Deductor Studio Academic 5.3

Учебное пособие

Редактор:
Н.А.Воронова

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Нижегородский государственный архитектурно-строительный университет»
603950, Нижний Новгород, ул. Ильинская, 65.
http://www. nngasu.ru, srec@nngasu.ru