Кириченко А.А.

Нейропакеты. Лекции.

2016

Кириченко А.А.

«Нейропакеты. Лекции», 2016. Сетевое электронное издание учебного пособия. 248 страниц, формат PDF.

ISBN 978-5-9904911-5-1

Нейрокомпьютинг предоставляет единую методологию решения очень широкого круга практически интересных задач, как правило - ускоряющих и удешевляющих разработку приложений. В число таких задач входят прогнозирование цен, оценка кредитоспособности, оптическое распознавание, обработка изображений, диагностика, лингвистический анализ, и др.

Использование нейросетей для решения перечисленных задач предусматривает выполнение типовой последовательности действий с помощью нейрокомпьютеров, или нейропакетов.

В лекциях рассматриваются возможности нейропакетов, использование их для моделирования разнообразных нейронных систем, настройка и обучение универсальных нейропакетов на решение задач с использованием наиболее доступных пакетов: Deductor Academic, Sharky, Matlab, Пермский нейросимулятор, MemBrain и др .

Лекции читались в 2016г. студентам, обучавшимся на майноре «Нейросетевые технологии» в Высшей школе экономики.

Кириченко А.А. профессор Департамента программной инженерии Факультета компьютерных наук Федерального государственного автономного образовательного учреждения высшего профессионального образования "Национальный исследовательский университет "Высшая школа экономики" при правительстве РФ”.

ISBN 978-5-9904911-5-1

Содержание.

Лекция 1. Архитектура нейросетевого исследования.

Что собой представляет нейросетевое исследование?

От ответа на этот вопрос зависит очень многое. Начиная от того, какие задачи решаются с помощью нейронных сетей, как для такого исследования готовится информация, как должна выглядеть нейронная сеть, и вплоть до того, как должен выглядеть нейропакет – программа, с помощью которой проводится нейросетевое исследование.

Неправильное представление об архитектуре нейросетевого исследования приводит к тому, что в нейропакет включаются несвойственные ему функции, чаще всего – заимствованные из дискретных методов математической статистики, сам пакет обычно содержит хорошо продуманный блок обучения нейросети и не имеет средств для решения задачи после окончания обучения, не говоря о том, что нейропакет должен содержать средства для анализа эффективности обучения, для исследования качества получаемого решения, для оценки степени влияния различных исходных данных на результат решения.

Проведение научного исследования чаще всего заключается в выявлении скрытых правил и закономерностей в наборах данных, формулировке гипотез и выявлении типовых структур (паттернов). Для этого приходится использовать различные методы обнаружения (добычи) знаний: абстрагирование, ассоциативное объединение, классификацию, кластеризацию, анализ временных рядов, прогнозирование и др.

Человеческий разум не приспособлен для восприятия больших массивов разнородной информации. В среднем человек не способен улавливать более двух-трех взаимосвязей даже в небольших выборках.

Для расширения аналитических возможностей человека можно использовать методы традиционной статистики, эвристические решающие устройства на основе экспертных систем, семантический дифференциал, теорию решения изобретательских задач (ТРИЗ), нейронные сети.

Традиционная статистика решает аналогичные задачи, но она оперирует усредненными характеристиками выборки, которые часто являются фиктивными величинами, например - средней платежеспособностью клиента, в то время, как необходимо уметь прогнозировать состоятельность и намерения конкретного клиента с учётом функции риска или функции потерь.

Методы математической статистики, эвристические решающие устройства, семантический дифференциал, ТРИЗ, так же, как и статистика относятся к дискретным методам. Для человека же несвойственно использовать при решении жизненных проблем дискретные методы.

Естественным для человека является использование основных принципов мозга — ассоциативное мышление, использование принципов обучения (самообучения) и адаптации, связей «если - то», «посылка - следствие», лежащих в основе распознавания, движения, управления, принятия решений.

Поэтому из различных способов расширения аналитических возможностей человека наиболее эффективными при исследовании задач, не имеющих общепризнанного алгоритма решения, является использование нейронных сетей.

Всё, что связано с использованием нейронных сетей получило название нейросетевых технологий.

Нейросетевые технологии не требуют программирования, а предусматривают работу по обучению нейронной сети на специально подобранных примерах.

Основной функцией обучения нейросети, воспроизводящей работу мозга и ассоциативное мышление является узнавание, умение определять сходство и различия.

На этапе обучения формируются основные отношения между входными параметрами и оформляются в незримые таблицы (образы), которые впоследствии будут использоваться при решении задач на сети.

Нейросети наиболее приспособлены к решению широкого круга задач, так или иначе связанных с обработкой образов. Вот список типичных математических постановок задач для нейросетей:

1. Узнавание (Классификация)
2. Кластеризация
3. Регрессия (прогнозирование, предсказание)
4. Понижение размерности

В задачах регрессии целью является оценка значения числовой (принимающей непрерывный диапазон значений) выходной переменной по значениям входных переменных.

В задачах анализа временных рядов целью является **прогноз** будущих значений переменной, зависящей от времени, на основе предыдущих значений ее и/или других переменных.

Как правило, прогнозируемая переменная является числовой, поэтому прогнозирование временных рядов - это частный случай регрессии. Однако такое ограничение часто в пакет не закладывается, так что в нем можно прогнозировать и временные ряды номинальных (т.е. классифицирующих) переменных.

В задаче узнавания сеть должна отнести каждое наблюдение к одному из нескольких классов (или, в более общем случае, оценить вероятность принадлежности наблюдения к каждому из классов).

Задачи классификации (кластеризации) заключаются в группировке похожих образов в классы (кластеры). В нейропакетах они решаются с помощью сети СОКК (самоорганизующаяся карта Кохонена), или так называемой радиальной базисной функцией (radial basis function - RBF)

Самоорганизующаяся карта Кохонена (СОКК или сеть Кохонена) принципиально отличается от всех других типов сетей. В то время как все остальные сети предназначены для задач с управляемым обучением (т.е обучением с учителем), сети Кохонена главным образом рассчитаны на неуправляемое обучение (без учителя).

В алгоритмах неуправляемого обучения значения весов и/или порогов меняются на основании только входных обучающих данных (выходные значения не требуются и, если они присутствуют, игнорируются).

Этот список можно было бы продолжить и дальше. За ними просматривается некий единый прототип, позволяющий при известной доле воображения сводить их друг к другу. Чаще всего - это типичные примеры некорректных задач, т.е. задач не имеющих единственного решения, алгоритмы которых неизвестны или не имеют строго обоснования.

Нейрокомпьютинг предоставляет единую методологию решения очень широкого круга практически интересных задач. Это, как правило, ускоряет и удешевляет разработку приложений.

В число таких задач входят прогнозирование цен, оценка кредитоспособности, оптическое распознавание (например, подписи), обработка изображений, диагностика, лингвистический анализ, и др.

Нейросети - это не что иное, как новый инструмент анализа данных.

И лучше других им может воспользоваться именно специалист в своей предметной области.

Основные трудности на пути еще более широкого распространения нейротехнологий - в неумении широкого круга профессионалов формулировать свои проблемы в терминах, допускающих простое нейросетевое решение, т.е. неумение проводить нейросетевую декомпозицию решаемой задачи.

Поскольку основные задачи, решаемые с помощью нейропакетов, связаны с поиском скрытых закономерностей, классификацией, прогнозированием, сокращением размерности, область их применения охватывает все направления человеческих знаний – и технических, и гуманитарных.

Изучение возможностей нейропакетов, моделирование разнообразных нейронных систем, разработка программ для автоматизации выполнения интеллектуальных операций, настройка и обучение универсальных нейропакетов на решение определённых задач – вот далеко не полный перечень задач, стоящих перед бакалаврами и магистрами национального исследовательского университета.

Нейросетевое исследование предусматривает выполнение следующих этапов:

1. Подготовка исходных данных.
2. Разметка исходных данных.
3. Формирование нейронной сети.
4. Предварительное обучение сети.
5. Анализ результатов обучения.
6. Оптимизация обучения сети.
7. Анализ подготовленных для исследования данных.
8. Проведение нейросетевого исследования.

Подготовка данных для исследования с помощью нейронной сети.

Данные в нейронной сети используются прежде всего – для обучения. С этой целью они оформляются в виде обучающей выборки, или обучающего набора данных.

Например, сеть необходимо обучить классификации на два класса по косвенным признакам или обучить прогнозированию.

Примерами таких задач могут служить следующие:

- «Мужчина/женщина»,
- «Ирисы Фишера»
- «Как выбирают американских президентов»
- и др.

В задаче «Мужчина/женщина» в общем виде обучающий набор данных может представлять собой таблицу вида:

Информационные (inf)	Исходные показатели (Input)				Результирующий (Pattern)
	Готовите ли вы дома пищу	Как часто вы убираете квартиру	Сколько времени в неделю вы тратите на ремонт автомобиля	Является ли только ваш заработок основным источником дохода семьи	
1	нет	редко	3 часа	да	м
2	Да	всегда	0	нет	ж
…	…	…	…	…	…

Разметка исходных данных.

При разметке исходных данных создаются тренировочный, тестовый и экзаменационный наборы данных. Тренировочный набор данных содержит обучающую выборку, он используется для обучения нейронной сети.

Тестирование необходимо для проверки создания обобщённых образов, которые были представлены в обучающей выборке данных.

Для адекватной оценки качества полученной сети используется ещё один совершенно независимый набор данных, называемый обычно экзаменационным. В процессе обучения сети, определения момента остановки обучения, определения дополнительных параметров и др. этот набор вообще никак не используется, т.е. при последующем применении сеть "видит" его впервые. Поэтому качество результатов, показанных сетью на экзаменационном наборе, является хорошей оценкой качества результатов, которые сеть сможет показать на новых, неизвестных ей данных. Разумеется, экзаменационный набор также должен быть представительным.

Программы пакета, предназначенные для решения задач классификации, позволяют задавать все три набора данных, как по отдельности, так и путём установления доли каждого из наборов в общем количестве данных. При применении сети статистика рассчитывается отдельно для каждого класса и каждого набора.

Формирование нейронной сети.

Формирование нейронной сети: определение типа сети, количества слоёв, количества нейронов во входном, выходном и промежуточных слоях, и др.

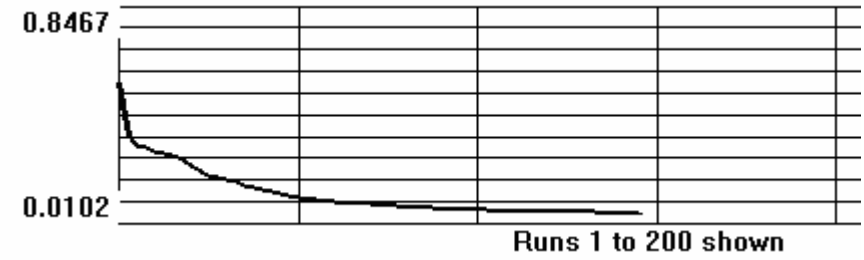
Установка параметров нейронной сети.

Состав параметров определяется в зависимости от используемого вида нейронной сети. Так, для многослойного перцептрона могут быть определены:

- допустимая погрешность обучения (training tolerance);
- допустимая погрешность тестирования (testing tolerance);
- условия остановки обучения:
    1. после окончания N эпохи;
    2. при отсутствии ошибок во время выполнения всей тестирующей выборки;
    3. при достижении M% правильных результатов обучающей выборки;
    4. при уменьшении ошибки сети ниже K%;
- особенности процесса тестирования:
    - тестирование сети после прохождения каждых k эпох;
    - или после каждых p обучающих примеров;
- особенности сохранения результатов (например, после завершения каждых r эпох);
- должна ли быть создана и сохранена в файле начальная матрица коэффициентов связи перцепторов с сумматором.
- и т.д.

Предварительное обучение, анализ результатов и оптимизация обучения сети

Проводится предварительное обучение, анализ результатов и оптимизация обучения сети. В процессе обучения сети ей на вход подаётся один из примеров обучающей выборки и фиксируется результат (output) на выходе сети. Поскольку сеть ещё не обучена, этот результат отличается от желаемого (target). Величина отличия называется ошибкой сети. Сначала она имеет большую величину, по мере обучения – уменьшается:

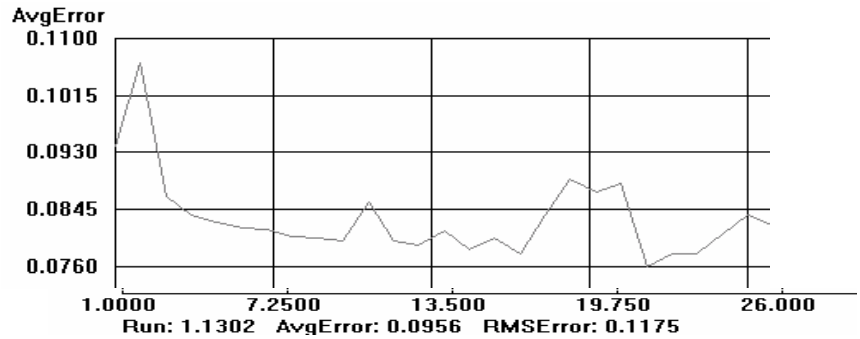


Процесс обучения сети заключается в изменении весов связей перцепторов с сумматором таким образом, чтобы ошибка сети уменьшилась.

Когда весь запас обучающих примеров исчерпан, считается, что закончилась одна эпоха. Подсчитывается средняя ошибка за эпоху и примеры из обучающей выборки снова используются для обучения той же слегка обученной сети – это вторая эпоха.

На приведенном рисунке приведен график, отражающий результаты 200 эпох.

После каждых k эпох обучения фиксируются результаты, и вместо обучающей выборки на вход сети подаются примеры из тестирующей выборки. По ним так же определяется ошибка сети, но обучение сети не проводится. Когда примеры из тестирующей выборки исчерпаны, определяется средняя ошибка тестирования. График изменения ошибки сети при тестировании:



Демонстрирует явление, которое называется «переучиванием»: т.е. достижение такого состояния сети, в котором сеть со 100% точностью распознаёт все примеры тренировочного набора данных, однако на других данных, не входивших в этот набор, может давать большую погрешность. Можно сказать, что такая сеть плохо обобщает. Из графика видно, что где-то между 7 и 13 эпохами есть точка, после которой ошибка сети резко возрастает. Это значит, что после эпохи, в которой получена эта точка, продолжать обучение по той же обучающей выборке бесполезно. Прекращение тренировки при минимуме ошибки на тестовом наборе позволяет зафиксировать сеть в том состоянии, когда она уже усвоила наиболее существенную информацию, содержащуюся в данных, и неплохо обобщает, однако ещё не успела переучиться.

Можно уточнить полученную информацию по результатам, сохранённым в файле при обучении. Читаем значения Run, AvgError и RMSError и заносим их в таблицу:

Run	AvgError	RMSError
3,2135	0,0859	0,1019
3,9947	0,0837	0,1008
4,1250	0,0835	0,1009
4,9062	0,0826	0,1019
6,0781	0,0817	0,1018
7,0546	0,0815	0,1030
8,0312	0,0803	0,1031
9,0072	0,0802	0,1033
9,9192	0,0798	0,1034
10,049	0,0800	0,1037
11,026	0,0853	0,1083

Судя по графику AvgError, улучшение работы сети продолжается до 9 эпохи. На 9-10 эпохе средняя ошибка минимальна. Возрастание её в дальнейшем связано с «переучиванием» сети. Желательно обучение сети на 10 эпохе закончить.

Оптимизация обучения сети заключается в том, что полученные результаты обучения стираются, восстанавливается сохраненная в файле начальная матрица коэффициентов связи перцепторов с сумматором, устанавливается ограничение условия остановки обучения: после окончания 10 эпохи, и снова проводится обучение нейронной сети. Заново обученную сеть можно считать оптимальной, она сохраняется в файле и в дальнейшем может быть использована для решения задач данного типа.

Подготовка данных для исследования на обученной сети.

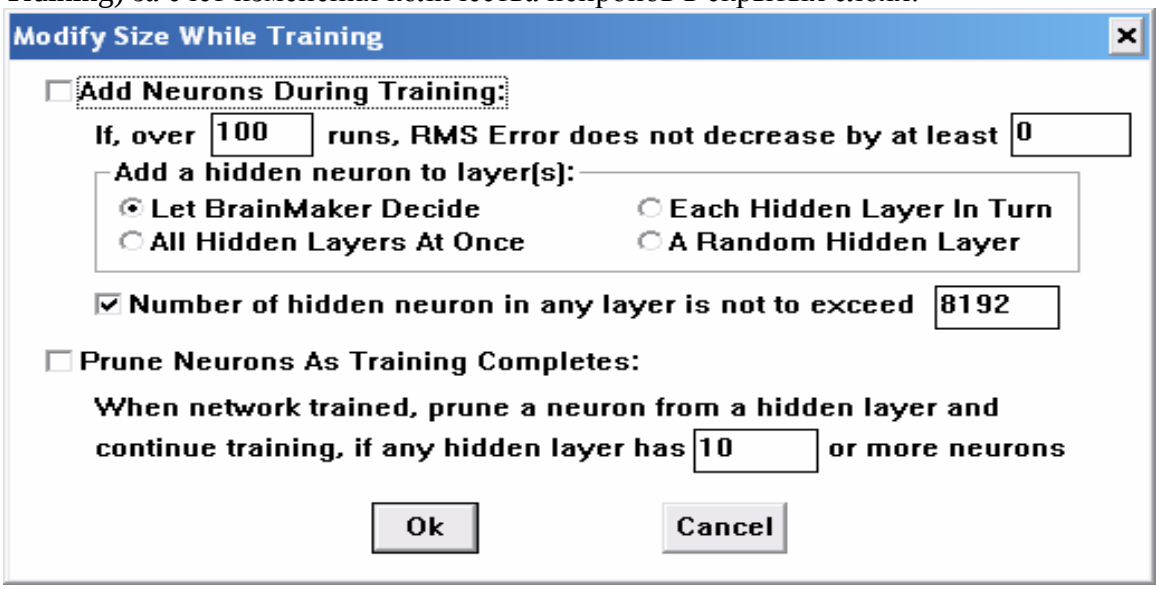
Исследование лучше проводить на заранее подготовленном файле, содержащем исходные данные. Такой файл не должен содержать колонку pattern, результат будет отражаться в колонке out.

Проведение нейросетевого исследования.

В составе нейросетевых пакетов можно использовать имеющиеся средства для наблюдения за работой (например, за процессом обучения).

а. В пакете Brain Maker для наблюдения за ошибкой распознавания можно использовать пункты меню Display и Parameters. Пункт “Network Progress Display” в меню “Display” в открывшемся окне позволяет наблюдать как уменьшается значение ошибки распознавания в процессе обучения.

б. Можно включить модификацию сети во время обучения (Modify Size While Training) за счёт изменения количества нейронов в скрытых слоях:

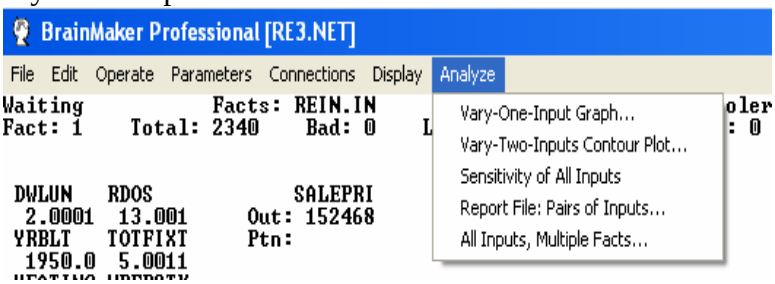


в. Есть ещё одно средство наблюдения - пункт “Show Histograms” в меню “Display” покажет, как нейронная сеть меняет свои внутренние свойства, в частности, теряет, по мере накопления знаний, *способность* обучаться.

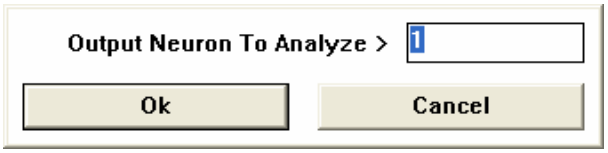
Если после обучения нейросеть эффективно обрабатывает данные обучающего множества, важным становится исследование эффективности работы с данными, которые не использовались для обучения.

В блок анализа нейропакетов обычно входит функция оценки чувствительности для одного входа. Она демонстрирует, насколько данный вход оказывает влияние на выход. Предполагается, что с помощью этой функции можно исследовать поведение нейронной сети с целью выявления мало чувствительных входов или поддиапозонов внутри множества принимаемых входом значений для их последующего исключения в момент проектирования нейронной сети.

Проверка чувствительности всех входов производится через меню Analuze → Sensitivity of All Inputs.



В котором необходимо указать, какой именно нейрон будет исследоваться:



В пакете BM указывается чувствительность входов с точностью +/-10%. Результат виден на графике (термометр) через меню Analyzing. Термометр влево укажет на негативное влияние на цену.

Основная литература

1. Уоссермен Ф. Нейрокомпьютерная техника : теория и практика. М. : Мир, 1992. – 240 с.
2. А.П.Пятибратов, Л.П Гудыно, А.А.Кириченко «Вычислительные системы, сети и телекоммуникации», ИНФРА-М, 2008. – 736с.:ил.
3. Нейронные сети Statistica Neural Networks. Методология и технологии современного анализа данных. Под ред. Боровикова В.П., М., Горячая линия – Телеком, 2008.
4. Н. М. Абдикеев «Проектирование интеллектуальных систем в экономике», М., 2003.
5. Методы добычи данных. Internet-ресурс – http://www.statsoft.ru

Лекция 2. Поиск информации.

1. Поисковые средства.

С точки зрения потребителя вся информация в Internet может быть разделена на телекоммуникационные информационные рынки (рис. 1 .).

Рис. 1 Структура телекоммуникационных информационных рынков.

Поиск необходимых сведений в большом объеме достаточно разнообразной информации - задача, которую человечество решает уже многие столетия. По мере роста объема информационных ресурсов были разработаны достаточно совершенные поисковые средства и приемы, позволяющие найти необходимый документ.

В качестве основного инструмента для поиска информации в библиотеках используются каталоги (алфавитные, систематические и предметные). Однако каждый инструмент имеет свои недостатки.

При больших объемах информации (которые характерны для Internet) поиск информации становится очень сложной процедурой. Для того, чтобы найти нужные сведения в Inernet необходимо иметь специальные знания и навыки. Специалист, обладающий такими знаниями и навыками и осуществляющий поиск информации по поступившим заказам называется информационным брокером. Он знает, как устроены классификаторы, как их интерпретируют систематизаторы, какие существуют инструменты для поиска информации в Inernet, технологические приемы и методики поиска, особенности различных поисковых машин и т.д. В беседе с заказчиком он изучает его информационную потребность и превращает ее в поисковое предписание. В нашей стране специалисты такого профиля - пока редкость, хотя потребность в них уже ощущается.

В Internet доступны информационно-поисковые системы (ИПС) трех типов: классификационные, словарные и предметные.

Классификационные ИПС используют иерархическую организацию информации, которая описывается с помощью классификатора. Разделы классификатора называются рубриками. В библиотечном деле для этой цели используется систематический каталог.

Классификатор разрабатывается и совершенствуется коллективом авторов. Затем его использует другой коллектив специалистов, называемых систематизаторами, которые, зная классификатор, читают документы и приписывают им классификационные индексы, указывающие, каким разделам классификатора эти документы соответствуют. В качестве примера классификационной ИПС в Internet можно назвать Yahoo! (www.yahoo.com), в которой одновременно работает более 100 систематизаторов, Excite, Look Smart, Yellow Web, “Созвездие Интернет”, “Ау”.

Классификационные ИПС обладают рядом специфических недостатков. Разработка классификатора связана с оценкой относительной важности различных областей человеческой деятельности. Любая оценка является социальным действием - она связана с обществом, культурой, социальной группой, к которой принадлежит производящий оценку человек. Поэтому классификаторы, созданные разными коллективами в разных странах сильно различаются. Кроме того, у систематизаторов возникают сложности с интерпретацией материалов, написанных на иностранных языках (не только исходных документов, но и классификаторов). Поскольку абсолютно строгой классификации не удается сделать никому, всегда существуют документы, которые можно отнести к нескольким разделам классификатора.

Систематизаторы в сложных случаях (когда неясно, к какому из разделов должен быть отнесен документ) применяют два приема: отсылка и ссылка. Отсылка (в Yahoo! она обозначается знаком @) помещается в тех разделах классификатора, в которые не попал данный документ - в ней указывается, к какой рубрике он отнесен систематизатором. Ссылка используется в тех случаях, когда аналогичная информация может находиться в других разделах классификатора.

Словарные ИПС используют базу данных, построенную из слов, встречающихся в документах Internet`a. В такой базе при каждом слове хранится список документов, из

которых оно взято. Поскольку все морфологические единицы в словаре упорядочены, поиск нужного слова может выполняться достаточно быстро, без последовательного просмотра.

По одному слову найти требуемую информацию довольно сложно. Поэтому, каждая словарная ИПС имеет свой язык запросов, позволяющий комбинировать слова, наиболее полно характеризующие искомую информацию.

К словарным ИПС Internet`а относятся такие, как Alta Vista, Rambler, Яndex, Апорт.

Словарные ИПС способны выдавать списки документов, содержащие миллионы ссылок. Даже простой просмотр таких списков затруднителен. Поэтому многие словарные ИПС предоставляют возможность ранжирования результатов поиска - наиболее важные документы помещаются в начало списка. В языке запросов таких ИПС предусмотрены специальные средства, например, в режиме сложного поиска в Alta Vista можно указать перечень терминов, которые повышают ранг найденного документа (что для этой ИПС особенно актуально, так как она показывает только первые 200 найденных документов). Rambler и Яndex позволяют указать вес каждого из терминов, что позволяет довольно точно настраивать порядок следования найденных документов.

В предметных ИПС с поисковым образом связаны списки ресурсов Сети, содержащих нужную информацию и ссылки на близкие по тематике сайты. В таких ИПС создаются кольцевые ссылочные структуры. Так, сервер www.webring.org содержит несколько десятков тысяч тематических колец (средний размер кольца - около 12 серверов, но есть и кольца-гиганты, в состав которых входят тысячи серверов). Пока кольца были небольшими, поиск информации трудностей не представлял. Для облегчения поиска на указанном сервере используются свои классификационная и словарная ИПС, помогающие найти необходимую информацию.

Аналогичную структуру имеет JumpStation Front Page .

JumpStation является способом нахождения ссылок на информацию, доступную на WWW. Пользователи получают множество связей на другие страницы Web, соответствующих их запросу. Для сбора данных JumpStation использует Robot, обеспечивающий средства поиска для темы, на которую есть ссылка в названии документа.

2. Виды информации.

С помощью информационно-поисковых систем можно искать вполне определенные информационные объекты, список которых приведен на рис. 2 .

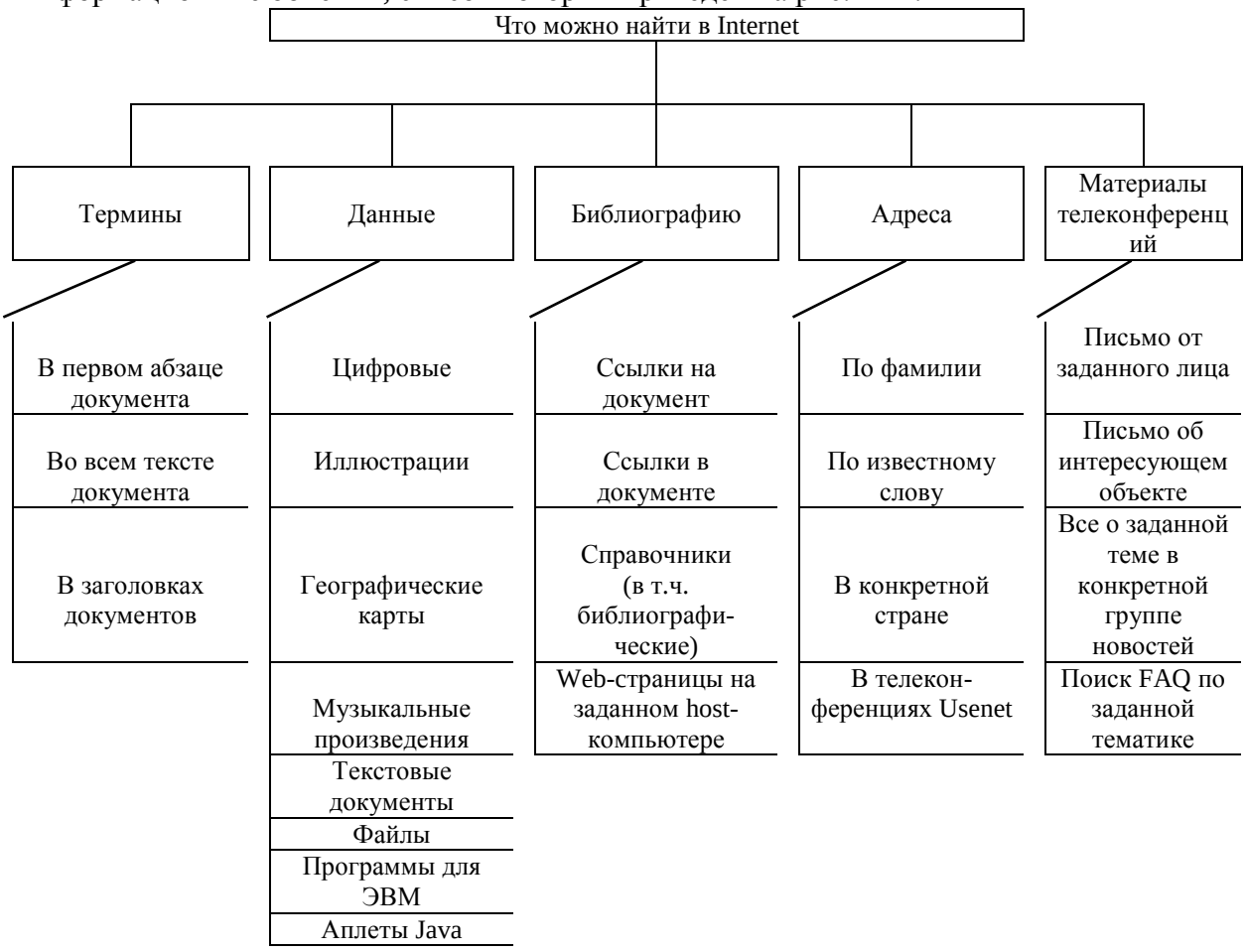


Рис. 2 . Поисковые объекты в Internet.

3. Поисковые серверы

Altavista
www.altavista.com
Lycos
www.lycos.com
Excite
www.excite.com
HotBot
www.hotbot.com
WebCrawler
www.webcrawler.com

InfoSeek
www.infoseek.com
Yahoo
www.yahoo.com
Opentext
www.opentext.com

Каталоги Интернет

Мета-индекс ресурсов Интернет
www.ncsa.uiuc.edu/SDG/Software/Mosaic/MetaIndex.html
Whole Internet Catalog
nearnet.gnn.com/wic/newrescat.toc.html
EINet Galaxy
galaxy.einet.net/galaxy.html#TOP
Поиск информации **по темам**
www.w3.org/hypertext/DataSources/bySubject/Overview.html
Представление о **доступной** информации
www.w3.org/hypertext/DataSources/byAccess.html
Список **сервисов** Internet
www.uwm.edu/Mirror/inet.services.html

Избранные страницы Интернет

Whole Internet Catalog **Top 50**
nearnet.gnn.com/gnn/wic/top.toc.html
Новинки и наиболее популярные страницы Internet
kzsu.stanford.edu/uwi/reviews.html
Самые бесполезные страницы Internet
www.primus.com/staff/paulp/useless.html
StackRambler
Мощная и быстрая поисковая машина компании Stack Ltd. (г. Пущино).
Адрес службы: *http://www.rambler.ru/*
Yandex
Поисковая машина компании CompTek.
Обеспечивает полнотекстовый поиск с полным учетом морфологии русского языка.
Адрес службы: *http://yandex.ru*
Русская поисковая машина
Охарактеризована авторами как первая русская поисковая машина, созданная на базе интегрированного набора поисковых инструментов - Harvest (разработан в 1996 г. группой американских исследователей).
Адрес службы: *http://search.interrussia.com/*
Russian Internet Search
В процессе создания. Также использует Harvest.
Адрес службы: *http://www.search.ru/*
Коммерческие конференции Relcom
Поиск по всем телеконференциям Relcom.
Адрес службы: *http://www.dux.ru/news.html*

Каталоги Интернет

Russia on the Net

Крупнейший индекс ресурсов Интернет в России
Адрес службы: *http://www.ru*

Локальные поисковые машины и Интерфейсы к поисковым серверам

Перенаправляет запрос к нескольким поисковым серверам (AltaVista, Excite, HotBot, Rambler, EuroSeek, WebCrawler)
Умеет переводить запрос с английского на русский и наоборот, а так же может включать в запрос слова во всех словоформах.
Адрес службы: *http://www.medialingua.ru/www/wwwsearc.htm*

Русскоязычная версия AltaVista

Интерфейс предоставлен только в KOI-8 (поиск же, конечно, можно осуществлять во всех кодировках).
Адрес службы: *http://www.altavista.telia.com/cgi-bin/telia?country=ru&lang=ru*

Поиск в AltaVista с учетом морфологии русского языка

Адрес службы: *http://www.cti.ru/alta.html*

Списки серверов Интернет

The List of Russian Web Servers

Имеется возможность поиска
Адрес: *www.mark-itt.ru/jwz/WWW_list/*

List of Russian WWW-servers

Имеется также Sensitive Map of Russia
Адрес: *www.ras.ru/map_list.html*

Обзоры российского Интернет

Вечерний Интернет

Ежедневный обзор Интернета
Доступные кодировки: Koi8-r, Mac, CP1251 (Win), CP866 (Dos), Translit
Адрес: *www.cityline.ru/vi/current.htm*

Новости российского Интернет от Ивана Паравозова

Доступные кодировки: Koi8-r, Mac, CP1251 (Win), CP866 (Dos), Translit
Адрес: *www.cityline.ru/paravozov-news/*

Наблюдения Кати Деткиной

Доступные кодировки: Koi8-R, Win, Translit
Адрес: *www.kulichki.com/kadet/*

4. Поисковая машина Alta Vista.

У каждой поисковой системы существует свой язык запросов, который определяет правила, в соответствии с которыми формулируются запросы на поиск информации.

В классификационных и словарных ИПС запрос составляется на основе ключевых слов, которые являются наиболее яркой характеристикой искомой информации (по сути, без этих слов данная информация обойтись не может). Лучше, если эти ключевые слова имеют

специфический смысл, присущий только искомому информационному материалу, отличающему данный материал от всех остальных.

Ключевые слова могут набираться на разных регистрах клавиатуры - в зависимости от этого поисковая машина будет по-разному проводить поиск.

Поисковая система AltaVista является одной из самых информационно насыщенных. Обратиться к ней можно по адресу:

http://www.altavista.com/

http://altavista.telia.com/tgi-bin/telia?country=ru&lang=ru

(этот адрес позволял обратиться к поисковой системе, работающей на русском языке);

http://home.microsoft.com/intl/ru/access/allinome.asp

(по этому адресу содержится доступ к нескольким поисковым машинам, в том числе - работающим на русском языке).

Рассмотрим правила составления поисковых запросов, использования операторов и команд в языке запросов системы AltaVista:

1) Запрос на поиск информации (поисковое предписание) представляет собой поисковый образ.

2) Поисковый образ может состоять из одного или нескольких ключевых слов.

3) В зависимости от способа соединения ключевых слов в поисковом запросе различают простые и сложные запросы.

4) Сложный запрос отличается от простого тем, что в нем можно указать дату создания искомого документа (чтобы выделить материалы, имеющие последнее обновление после указанной даты), специальную логику поиска (определяемую использованием операторов AND, OR, NOT, NEAR), выбрать один из трех вариантов упорядочивания результатов поиска при их выводе: “только в качестве итога”, “компактная форма”, и “стандартная форма” (последняя используется по умолчанию), и использовать круглые скобки для выделения логически самостоятельных частей запроса.

5) Наличие в ключевом слове заглавной буквы заставит поисковую машину при простом поиске искать слова именно с таким написанием, как в запросе. Если же заглавные буквы не использовались, то поисковая машина учитывает любые варианты написания этих слов. Например, если поисковое предписание состоит из одного слова Computer, будут найдены информационные материалы, содержащие это слово именно в таком начертании. Если же это слово не будет содержать заглавных букв, то при поиске будут учитываться слова в таких начертаниях, как computer, COMPUTER, COMPuter, и др. Необходимо учитывать, что при использовании поискового образа, состоящего только из одного слова computer, AltaVista предоставляет около 2000 ссылок. Просмотреть такое количество ссылок практически невозможно, а значит, информационный поиск нельзя считать эффективным (при правильно составленном запросе необходимая информация находится в числе первых двух десятков ссылок).

6) В том случае, если неизвестно правильное написание слова, или интерес представляет множество однокоренных слов, используется оператор неопределенности - “*” (звездочка). Поставив этот символ после любой последовательности букв (не менее трех), влияние которых необходимо учесть при поиске, можно произвести широкий поиск, при котором ключевое слово будет модифицироваться: поиск будет весьтись как для жестко указанной до звездочки совокупности букв, так и для слов, содержащих любые буквы (числом до 5) вместо звездочки. Например, если указать ключевое слово comp* , то при поиске будут учитываться , как ключевые - computer, computers, compute, и др.

7) Для соединения нескольких ключевых слов могут использоваться операторы “пробел”, “кавычки”, логические операторы “+”, “-”, AND, OR, NOT, NEAR.

8) Оператор “пробел” соединяет слова в поисковом предписании таким образом, что для поиска каждое из этих слов используется отдельно. При этом, порядок слов в запросе не имеет значения. В процессе поиска учитывается только расстояние каждого слова от начала документа и частота его использования в документе.

9) Оператор “кавычки” соединяет слова так, что они образуют фразу, в которой все указанные в предписании слова в документе стоят рядом друг с другом и в той же последовательности, как это указано в предписании. Поэтому, если задать поисковое предписание в виде слов “personal computer” и в виде “computer personal”, то результаты поиска будут разными.

10) Оператор “+”, соединяющий слова, сообщает поисковой машине, что в документе необходимо искать основное слово (первое), но документ надо показывать в результате поиска только если далее в тексте встречаются остальные слова из поискового предписания. Оператор ставится непосредственно перед каждым второстепенным словом. Например, по поисковому образу:

computer +personal +digital

будет вестись поиск основного слова computer, но текст будет считаться актуальным только если в нем встречаются так же слова personal и digital.

11) Оператор “-”, стоящий перед словом, обозначает, что основное слово должно использоваться в тексте без второстепенного. Например, поисковое предписание computer -personal сообщает поисковой машине, что надо искать основное слово computer, но в тексте не должно встречаться слово personal (т.е. интересуют материалы о компьютерах, но не персональных).

12) Операторы AND, OR, NOT, NEAR используются в сложных запросах.

13) Оператор AND (вместо него можно использовать символ &) определяет, что соединяемые им слова должны встречаться вместе (т.е. в простых запросах он эквивалентен знаку “+”).

14) Оператор OR (вместо него можно использовать знак “|”) определяет, что соединяемые им слова независимы друг от друга (в простых запросах он эквивалентен пробелу).

15) Оператор NOT обозначает отрицание (в простых запросах он эквивалентен знаку “-”).

16) Оператор NEAR (вместо него можно использовать символ “~”) определяет, что в искомом тексте указанное им ключевое слово отстоит от основного не далее, чем на 10 слов (например, в поисковом предписании:

провайдер* NEAR “очень дешево”

предусматривается, что в искомом тексте слово “провайдер” и словосочетание “очень дешево” находятся не в разных концах текста, а рядом друг с другом - между ними может находиться не более 10 слов).

Для ограничения поиска в AltaVista используются специальные команды (тэги): anchor, applet, title, url, host, link, image,from, subject.

1) Команда anchor позволяет найти в Сети слово, содержащееся в “теле” ссылки. Для этого после команды anchor через двоеточие указывается искомое слово. Например, поисковый образ содержит:

anchor:home

По этому запросу будет найдено все множество страниц, содержащих внутри ссылок слово home, в том числе - и в такой ссылке: “If you would like go home, press here”.

2) Команда applet позволяет найти заданный названием модуль Java. Например, если модуль Java называется word, то найти его можно, записав поисковый образ: applet:word .

3) Команда title используется в том случае, если искомое слово находится в заголовке текста. Например, по запросу вида:

title:links

будут найдены документы, содержащие слово links в заглавии, в том числе текст с заглавием “Cool Links”.

4) Команда url предписывает искать url-адрес, содержащий заданное слово. Например, если неизвестно, в каком корневом домене находится host-компьютер МЭСИ, можно задать поисковое предписание: url:mesi . Среди множества адресов с таким словом будет и адрес http://www.mesi.ru/ .

5) Команда host позволяет узнать, какие Web-сайты есть на заданном host-компьютере. Например, для того, чтобы узнать, какие сайты есть на хосте www.intel.ru необходимо набрать запрос: host:intel.ru . Если же в запросе указать только часть имени, то в результате поиска будут найдены сайты, имеющие другие адреса, но содержащие заданную часть имени.

Используя эту команду, можно вести поиск в заданной стране. Например, по запросу host:*.ru +kreml будет найдена информация о Московском, Рязанском и других Кремлях. При этом нужно помнить, что поиск ведется только для сайтов, зарегистрированных в поисковой системе AltaVista, другие сайты ей недоступны.

6) Команда link позволяет найти адреса страниц (сайтов), содержащих ссылку на конкретную (заданную в поисковом образе) Web-страницу. Например, для того, чтобы узнать, кто ссылается на сайт www.mesi.ru необходимо задать предписание: link:www.mesi.ru . Результатом будет список страниц, на которых содержатся ссылки на сайт mesi.ru .

7) Команда image позволяет найти иллюстрацию в Internet. Для этого надо знать название файла, в котором она хранится. Формат команды тот же.

8) Команда from позволяет искать в телеконференциях Usenet почтовое сообщение, отправленное конкретным человеком, имя которого указывается после двоеточия в команде. Например: from:Иван +Федоров (или Ivan +Fedorov).

9) Команда subject позволяет искать сообщения в телеконференциях Usenet на конкретную, заданную в поисковом предписании тему.

Поисковая система AltaVista может работать (и вести поиск) на разных языках, в том числе и на русском.

Описанные принципы управления поисковой системой во многом аналогичны используемым и в других поисковых системах.

1. Задание.

Ознакомьтесь с описанием поисковой системы AltaVista.

Обратите внимание на:

- В каких частях документа можно вести поиск и как оформляется для этого поисковое предписание;
- Какие виды информации можно искать с помощью этой поисковой системы;
- Как она настраивается на работу на русском языке
- Решите задачу: определите, кто является партнёром заданной Вам фирмы.
- Возможная последовательность решения задачи:
- Найдите сайты этой фирмы (их может быть несколько);
- С помощью AltaVista получите список – кто ссылается на сайт фирмы;
- Проведите анализ ссылок (в связи с чем эти ссылки сделаны). Выберите тех, кто является партнёром фирмы (поставщики, потребители, покупатели, соисполнители,…);

- Оформите результаты поиска с описанием технологии поиска. Сделайте чёткие выводы, проведите классификацию партнёров и охарактеризуйте каждого из них. Оцените, в какой степени характеризуют деятельность фирмы её сайты.
- Сформулируйте последовательность решения аналогичной задачи – поиска конкурентов фирмы.
- Определите, какие сайты зарегистрированы в данной стране. Страну выберите по указанию преподавателя (из списка ISO 3166).
- Аналитический отчёт представьте преподавателю.

5. Поиск информации с использованием пакета Google.

Internet и, в частности, WWW (как наиболее удобный способ навигации) предлагают доступ к огромному массиву информации - и крайне необходимой, и абсолютно бесполезной. Логично предположить, что с возрастанием количества доступной информации способы ее обнаружения будут усложняться. Но на самом деле все намного проще. В WWW существуют десятки служб, облегчающих поиск необходимой информации.

Правда, большинство пользователей использует не более 10% имеющейся в Internet информации. Во-первых, основные возможности поисковых машин направлены на текстовую часть документов, а информация может находиться в рисунках, в заголовках, в подрисуночных надписях, и др. Во-вторых, информационные ресурсы не всегда лежат на поверхности, есть и глубинные ресурсы – они как бы «закопаны» в другой информации, хотя специально доступ к ним никто не ограничивал.

В рамках Академии Информационных Систем президентом консорциума «ИНФОРУС» Андреем Масаловичем создано направление конкурентной разведки в Интернете, которое рассматривается как одно из самых актуальных на сегодняшний день направлений дополнительной профессиональной подготовки вследствие стремительного развития глобальной Сети, резкого увеличения количества информационных ресурсов и усиления влияния Интернет на деятельность предприятий и организаций.

В рамках данного факультатива придётся выполнять специальную работу – обеспечивать своё исследование информацией. Поэтому начинаем работу в факультативе со знакомства с поисковыми системами и методами поиска информации в Internet.

Для поиска ресурсов Сети разработано огромное количество поисковых систем. Наиболее известными из них являются Google, Yandex, Rambler, Aport. Менее известной является AltaVista, хотя ранее это была одна из самых мощных поисковых систем. Стандартной для всех этих систем является базовая поисковая техника. Но у всех них имеются нестандартные возможности, основанные на внутренних командах поиска.

Используя поисковые машины, такие как Google, можно решить многие исследовательские задачи. Нестандартные возможности поисковой машины Google подробно рассмотрены в «**Google Hacks, 2nd Edition,** By Tara Calishain, Rael Dornfest, **Publisher:** O'Reilly, 2004, **ISBN:** 0-596-00857-0, **Pages**: 479».

Базовая поисковая техника Google.

Для поиска требуемой информации составляется поисковое предписание, содержащее характеризующие эту информацию ключевые слова.

Дополнительные правила составления поискового запроса:

- Google не воспринимает более 10 слов для поиска, включая специальный синтаксис.
- Знак (+) в поисковом предписании используется для поиска наиболее общих слов (пробелы между знаком и помечаемым этим знаком словом не допускаются). Часто используемые слова "I", "a", "the", "of" и т.п. игнорируются поисковиком, но можно заставить его их искать если поставить перед ними "+". Например: "Война +и мир"
- Знак (-) в поисковом предписании используется для исключения термина из поиска (пробелы между знаком и помечаемым этим знаком словом не допускаются)
- Для поиска фразы надо поместить её в кавычки «»
- Знак «*» означает любое слово. Google не поддерживает поиска по корням слов (stemming), то есть возможности использования звёздочки (или другого знака маски) вместо букв в искомом слове. Например, moon* в поисковике, поддерживающем маски, найдёт "moonlight," "moonshot," "moonshadow," и т.д. Google же использует звёздочку как заменитель целого слова. Поиск по фразе "three * mice" в Google даст в результате "three blind mice," "three blue mice," "three red mice," и т.д.
- Обойти лимит в 10 слов можно используя звёздочки. Каждая звёздочка заменяет одно слово. Как оказалось, Google просто не считает количество звёздочек в запросе.
  "do as * say not as * do" quote origin English usage - замена "do as I say not as I do" quote origin English usage

Нестандартные возможности поисковой машины Google.

Так как Google является полнотекстовым поисковиком, он индексирует всё содержимое страниц. Дополнительные команды, называемые спец. синтаксисом (операторы для продвинутого поиска) позволяют пользователям Google искать конкретные части web страниц или тип информации. Это позволяет сузить число результатов поиска.

Такие операторы имеют следующий синтаксис: «operator:search_term» (в этом выражении также не должно быть никаких пробелов).

Примеры этих операторов:

- **site:** инструктирует Google ограничить поиск конкретным web-сайтом (доменом); название сайта (домена) указывается сразу после двоеточия и без пробела. "site:" сужает поиск до одного сайта или домена верхнего уровня. Например:

site:loc.gov
site:thomas.loc.gov
site:edu
site:nc.us

- **filetype:** инструкция произвести поиск только в пределах текста указываемого типа файлов. Тип файла указывается после двоеточия (точку перед расширением файла указывать не нужно). "filetype:" ищет среди расширений файлов, а точнее - в файлах с определённым расширением. Например:

homeschooling filetype:pdf
"leading economic indicators" filetype:ppt

- **link:** производить поиск внутри гиперссылок содержащих поисковый запрос. "link:" возвращает список страниц, имеющих ссылку на заданную. Например,

введите "link:www.google.com" и увидите список сайтов имеющих ссылку на Google. Не обязательно вводить "http://"; Google проигнорирует эту часть текста даже если её ввести. "link:" отлично работает как с "глубокими" адресами, вроде "http://www.raelity.org/apps/blosxom/", так и с верхнеуровневыми URL, такими как "raelity.org".

- **cash:** оператор демонстрирует версию страницы, которая существовала, когда она индексировалась Google. URL страницы указывается сразу после двоеточия. "cache:" ищет копию страницы проиндексированной Google даже если страница уже недоступна по оригинальному URL или её содержимое полностью изменилось. Например:

cache:www.yahoo.com

- **intitle:** производить поиск внутри названия документа. "intitle:" ограничивает поиск до заглавий страниц (titles). Вариации, "allintitle:" ищет страницы в заглавии которых находятся все слова поиска. Например:

intitle:"george bush"
allintitle:"money supply" economics

- **inurl:** искать внутри URL документа. "inurl:" ограничивает поиск до адресов (URL) страниц. Команда хороша для поиска страниц помощи и поиска, так как они имеют довольно стабильную структуру. "allinurl:" вариация, которая ищет все введённые слова в URL. Например:

inurl:help
allinurl:search help

- **intext:** ведёт поиск только по тексту страниц (т.е. игнорирует текст ссылок, URL, и заглавий). Есть вариация "allintext:", но она плохо ладит с другими командами. Например:

intext:"yahoo.com"
intext:html

- **inanchor:** ищет текст в якорях ссылок на страницах (anchors). Якори ссылок – это текст описания ссылки. Например, во фрагменте кода HTML [a href="http://www.oreilly.com" O'Reilly and Associates] якорем ссылки является "O'Reilly and Associates." Пример:

inanchor:"tom peters"

- related: находит страницы, похожие на запрашиваемую. Например, поиск "related:google.com" даст множество поисковиков, включая HotBot, Yahoo!, and Northern Light. Аналогично:

related:www.yahoo.com
related:www.cnn.com

- info: предоставляет ссылки на более подробную информацию о запрошенном URL. Информация включает ссылки на кэш URL, список страниц имеющих ссылки на данную, страницы, связанные с данной, страницы, содержащие данный URL. Например:

info:www.oreilly.com
info:www.nytimes.com/technology

Такие операторы можно использовать как по отдельности, так и в различных сочетаниях, в том числе – с различными ключевыми словами. Но некоторые из них отлично работают в сочетании друг с другом, некоторые друг другу мешают, а некоторые просто друг с другом не работают.

Индивидуальные, не сочетающиеся с другими, команды: rphonebook:, bphonebook:, phonebook:, link:. Остальные спец. команды можно смешивать как угодно.

Примеры нестандартного поиска.

Карта сайта

Чтобы выявить каждую страницу на сайте, Google сканирует его, используя оператор "site:" и дополнительные ключевые слова, которые должны содержаться на **каждой** странице сайта.

Например, составим запрос такого вида: **site:http://www.microsoft.com microsoft**. Этот запрос выполняет поиск по слову «microsoft» в пределах сайта http://www.microsoft.com.

Как много страниц на сервере Microsoft содержат слово «Microsoft»? Для выяснения этого вопроса надо иметь в виду, что Google исследует не только содержание страниц, но также их название и URL. Слово «Microsoft» стоит в URL каждой страницы http://www.microsoft.com.

Таким образом - единственным запросом можно инициировать обработку каждой страницы на сайте Microsoft, проиндексированной Google.

Нахождение листинга директории

Листинг директории представляет собой список файлов и директорий удаленного сервера в окне браузера. Такие листинги открывают широкие возможности для углубленного сбора информации. Как правило: такие страницы директорий имеют в Title и теле страницы выражение «Index Of». Отсюда очевидно и строение запроса для поиска таких листинов - это «intitle:index.of». В результате такого запроса будут найдены страницы со словом «index of» в разделе Title документа.

К сожалению – этот запрос вернет слишком большое число страниц не по теме, к примеру, страницы вида:

• Index of Native American Resources on the Internet
• LibDex—Worldwide index of library catalogues
• Iowa State Entomology Index of Internet Resources

Исходя из названий найденных документов, очевидно, что эти страницы не соответствуют заданному запросу и вряд ли окажутся искомыми списками директорий.

Следующие запросы обеспечат более точные результаты:
intitle:index.of "parent directory"
intitle:index.of name size

Такие запросы более точно выдадут то, что нам нужно, поскольку ориентированы не только на фразу «index of» в Title страницы, но и на ключевые слова, всегда имеющиеся в листингах директорий: «parent directory», «name», «size».

Определение версии WEB-сервера

Точная версия программного обеспечения web сервера – это один из элементов, необходимых администратору для точной настройки Web-сайта. Если непосредственно соединиться с сервером, то HTTP (web) заголовки (headers) этого сервера предоставят нужную информацию. Однако можно получить эту информацию **из кэша Google безо всякого соединения с сервером**. Такой метод основан на использовании списка директорий.

Список файлов директории включает имя серверного софта и его версию.

Выглядит такой запрос просто: **«intitle:index.of server.at»**

Он основан на содержании фразы **«index of»** в разделе **title** страницы директории и фразы **«server.at»**, содержащейся в конце любого листинга директории. К примеру, так выглядит запрос, определяющий версию сервера aol.com:

«intitle:index.of server.atsite:aol.com».

Использование Google в качестве сканера CGI директорий

Для выполнения подобной задачи, CGI сканер изначально знает - какие именно директории нужно искать на сервере. Как правило - это директории, в которых располагаются файлы данных и имеют вид, подобный представленным ниже:

/cgi-bin/cgiemail/uargg.txt
/random_banner/index.cgi
/random_banner/index.cgi
/cgi-bin/mailview.cgi
/cgi-bin/maillist.cgi
/cgi-bin/userreg.cgi
/iissamples/ISSamples/SQLQHit.asp
/iissamples/ISSamples/SQLQHit.asp
/SiteServer/admin/findvserver.asp
/scripts/cphost.dll
/cgi-bin/finger.cgi

Зная синтаксис требуемых директорий, а также владея техникой поиска, изложенной выше, можно использовать Google как CGI сканер.

- Например, поиск в Google следующего вида: **allinurl:/random_banner/index.cgi** вернет документы с адресами страниц конкретных программ генерации рекламных баннеров.

Использование Google как внутренней поисковой системы Web-сайта.

При поиске информации на серверах можно использовать не только их "родные" формы поиска, но и Google. Например? Поисковый запрос:
""george bush" site:nytimes.com" - поиск статей про Дж.Буша на сайте Нью Йорк Таймс.

6. Литература.

http://www.sbnet.ru/navigation/search.ru.html - Средства поиска информации в www

Гиперссвлки на Мировой Интернет

- Поисковые серверы
- Каталоги Интернет
- Списки поисковых серверов Интернет
- Интерфейсы к поисковым серверам
- Списки серверов Интернет
- Избранные страницы Интернет

Русский Интернет

- Поисковые серверы
- Каталоги Интернет
- Локальные поисковые машины
- Интерфейсы к поисковым серверам

- Списки серверов Интернет
- Обзоры российского Интернет

Списки серверов Интернет По **странам**

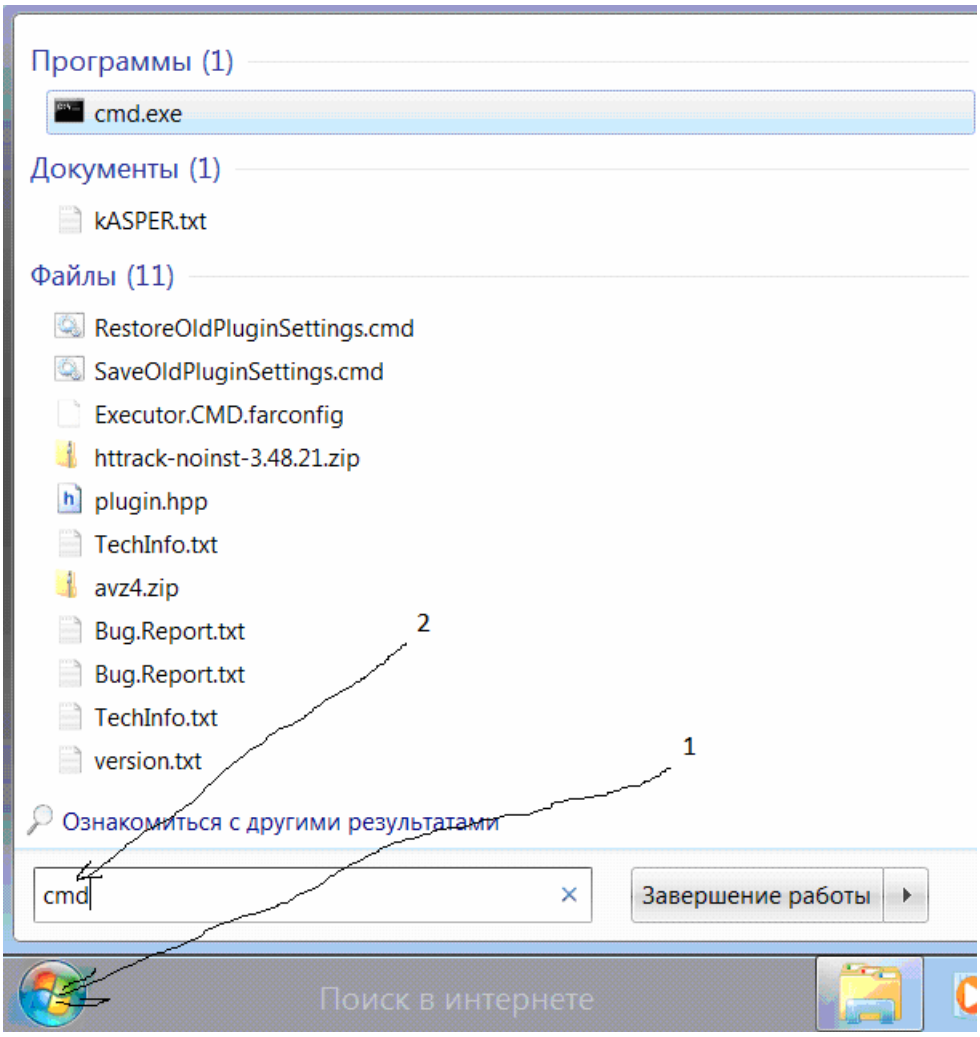
www.w3.org/hypertext/DataSources/WWW/Servers.html

Лекция 3. Препроцессинг и постпроцессинг.

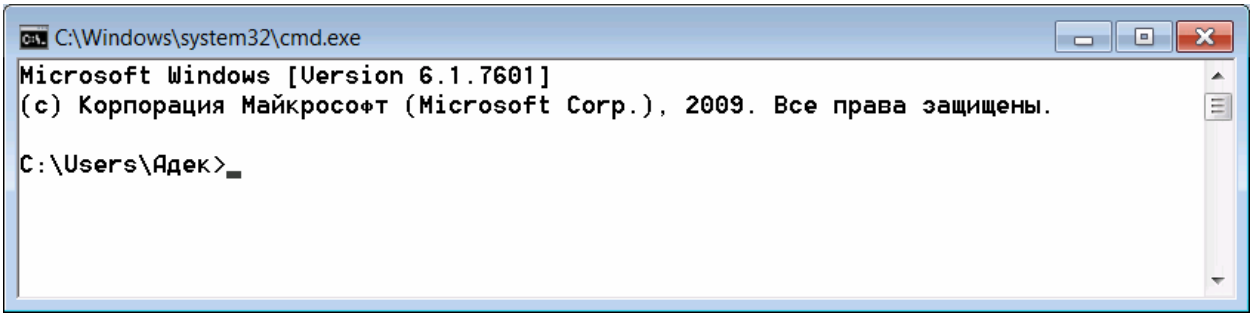
Препроцессинг.

Настройка ОС для подготовки данных к НС-исследованию.

Для настройки операционной системы к подготовке данных для нейросетевого исследования нажмите кнопку Пуск и в открывшемся над ней окошке наберите имя программы cmd:



Появится окно чёрного цвета. Поменяйте в нём цвета символов и фона:



Подготовку данных к нейросетевому исследованию рассмотрим на следующем примере: имеющийся файл class1.xls надо переименовать в class1.txt, а операционная система настроена так, что расширения файлов не видны.
Для работы необходимо все свои данные держать в одной папке, а саму папку (назовём её u) расположить в корне рабочего диска. Папку u можно создать в корне диска C стандартными средствами операционной системы. Сохраните в своей папке файл class1.xls. затем выполняйте в окне CMD.exe команды:
перейти в корень диска C и войти в созданную папку u. После набора выделенных красным цветом символов нажимать Enter:

```
C:\Users\Адек>cd \

C:\>cd u
```

Проверить содержимое текущей папки u:

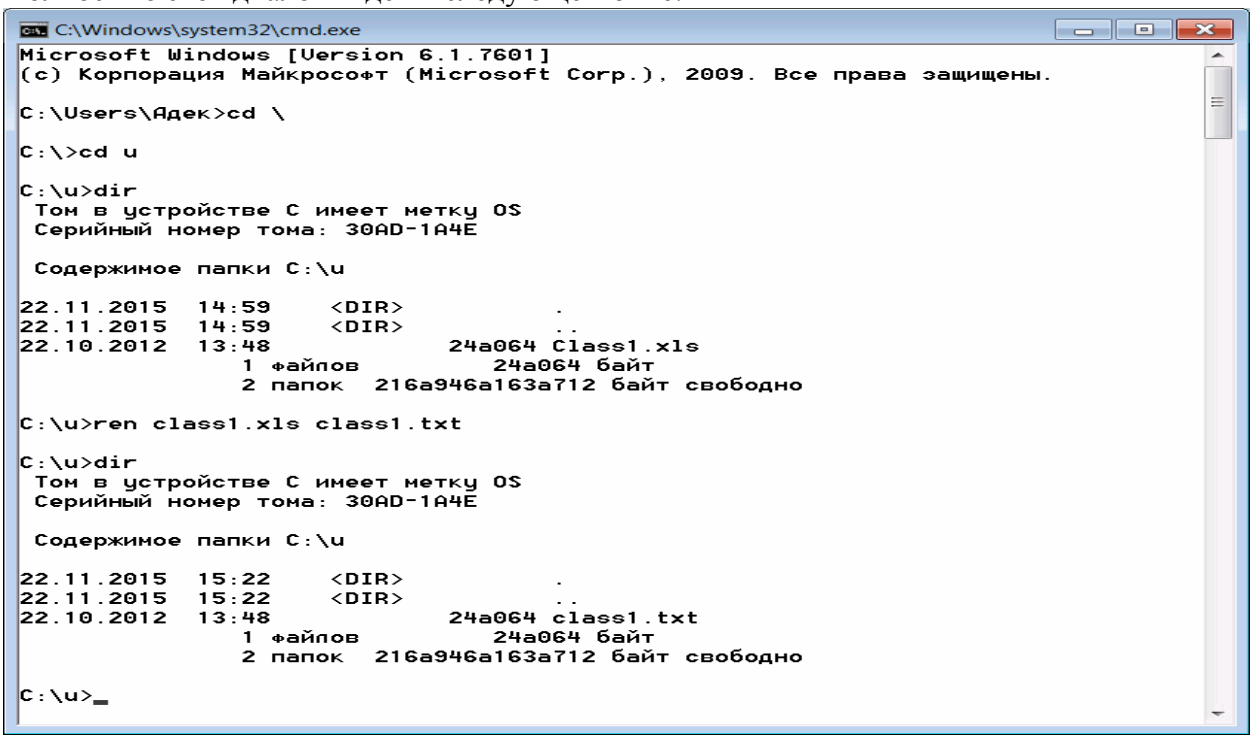
```
C:\u>dir
 Том в устройстве C имеет метку OS
 Серийный номер тома: 30AD-1A4E

 Содержимое папки C:\u

22.11.2015  14:59    <DIR>          .
22.11.2015  14:59    <DIR>          ..

22.10.2012  13:48            24 064 Class1.xls
               1 файлов         24 064 байт
               2 папок  216 946 163 712 байт свободно
```

Теперь расширения файлов видны.

Переименовать class1.xls в class1.txt и проверить изменение:

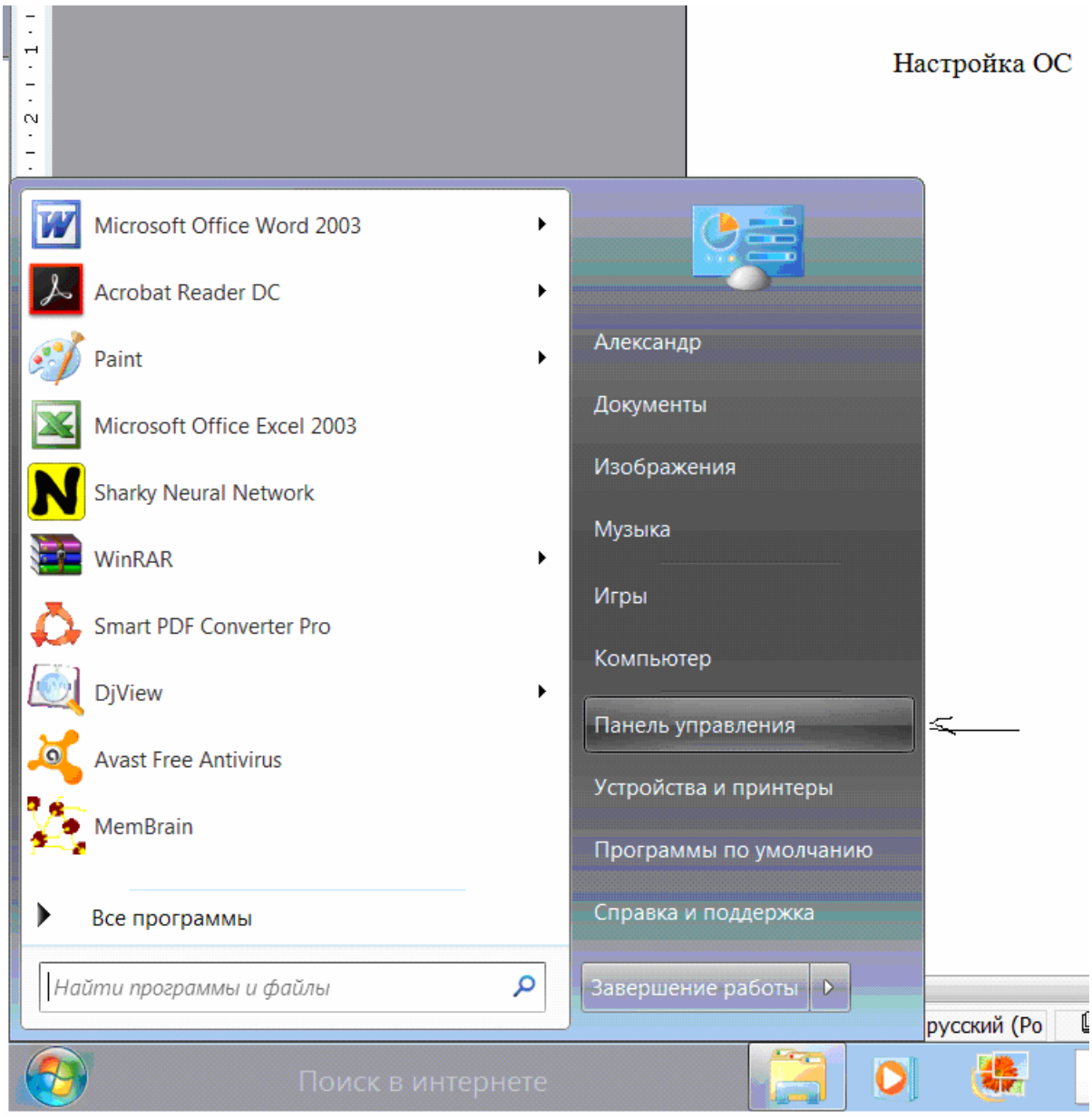
```
C:\u>ren class1.xls class1.txt

C:\u>dir
 Том в устройстве C имеет метку OS
 Серийный номер тома: 30AD-1A4E

 Содержимое папки C:\u

22.11.2015  15:22    <DIR>          .
22.11.2015  15:22    <DIR>          ..
22.10.2012  13:48            24 064 class1.txt
               1 файлов         24 064 байт
               2 папок  216 946 163 712 байт свободно

C:\u>
```

Полностью этот диалог виден в следующем окне:

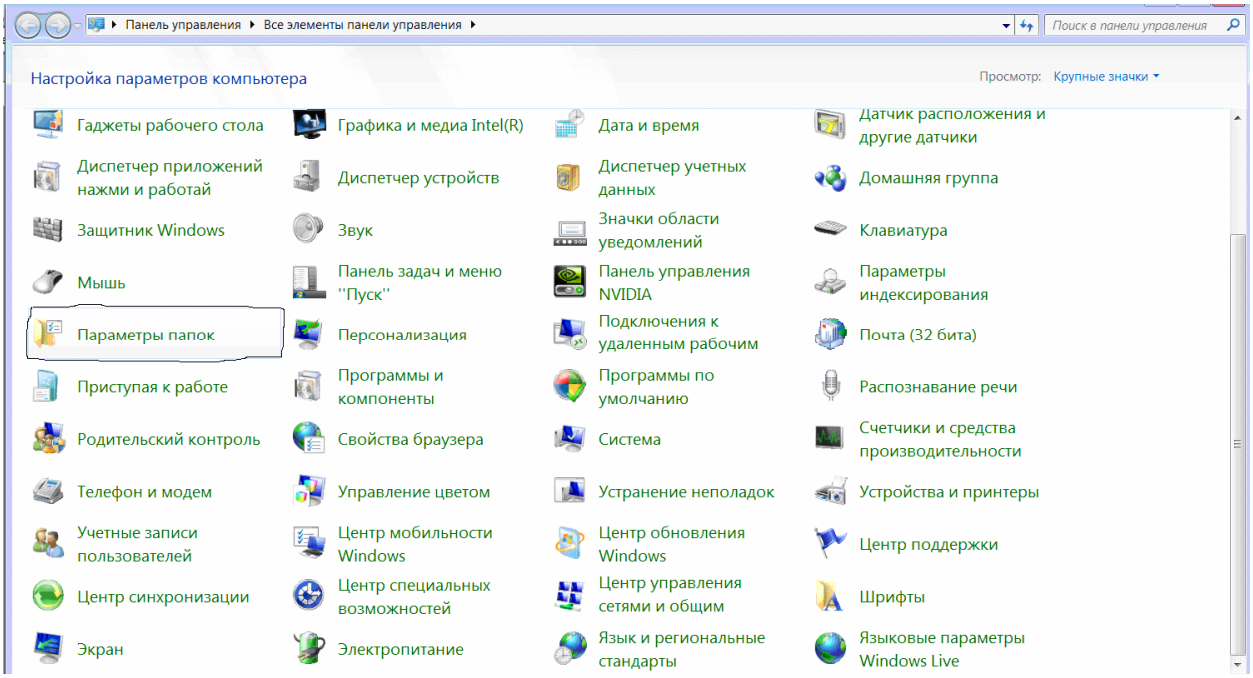


2 способ настройки ОС.

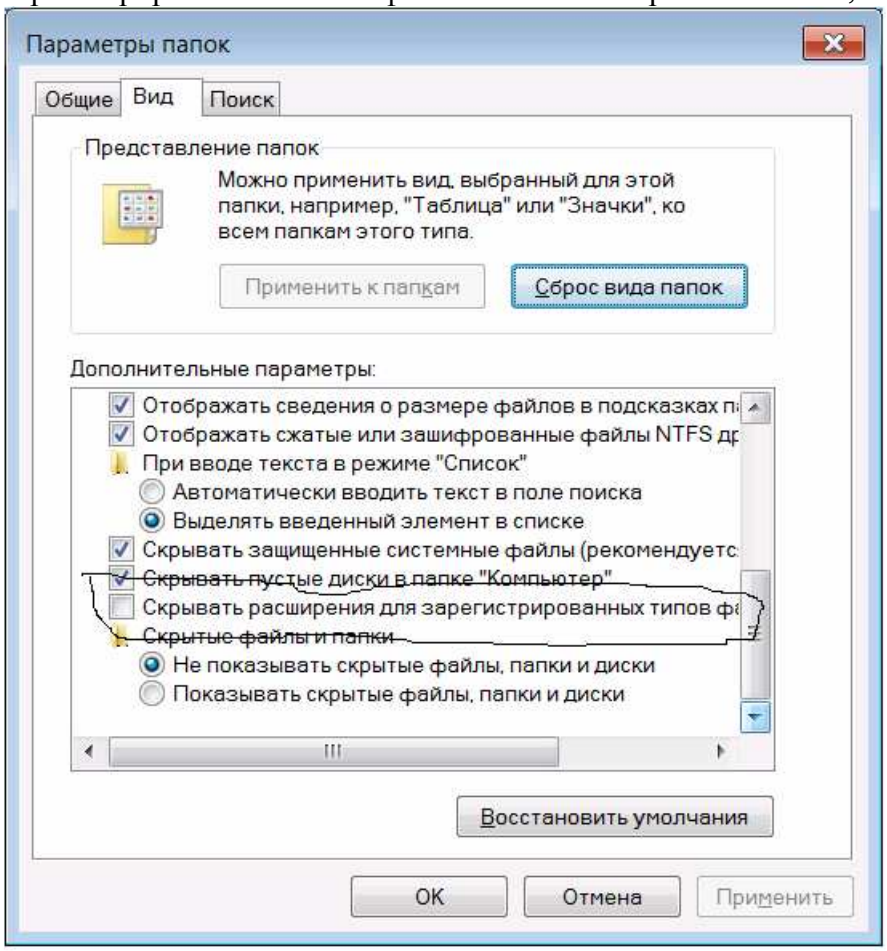
То же самое можно сделать и без обращения к командам операционной системы.
Нажмите кнопку Пуск и вызовите Панель управления:



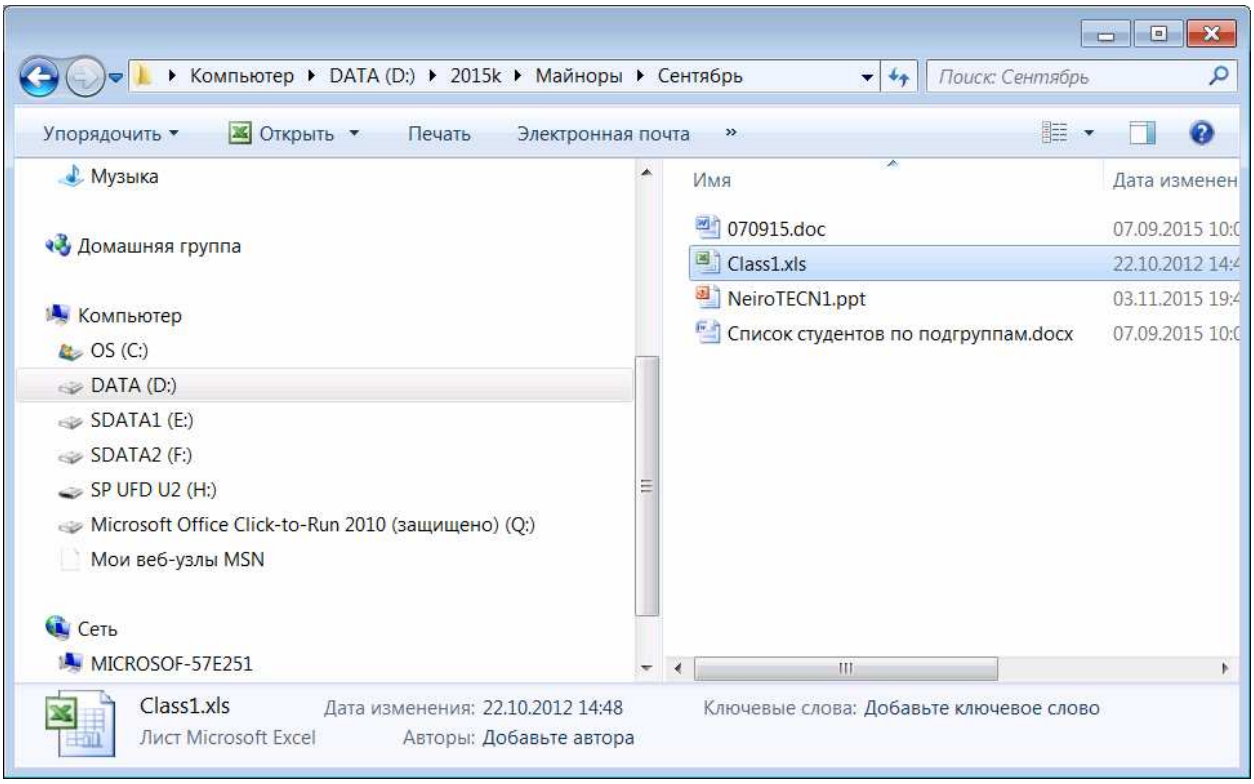
В панели управления выделите Параметры папок:



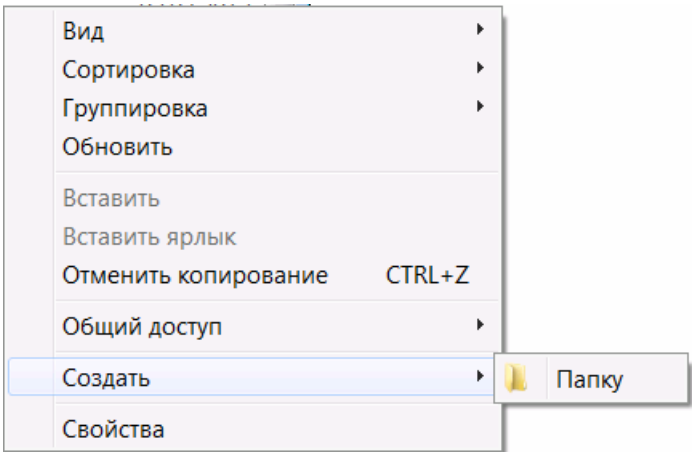
В Параметрах папок перейти на вкладку Вид и отыскать строку Скрывать расширения для зарегистрированных типов файлов. Если эта строка помечена, пометку надо снять:



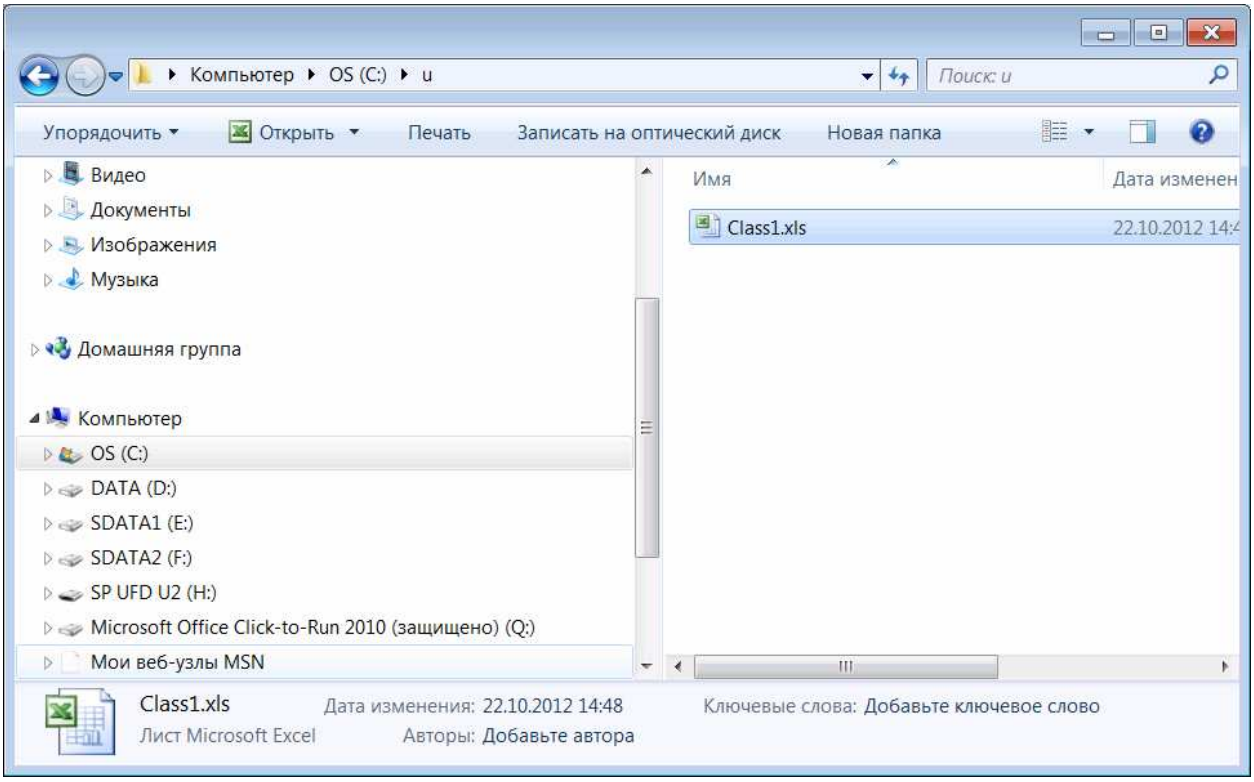
Для проверки надо вызвать на экран названия файлов:



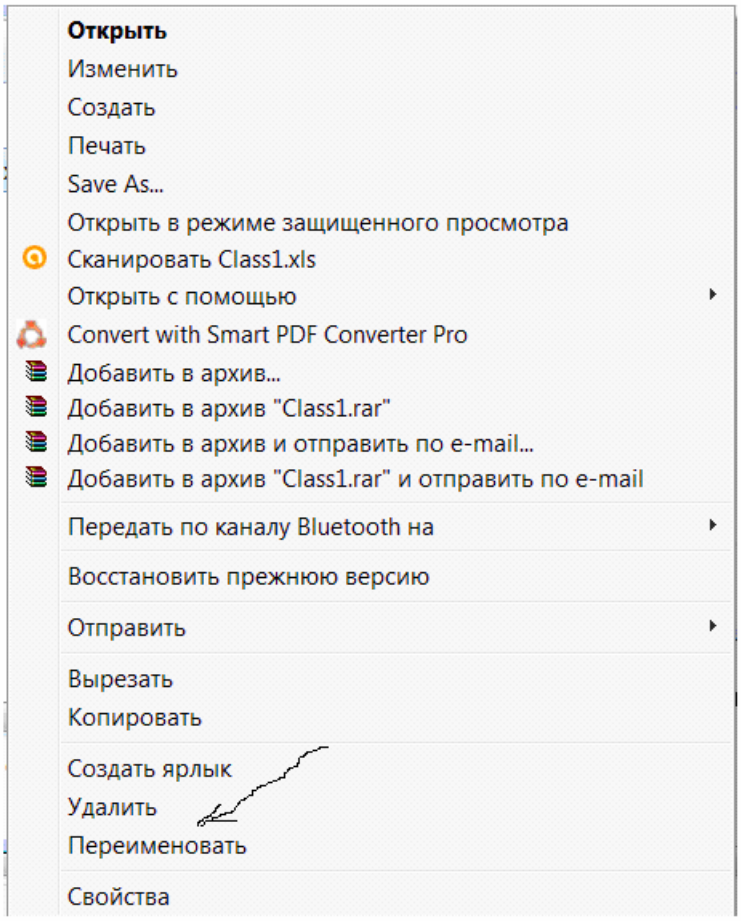
Обычным путём с помощью мышки создаём папку u на диске C:



Переносим в неё файл с расширением xls.

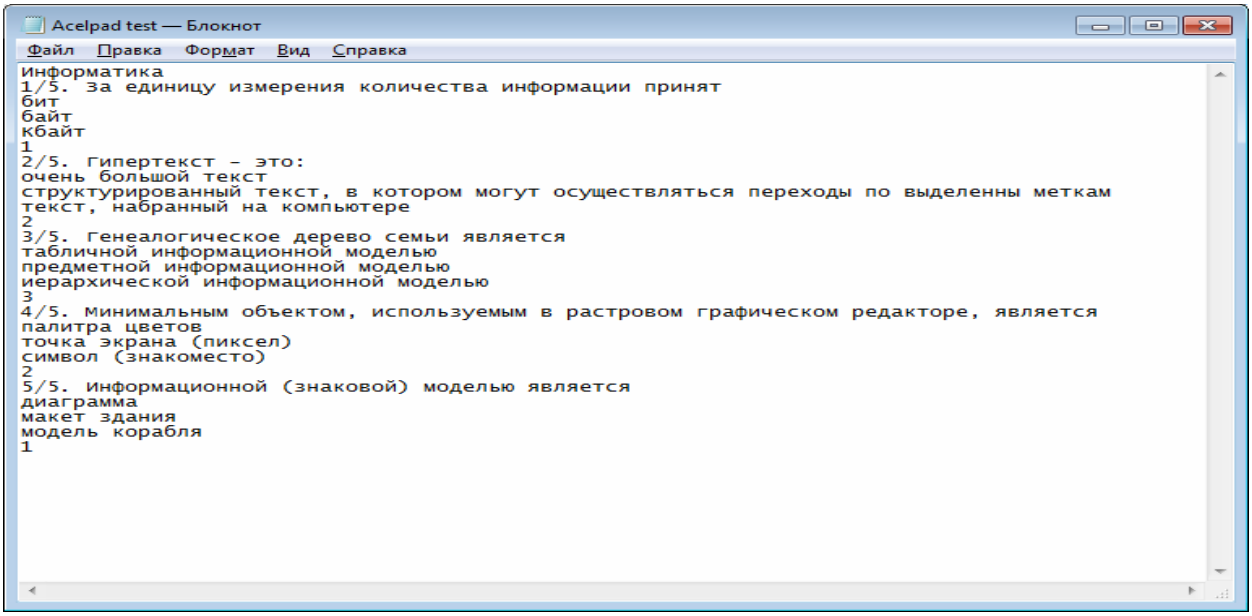


При правильной настройке операционной системы расширение файла видно на экране и может быть изменено правой кнопкой мыши:

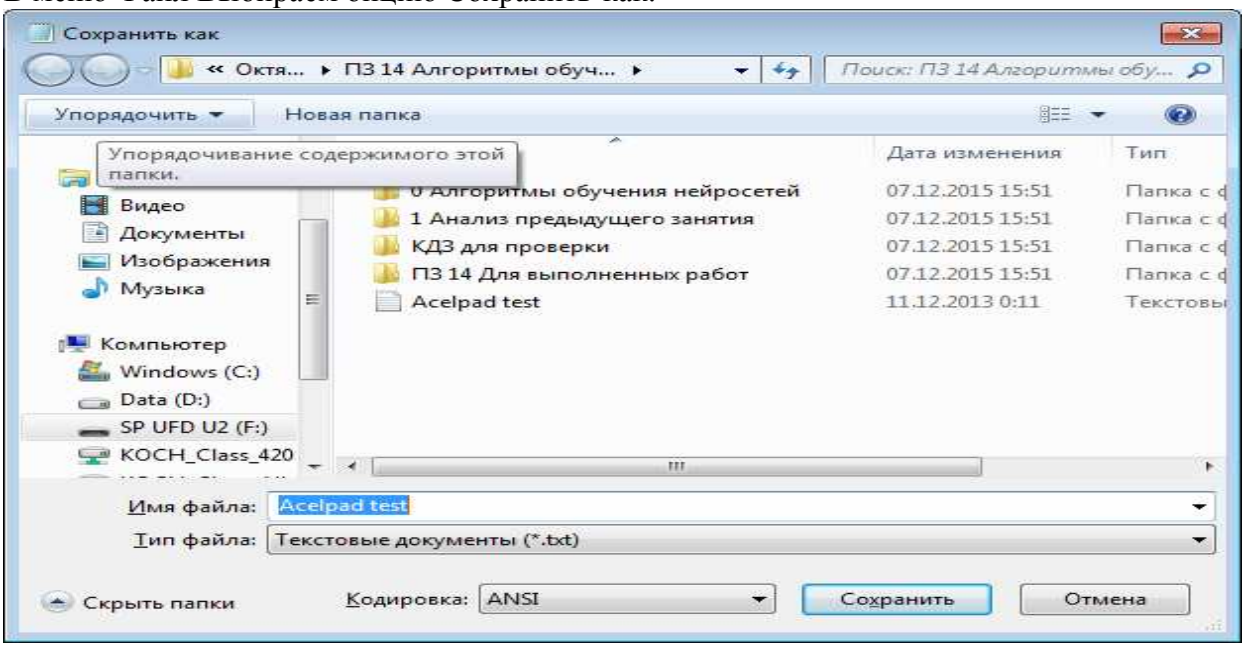


Преобразование кодировки txt файла

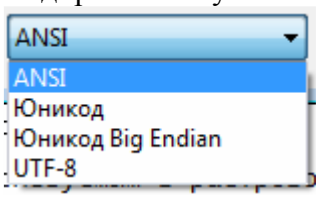
Преобразовать кодировку текстового файла можно используя Excel, различные редакторы текста, например – блокноты.
Открываем в блокноте txt файл:



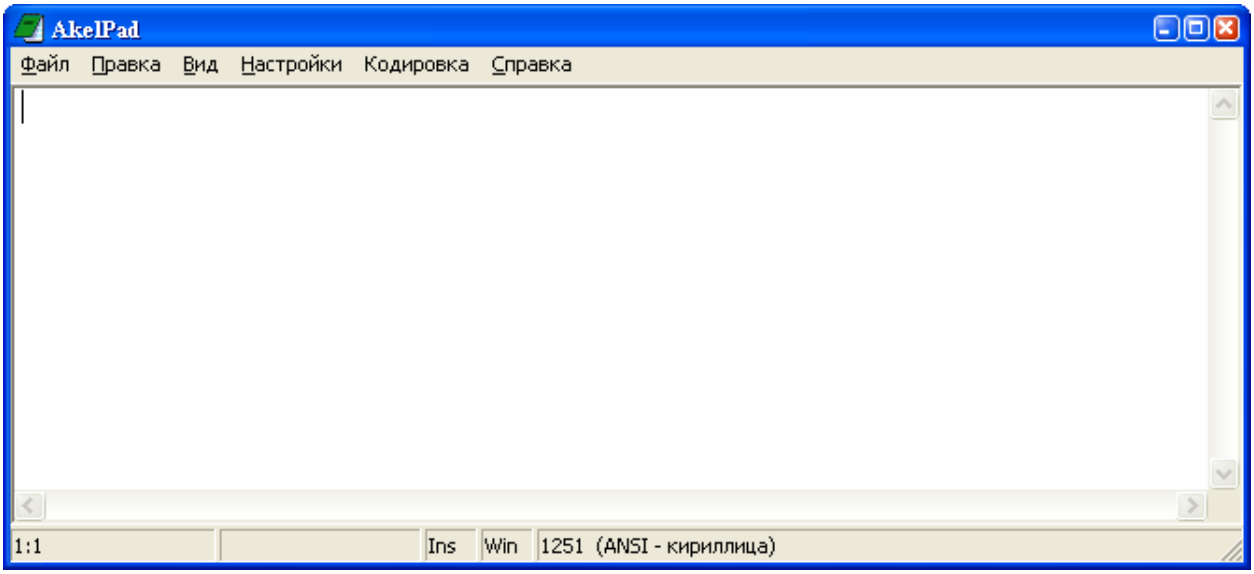
В меню Файл Выбираем опцию Сохранить-как.



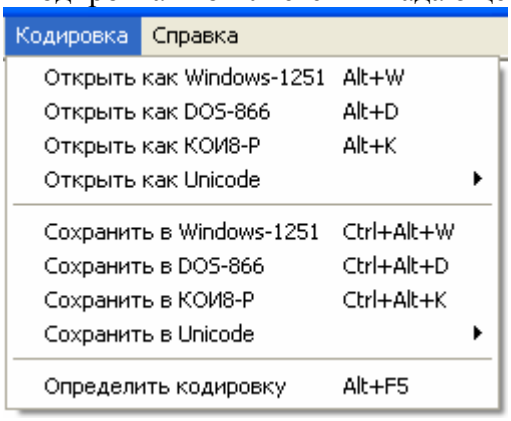
В нижней части окна появляется Кодировка и в ней указано, в каком коде файл сюда записан. Кодировки могут быть такими:



Для Матлаб текстовые файлы должны иметь кодировку ANSI.
В других документах типа “блокнот” может быть явно указана функция “Кодировка”:



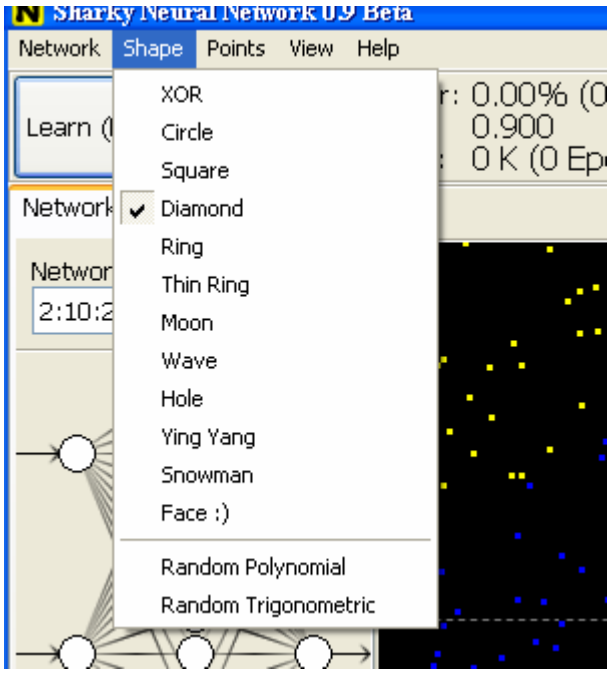
В нижней строке такого окна указан текущий тип кодировки файла. При выборе функции “Кодировка” появляется выпадающее меню:



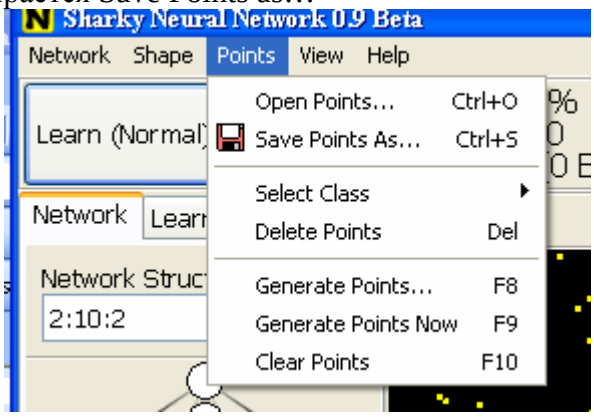
Пример переноса файла из Sharky в Matlab.

Для загрузки файла, полученного из Sharky (Sharky Neural Network – это компьютерная программа фирмы SharkTime Software (http://www.sharktime.com) для игровой демонстрации возможностей нейросетевого классификатора. Программа freeware, работает под ОС Windows 2000, Windows XP, и Windows Vista.

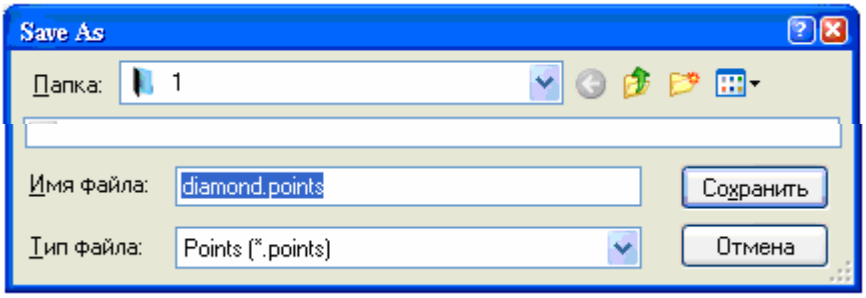
Программа реализует нейронную сеть типа многослойного перцептрона, предназначенную для классификации 2D-точек в два различных класса (жёлтый и синий). Для переноса файла с данными в Матлаб сначала в меню Sharky выделяем необходимый набор, например, Diamond:



Затем в меню Points выбирается Save Points as…



Задаём название папки и имя файла:

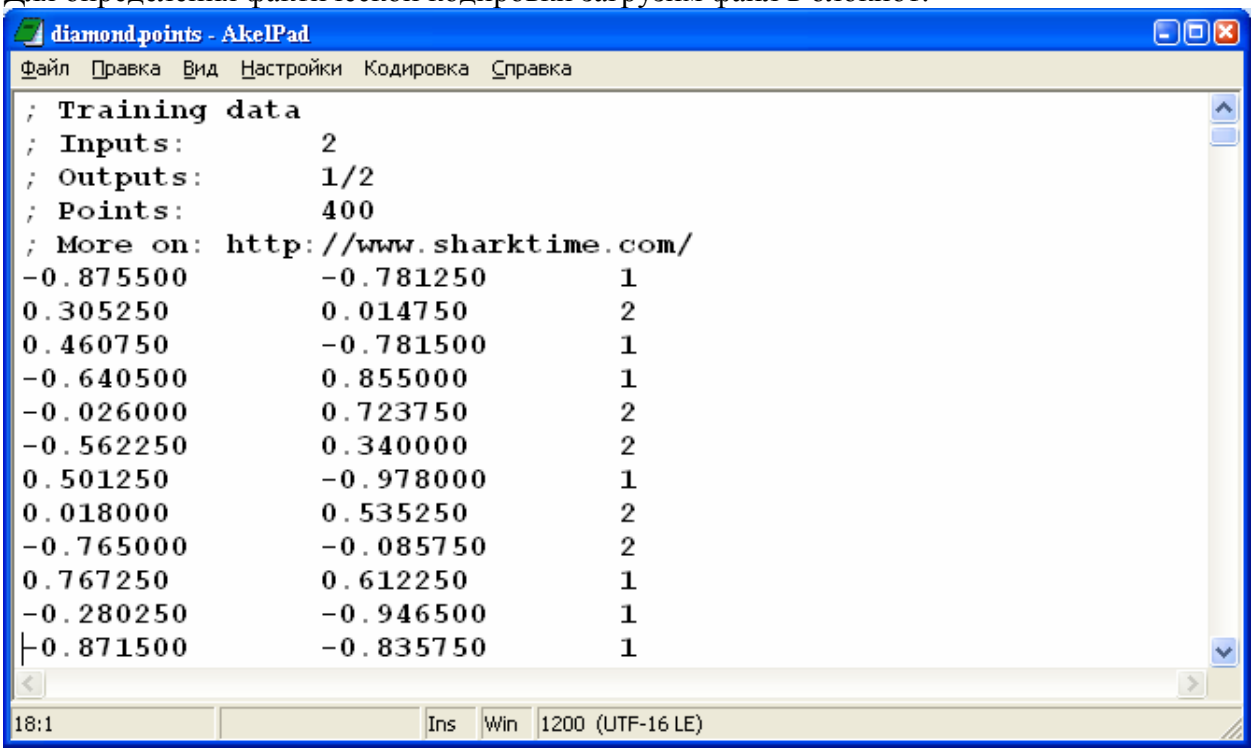


После сохранения получаем:

Этот файл содержит:

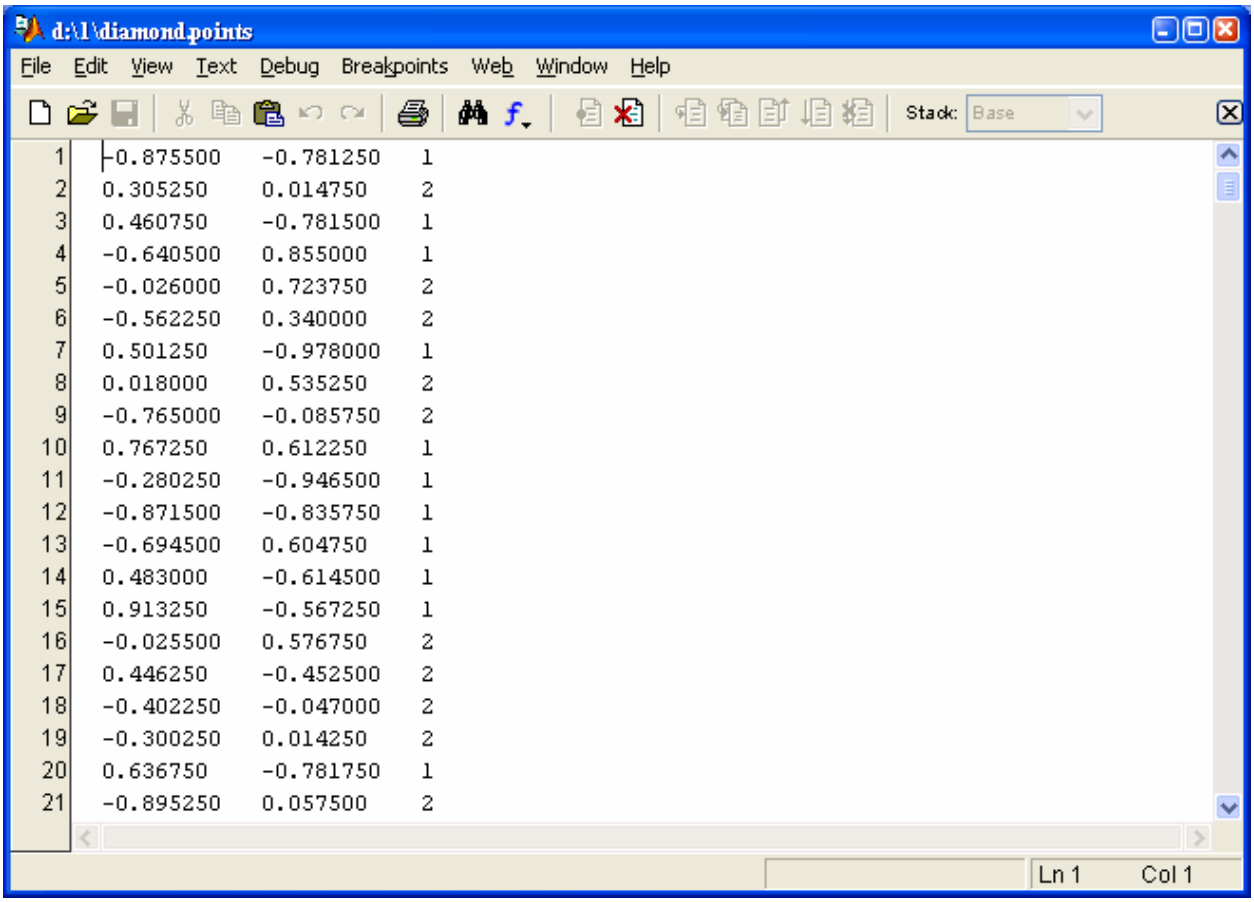
```
; Sharky Neural Network 0.9 Beta
; Training data
; Inputs:   2
; Outputs:  1/2
; Points:   400
; More on: http://www.sharktime.com/
-0.875500   -0.781250   1
0.305250    0.014750    2
0.460750    -0.781500   1
-0.640500   0.855000    1
-0.026000   0.723750    2
-0.562250   0.340000    2
0.501250    -0.978000   1
0.018000    0.535250    2
-0.765000   -0.085750   2
0.767250    0.612250    1
-0.280250   -0.946500   1
```

Необходимо открыть его и загрузить в переменную «a» в рабочем документе Матлаб. Для этого нужно, чтобы кодировка файла соответствовала требованиям Матлаб :
(Ins Win 1251 (ANSI - кириллица). И чтобы файл содержал только однородную информацию, без верхних 5 строк.
Для определения фактической кодировки загрузим файл в блокнот:



В нижней строке читаем текущую кодировку файла: Win 1200 (UTF-16LE).
В меню «Кодировка» выбираем «Сохранить в Windows – 1251».
В блокноте убираем из файла лишние строки.
После этого в Матлаб набираем:
>> open('d:\1\diamond.points') % для открытия файла
>>load('d:\1\diamond.points') % для загрузки файла в рабочую область Матлаб
>>

И получаем результат:



Если при загрузке содержимое файла присвоить переменной “a”, то получим:

```
>> open('d:\1\diamant-1.txt')
>> a=load('d:\1\diamant-1.txt')

a =

  -0.2758   -0.7855    1.0000
  -0.0343   -0.7943    2.0000
   0.2610   -0.5185    2.0000
   0.1630   -0.7815    2.0000
  -0.9547    0.9250    1.0000
   0.5837   -0.6590    1.0000
  -0.2580    0.7823    1.0000
  -0.5062    0.7907    1.0000
  -0.6212    0.3967    1.0000
   0.1328   -0.1213    2.0000
  -0.4108   -0.1948    2.0000
  -0.4725   -0.5430    1.0000
   0.7087    0.5933    1.0000
   0.6445   -0.9520    1.0000
  -0.6820   -0.0135    2.0000
```

0.7215 -0.1103 2.0000

...

только файл может оказаться очень длинным, не обязательно его показывать на экране.

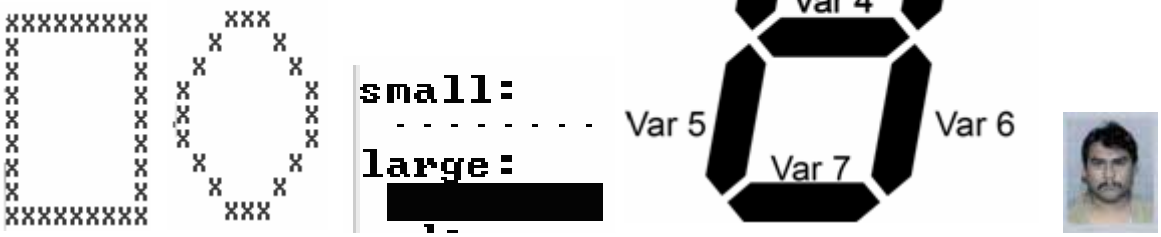
Типовое представление данных в нейронных сетях.

Нейросети наиболее приспособлены к решению широкого круга задач, так или иначе связанных с обработкой образов. Вот список типичных математических постановок задач для нейросетей:

- Аппроксимация функций по набору точек (регрессия).
- Классификация данных по заданному набору классов.
- Кластеризация данных с выявлением заранее неизвестных классов-прототипов.
- Сжатие информации.
- Восстановление утраченных данных.
- Ассоциативная память и ассоциативные процессы
- Оптимизация, оптимальное управление.

Для решения задач классификации могут использоваться либо графические, либо – псевдографические способы отображения.

Графические способы отображения информации.



Табличный способ отображения информации.

Этот способ более универсальный, чаще других используется для решения различных задач:

EX_1.DAT - Блокнот

Файл Правка Формат Вид Справка

Input Gold	Input Yen	Input DM	Input Gold1	Input Yen1	Input DM1	Input Gold2	Input Yen2	Input DM2	Pattern Yen_F
380	158	3.66	390	150	4.02	375	150	3.88	175
390	150	4.02	375	150	3.88	370	150	3.92	158
390	157	3.54	378	150	4.18	381	150	3.78	170
378	150	4.18	381	150	3.78	365	150	3.89	157
392	158	3.67	375	150	4.11	380	150	3.68	180
375	150	4.11	380	150	3.68	372	147	3.78	158
395	170	3.78	374	152	3.88	384	150	3.72	180
374	152	3.88	384	150	3.72	366	146	4.32	170
401	176	3.66	376	159	3.76	389	150	3.87	180
376	159	3.76	389	150	3.87	373	148	4.08	176

Подготовка данных для обработки в нейронной сети.

Перед тем, как данные будут использованы в сети, они должны быть определенным образом подготовлены, при этом должны быть выполнены такие операции, как:

- отбор входных данных
- поиск оптимальных комбинаций входных переменных
- определение цикличности, корреляции и нелинейных зависимостей
- подавление незначимых переменных
- понижение размерности исходных данных
- масштабирование входных и выходных данных (в том числе шкалирование по минимальному/максимальному значениям и по среднему/стандартному отклонению);
- перекодирование переменных с номинальными значениями (например, Пол={Муж,Жен}), в том числе по методу 1-из-N кодирования.
- проведение работы с пропущенными данными
- оценка чувствительности данных

Первичная подготовка данных.

- Проверка цикличности. Эта процедура проводится с помощью специальной процедуры нейросетевого пакета (в пакете BrainMaker с помощью программы Netmaker). Заключается она в контроле, нет ли ярко выраженных пиков на графике. Если – нет, то считается, что данные изменяются случайным образом. В противном случае считается, что данные не случайные, а имеют периодичность, и поэтому необходимо определить период изменения данных.
- Рандомизация строк – для того, чтобы сеть давала более качественные результаты после обучения, строки данных надо располагать случайным образом – при этом исключается систематическая погрешность. Сеть перестаёт зависеть от расположения данных – остаётся зависимость только от значения данных.
- Разделение данных на 3 выборки:
  - **обучающая** (training) {в Brain Maker такие данные располагаются в файле с расширением .fct;
  - **контролирующая** (testing) – она образуется за счёт отрезания (5 – 10)% данных из обучающей выборки; в BM имеет расширение tst;

- **экзаменационная** - для проведения расчётов.

Нормализация значения полей.

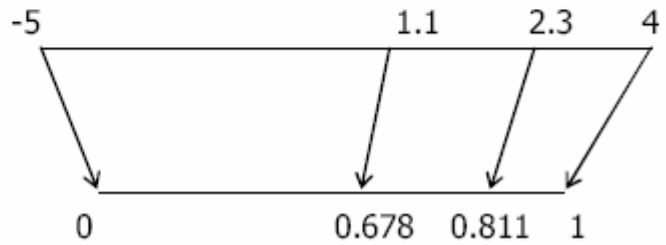
Цель нормализации полей в обучающей выборке – преобразование данных к виду, наиболее подходящему для работы обучающего алгоритма.
Поступающие на вход данные должны быть числового типа, а их значения должны быть распределены в определённом диапазоне.
Рассмотрим три способа нормализации:

- Линейная нормализация; {только для непрерывных числовых полей. Позволяет привести числа к диапазону [min…max]}.
- Уникальные значения; {только для дискретных значений – такими являются строки, числа или даты, заданные дискретно. Непрерывные числа можно перевести в дискретные с помощью процедуры квантования: числа можно расположить в порядке возрастания и перенумеровать. А значения их – заменить порядковым номером}.
- Битовая маска. {для дискретных значений, которые можно только проверять на равенство или неравенство, но нельзя сказать, какое из них больше, а какое – меньше. Все значения записываются порядковыми номерами, а номер записывается в двоичном виде и представляет собой маску, состоящую из 0 и 1}.

1. Линейная нормализация. Приведем значения к диапазону [0..1].

Поле до нормализации	Поле после нормализации
-5	0
2.3	0,81111
1.1	0,67778
4	1
3.5	0,94444

Графически такое преобразование можно представить так.



2. Уникальные значения.

Поле до нормализации	Поле после нормализации
Маленький	1
Средний	2
Большой	3
Огромный	4

3. Битовая маска. Заменим значения битовой маской и приведем к диапазону [-1..1].

Поле до нормализации	Маска	Поля после нормализации	
Москва	00	-1	-1
Воронеж	01	-1	1
Рязань	10	1	-1
Тула	11	1	1

Постпроцессинг.

Возможности Дедактора.

Пример.

Рассмотрим пример построения системы оценки кредитоспособности физического лица. Предположим, что эксперты определили основные факторы, определяющие кредитоспособность. Ими оказались: возраст, образование, площадь квартиры, наличие автомобиля, длительность проживания в данном регионе. В организации была накоплена статистика возвратов или невозвратов взятых кредитов. Эта статистика представлена таблицей.

Сумма кредита	Возраст	Образование	Площадь квартиры	Автомобиль	Срок проживания	Давать кредит
7000	37	Специальное	37	отечественная	22	Да
7500	38	Среднее	29	импортная	12	Да
14500	60	Высшее	34	Нет автомобиля	30	Нет
15000	28	Специальное	14	отечественная	21	Да
32000	59	Специальное	53	отечественная	29	Да
11500	25	Специальное	28	отечественная	9	Да
5000	57	Специальное	18	отечественная	34	Да
61500	29	Высшее	26	Нет автомобиля	18	Нет
13500	37	Специальное	46	отечественная	28	Нет
25000	36	Специальное	20	Нет автомобиля	21	Нет
25500	68	Высшее	45	отечественная	30	Нет

Это обучающая выборка.

Теперь необходимо нормализовать поля. Поля «Сумма кредита», «Возраст», «Площадь квартиры» и «Длительность проживания» – непрерывные значения, которые преобразуем к интервалу [–1..1]. Образование представлено тремя уникальными значениями, которые можно сравнивать на большее или меньшее, а точнее лучшее или худшее, т.е. образование можно упорядочить так: среднее, специальное, высшее. Значения поля с наличием автомобиля упорядочить нельзя. Его нужно преобразовать к битовой маске. Для кодирования трех значений требуется два бита. Следовательно, это поле будет разбито на два.

Наличие автомобиля	Первый бит маски	Второй бит маски
Импортная	0	0
Отечественная	0	1
нет автомобиля	1	0

На этом нормализация закончена.

Обучающую выборку разобьем на обучающее и тестовое множества так, как программа предлага это сделать по умолчанию, т.е. в обучающее множество попадут случайные 95 процентов записей а остальные 5 процентов – в тестовое.

Конфигурация сети будет такой: во входном слое – 7 нейронов, то есть по одному нейрону на оди вход (в обучающей выборке 6 столбцов, но столбец «Автомобиль» представлен битовой маской и двух бит, для каждого из которых создан новый вход). Сделаем один скрытый слой с двумя нейронами. В выходном слое будет один нейрон, на выходе которого будет решение о выдаче кредита.

Выберем алгоритм обучения сети Resilient Propagation с настройками по умолчанию. Условие окончания обучения оставим без изменения.

Обученную таким образом нейросеть можно использовать для принятия решения о выдаче кредит физическому лицу. Это можно сделать, применяя анализ «что-если». Для его включения нужно выбрать визуализатор «Что-если». Тогда откроется форма, представленная на рисунке.

Поле	Значение
Входные	
9.0 Сумма кредита	7000
9.0 Возраст	37
ab Образование	специальное
9.0 Площадь квартиры	37
ab Автомобиль	отечественная
9.0 Срок проживания	22
Выходные	
ab Давать кредит	Да

После изменения в этой таблице входных полей система сама принимает решение о выдаче кредита и в поле «Давать кредит» проставляет либо «Да», либо «Нет». Столбцы «Минимум» и «Максимум» определяют диапазон значений, на которых обучалась нейросеть. Следует придерживаться этих ограничений, хотя и возможно взять значения немного выходящие за границы диапазона.

Кроме такой таблицы анализ «что-если» содержит диаграмму, на которой отображается зависимость выходного поля от одного из входных полей при фиксированных значениях остальных полей. Например, требуется узнать, на какую сумму кредита может рассчитывать человек, обладающий определенными характеристиками. Это можно определить по диаграмме.

То есть, человек в возрасте 37 лет со специальным образованием, имеющий квартиру площадью 37 кв. м, отечественный автомобиль и проживающий в данной местности 22 года может рассчитывать на сумму кредита не больше 20000.

Качество построенной модели можно определить по таблице сопряженности, которая является одним из визуализаторов.

	Классифицировано		
Фактически	Да	Нет	Итого
Да	50	9	59
Нет	27	63	90
Итого	77	72	149

Чем больше правильно классифицированных записей, тем лучше построенная модель.

Примеры обучающих наборов данных.

Данные в нейронной сети используются прежде всего – для обучения. С этой целью они оформляются в виде обучающей выборки, или обучающего набора данных.

Например, сеть необходимо обучить классификации на два класса по косвенным признакам или обучить прогнозированию.

Примерами таких задач могут служить следующие:

- «Мужчина/женщина»,
- «Студент/преподаватель»,
- «Студенты живущие дома/в общежитии»
- «Возможность тренировки парашютистов»
- «Ирисы Фишера»
- «Как выбирают американских президентов»
- «Выбор оптических линз»
- «Ассоциативный поиск текстовой информации»
- «Оценка кредитоспособности»
- «Уровень развиития демократии в разных странах»
- «Прогнозирование значений инфляции»

- «Экономико-политические процессы в исламском мире»

Мужчина/женщина.

В задаче «Мужчина/женщина» в общем виде Обучающий набор данных может представлять собой таблицу вида:

Инфор мационные (inf)	Исходные показатели (Input)				Результирующий (Pattern)
	Готовите ли вы дома пищу	Как часто вы убираете квартиру	Сколько времени в неделю вы тратите на ремонт автомобиля	Является ли только ваш заработок основным источником дохода семьи	
1	нет	редко	3 часа	да	м
2	Да	всегда	0	нет	ж
…	…	…	…	…	…

В качестве исходных показателей лучше использовать переменные, получающие своё значение при ответе на косвенные вопросы.

Примером косвенного вопроса в задаче «Мужчина/женщина» может служить вопрос «Носите ли Вы дома халат», однако вопросы «Носите ли вы дома юбку» или «Приходится ли Вам по утрам бриться» косвенными считаться не могут.

Возможность тренировки парашютистов.

Принимая во внимание опасность данного вида спорта, тренировка возможна не при любой погоде.

Температура	**Влажность**	**Ветер**	**Облачность**	**Тренировка**
Жарко	Высокая	Нет	Солнце	Нет
Жарко	Высокая	Да	Солнце	Нет
Жарко	Высокая	Нет	Облачность	Да
Норма	Высокая	Нет	Дождь	Да
Холодно	Норма	Нет	Дождь	Да
Холодно	Норма	Да	Дождь	Нет
Холодно	Норма	Да	Облачность	Да
Норма	Высокая	Нет	Солнце	Нет
Холодно	Норма	Нет	Солнце	Да
Норма	Норма	Нет	Дождь	Да
Норма	Норма	Да	Солнце	Да
Норма	Высокая	Да	Облачность	Да
Жарко	Норма	Нет	Облачность	Да
Норма	Высокая	Да	Дождь	Нет

В этой таблице Температура, Влажность, Ветер, Облачность – это исходные (объективнык) показатели, на основе которых принимается решение, можно ли проводить тренировку парашютистов (прыжки) – так называемые «входные переменные».

А колонка «Тренировка» – это решение руководителя (тренера), «выходные, или целевые (target)» переменные.

Пример ирисов Фишера:

Господин Фишер вырастил 3 разновидности Ирисов: Setosa, Virjinic, Versicolor.

Цветки этих ирисов отличаются размерами: (длина (length), ширина (wide)) листьев (petal) и чашелистников (sepal) – всего 4 параметра. При снятии характеристик с 13 случайно отобранных цветков образуется таблица:

№	SEPL	SEPW	PETL	PETW	IRISTYPE
1	5,0	3,3	1,4	0,2	SETOSA
2	6,4	2,8	5,6	2,2	VIRGINIC
3	6,7	3,1	5,6	2,4	VIRGINIC
4	6,3	2,8	5,1	1,5	VIRGINIC
5	4,6	3,4	1,4	0,3	SETOSA
6	6,9	3,1	5,1	2,3	VIRGINIC
7	4,6	3,6	1,0	0,2	SETOSA
8	6,5	3,0	5,2	2,0	VIRGINIC
9	6,5	3,0	5,5	1,8	VIRGINIC
10	5,8	2,7	5,1	1,9	VIRGINIC
11	6,8	3,2	5,9	2,3	VIRGINIC
12	5,1	3,3	1,7	0,5	SETOSA
13	6,2	3,4	5,4	2,3	VIRGINIC

Необходимо провести классификацию отобранных наблюдений и определить, к какому типу относится цветок, обладающий следующими характеристиками: 6,0 3,5 5,4 2,7.

Данные об ирисах имеют несколько интересных особенностей:

1. Один из классов (Iris Setosa) линейно отделим от других. Однако, другие два класса нельзя разделить линейно.
2. Классы Versicolor и Virginica пересекаются, поэтому важно достигнуть наилучшего уровня классификации.
3. Среди четырех переменных существует некоторая избыточность, поэтому можно достигнуть хорошего решения, пользуясь только тремя или двумя переменными, однако неизвестно, какие именно переменные нужно выбрать.

Здесь таблица составлена неправильно – она содержит характеристики только двух разновидностей. Желательно, чтобы таблица содержала примерно одинаковое количество строк, характеризующих различные типы цветков.

sepal – чашелистник
length – длина
wide - ширина
petal - лепесток

№	SEPALLEN	SEPALWID	PETALLEN	PETALWID	IRISTYPE
1	5,0	3,3	1,4	0,2	SETOSA
2	6,4	2,8	5,6	2,2	VIRGINIC
3	6,5	2,8	4,6	1,5	VERSICOL
4	6,7	3,1	5,6	2,4	VIRGINIC
5	6,3	2,8	5,1	1,5	VIRGINIC
6	4,6	3,4	1,4	0,3	SETOSA
7	6,9	3,1	5,1	2,3	VIRGINIC
8	6,2	2,2	4,5	1,5	VERSICOL
9	5,9	3,2	4,8	1,8	VERSICOL

Как выбирают американских президентов

Имеется таблица данных с результатами 31-ой предвыборной ситуации (с 1860 по 1980 г.). Для каждых выборов в таблице содержатся данные по 12-ти бинарным признакам:

1. Правящая партия была у власти более одного срока?
2. Правящая партия получила более 50% голосов на прошлых выборах?
3. В год выборов была активна третья партия?
4. Была серьезная конкуренция при выдвижении кандидата от правящей партии?
5. Кандидат от правящей партии был президентом в год выборов?
6. Был ли год выборов временем спада или депрессии?
7. Был ли рост среднего национального валового продукта на душу населения более 2,1%?
8. Произвел ли правящий президент существенные изменения в политике?
9. Во время правления были существенные социальные волнения?
10. Администрация правящей партии виновна в серьезной ошибке или скандале?
11. Кандидат от правящей партии – национальный герой?
12. Кандидат от оппозиционной партии – национальный герой?

Также в таблице содержится информация о результатах выборов (победе правящей или оппозиционной партии).

Значения бинарных признаков равны -1 (ответ "нет" для входного признака или победа правящей партии) и 1 (ответ "да" для входного признака или победа оппозиции).

Нейронные сети, обученные на этой таблице данных, уверенно предсказывали результаты вторых выборов Рейгана, победу Буша над Дукакисом, обе победы Клинтона.

Рекомендации по выбору оптических линз.

Если на основе этой таблицы провести обучение нейронной сети, то обученная сеть может при предоставлении ей информации о Возрасте, Диагнозе, Астигматизме, Индексе чувствительности дать рекомендацию о предпочтительном типе оптических линз для данного человека.

Возраст	Диагноз	Астигматизм	Индекс чувствительности	Рекомендация
Молодой	Близорукость	Нет	Пониженный	Нет
Молодой	Близорукость	Нет	Нормальный	Мягкие
Молодой	Близорукость	Да	Пониженный	Нет
Молодой	Близорукость	Да	Нормальный	Жесткие
Молодой	Дальнозоркость	Нет	Пониженный	Нет
Молодой	Дальнозоркость	Нет	Нормальный	Мягкие
Молодой	Дальнозоркость	Да	Пониженный	Нет
Молодой	Дальнозоркость	Да	Нормальный	Жесткие
Средний	Близорукость	Нет	Пониженный	Нет
Средний	Близорукость	Нет	Нормальный	Мягкие
Средний	Близорукость	Да	Пониженный	Нет
Средний	Близорукость	Да	Нормальный	Жесткие

Средний	Дальнозоркость	Нет	Пониженный	Нет
Средний	Дальнозоркость	Нет	Нормальный	Мягкие
Средний	Дальнозоркость	Да	Пониженный	Нет
Средний	Дальнозоркость	Да	Нормальный	Нет
Пожилой	Близорукость	Нет	Пониженный	Нет
Пожилой	Близорукость	Нет	Нормальный	Нет
Пожилой	Близорукость	Да	Пониженный	Нет
Пожилой	Близорукость	Да	Нормальный	Жесткие
Пожилой	Дальнозоркость	Нет	Пониженный	Нет
Пожилой	Дальнозоркость	Нет	Нормальный	Мягкие
Пожилой	Дальнозоркость	Да	Пониженный	Нет
Пожилой	Дальнозоркость	Да	Нормальный	Нет

Ассоциативный поиск текстовой информации

Традиционные методы поиска и фильтрации документов были разработаны для библиотечных баз данных ограниченного объема и заранее известной структуры. Создание глобальной сети привело к тому, что число поставщиков информации стало стремительно расти, при том, что публикуемая ими информация не имеет однородной структуры.

Последовавший информационный взрыв стал вызовом стандартным информационным технологиям. Новые масштабы с одной стороны сделали аутсайдерами некоторые ранее конкурентоспособные интеллектуальные технологии, а с другой - стимулировали интенсивные исследования в области статистических методов обработки текстовой информации и новых способов навигации в информационном море.

Нейросети являются перспективным инструментом извлечения статистических закономерностей в текстах, и использования этих закономерностей для прецизионной фильтрации документов.

Примером может служить система категоризации текстов, выделяющая в больших массивах документов базовые *тематические категории*.

Таким образом можно представить содержание любого документа вектором в автоматически формируемом нейросетью пространстве категорий, поиск в котором уже не зависит от конкретных ключевых слов, а определяется именно содержанием документов.

Частным случаем документов являются отдельные слова. Табл. 1 иллюстрирует какие ассоциации у обученной нейросети вызывают различные термины.

Табл. 1 Нейро-ассоциации, полученные при обучении на 8000 аннотаций статей научных конференций общества SPIE

Термин	Слова, ассоциируемые с термином (в порядке убывания степени близости)
THE	OF IN FROM THAT ARE ON TO WHICH FOR AND AS TWO IS BOTH A BY IT WITH BETWEEN ALSO AN RESULTS HAS WITHIN INTO BE TIME USED OR
NEURAL	LEARNING CLASSIFIERS UNSUPERVISED TRAINED BACK-PROPAGATION SUPERVISED NEURONS WEIGHTS TRAINING HIDDEN HOPFIELD BACKPROPAGATION NETWORK IMPULSIVE NETS FEEDFORWARD PREDICTOR NETWORKS

	TEXTURAL SPEAKER TELEPHONE PERCEPTRON LEARN AMBIGUITIES DIGITS MULTIDIMENSIONAL BP MLP CLASSIFIER
CANCER	ORGANS LESIONS THERAPY TUMOR VIVO CAM PHOTOSENSITIZERS TUMORS RAT MOUSE PATIENTS AUTOFLUORESCENCE ADMINISTRATION NECROSIS SENSITIZERS VASCULAR RESECTION ADMINISTERED VITRO CLEARANCE INCUBATION PP ACUTE DRUG BALLOON PROSTATE SKIN DISORDERS EPITHELIAL

Как видно из этих примеров, к артиклю «the» ближайшими оказываются служебные слова: артикли, союзы и т.д. К словам же «neural» и «cancer» ближайшими являются слова из той же предметной области.

Этот пример иллюстрирует эффективное нейросетевое **сжатие информации** – текст характеризуется уже не частотами всех входящих в него слов, которых могут быть многие тысячи, а относительно небольшим числом смысловых категорий.

Векторное представление смысла текстов позволяет количественно определить их тематическую близость, построить системы ассоциативной выборки информации из больших архивов текстовых документов.

Такие системы могли бы существенно облегчить поиск юридических документов, патентной информации и безусловно окажутся востребованы в Internet-приложениях.

http://www.statsoft.ru/statportal/tabID__32/MId__141/ModeID__0/PageID__192/DesktopDefault.aspx

Построение модели поведенческого скоринга

В данном примере рассмотрим схему построения нейросетевой модели для задачи поведенческого скоринга. Поведенческий скоринг (behavior scoring) используется для принятия решений по уже выданным кредитам.

Основные решения, принимаемые с использованием поведенческого скоринга, можно сформулировать следующим образом:

- Предложение новых услуг и улучшение уже предоставляемых услуг.
- Решение, выдавать ли кредитную карту заново после истечения срока действия, или нет.
- Меньший стартовый кредитный лимит или максимальное значение кредита на кредитной карточке.
- Более строгий сбор платежей с нарушителей или отправка данных о них в агентства сбора платежей.
- Повышение кредитного лимита.
- Помещение под наблюдение ввиду потенциальных мошеннических действий и т.д.

В данном примере необходимо оценить кредитоспособность существующих заёмщиков на основании данных о графике погашения кредитов и динамики движения средств на счетах клиента.

Структура данных

Каждого клиента будем характеризовать 22 признаками. 20 переменных относятся к анкетным данным, которые заполняются в анкете для получения кредита. К этим переменным относятся:

- Текущий баланс счета
- Продолжительность в мес
- Назначение кредита
- Сумма кредита
- Объем сбережений
- Время работы на данном рабочем месте
- Семейное положение/пол
- Длительность проживания по текущему адресу
- Возраст в годах
- Число предыдущих кредитов в банке
- Должность

и другие.

На основании перечисленных факторов все клиенты подразделяются на “хороших” и “плохих”. Разбиение на эти группы записано в результирующей переменной - *Кредитоспособность* (Creditability).

Всего имеются данные по 1000 клиентов. При этом, 30% относятся к “плохим”, а остальные 70% - к “хорошим”. Процент невыплат по всей совокупности данных около 3% (данная величина относится к одному месяцу). Элемент таблицы данных показан на рис.1.

Данные: creditsc.sta (25v * 1000c)

	1 Кредито-способность	2 Текущий баланс счета	3 **Продолжи-тельность в мес**	4 Выплаты по предыдущим кредитам	5 Назначение кредита сумма кредита
1	плохой	нет текущего счета	36	нет проблем	переподготовка
2	хороший	нулевой баланс	48	неуверенное	переподготовка
3	плохой	>=200	36	не было кредитов	подержанная м
4	хороший	нет текущего счета	24	выплачены	новая машина
5	хороший	>=200	24	не было кредитов	переподготовка
6	хороший	нулевой баланс	12	не было кредитов	переподготовка
7	плохой	нет текущего счета	30	не было кредитов	подержанная м
8	хороший	нулевой баланс	15	выплачены	предметы мебе.
9	хороший	>=200	15	выплачены	предметы мебе.
10	плохой	нулевой баланс	27	выплачены	предметы мебе.
11	плохой	нулевой баланс	24	не было кредитов	подержанная м
12	плохой	нет текущего счета	18	не было кредитов	ремонт
13	плохой	нулевой баланс	36	нет проблем	переподготовка
14	хороший	>=200	6	нет проблем	переподготовка
15	хороший	нет текущего счета	12	не было кредитов	другой
16	хороший	>=200	42	не было кредитов	предметы мебе.
17	хороший	нет текущего счета	48	не было кредитов	новая машина
18	плохой	нулевой баланс	24	с проблемами	ремонт
19	хороший	нулевой баланс	24	выплачены	другой
20	хороший	нулевой баланс	48	не было кредитов	бизнес

Рис.1. Фрагмент исходной таблице данных

Классификация политических режимов стран на основе индексов развития новых медиа

Поставлена задача классификаци с помощью ANN на основе данных о развитии интернета и локальных условий регламентации медиа.

Предполагается, что обученная сеть способна определять тип политического режима (уровень развития демократии) на основе анализа того, насколько недовольные пользователи интернета могут выражать свои мысли в сети

Ход проведения исследования

В выборку были включены данные об уровне развития демократии, количестве пользователей интернета, уровне удовлетворённости жизнью и индексы свободы слова.

Данные были взяты из различных статистических БД, созданных для Secondary Data Analysys.

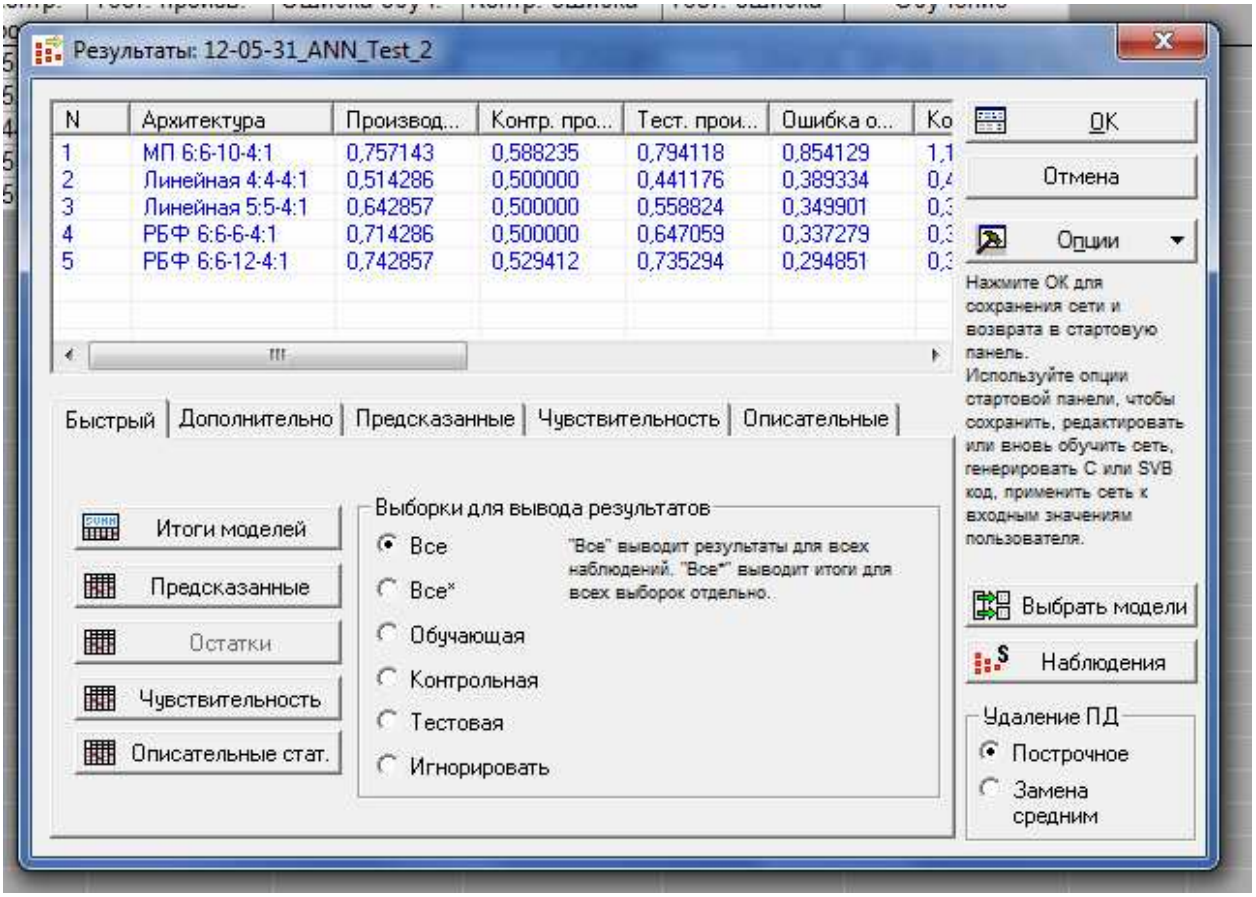
	A	B	C	D	E	F	G	H	I	J
1	Country Name	NNSET	Democracy Index 2011	Democracy Index 2011 Category	IT.NET.USER_2010	IT.NET.USER_2008	Life Satisfaction 2009	Happy Planet Index 2009	Free Press 2010	Free Press 2009
2	Albania	Select	5,81	Hybrid regime	1441927,80	759081,32	5,47	47,91	21,50	21,75
3	Algeria	Select	3,44	Authoritarian regime	4433526,00	3504773,25	5,59	51,23	47,33	49,56
4	Angola	Train	3,32	Authoritarian regime	1908191,20	829746,34	4,27	26,78	28,50	36,50
5	Argentina	Select	6,84	Flawed democracy	14548455,36	11164731,06	7,14	58,95	16,35	11,33
6	Armenia	Train	4,09	Hybrid regime	1360511,68	191211,30	5,03	48,28	27,50	31,13
7	Australia	Train	9,22	Full democracy	16923971,84	15418952,64	7,88	36,64	5,38	3,13
8	Austria	Train	8,49	Full democracy	6102179,19	6078674,03	7,80	47,69	0,50	3,00
9	Azerbaijan	Select	3,15	Authoritarian regime	4226380,18	1527566,88	5,28	41,21	56,38	53,50
10	Bangladesh	Select	5,86	Hybrid regime	5501608,85	3636957,50	5,25	54,09	42,50	37,33
11	Belarus	Select	3,16	Authoritarian regime	3041748,46	2226558,70	5,83	35,67	57,00	59,50
12	Belgium	Train	8,05	Full democracy	8034049,50	6997430,22	7,61	45,36	4,00	2,50
13	Benin	Train	6,06	Flawed democracy	277001,62	154585,63	3,02	24,58	19,00	16,00
14	Bhutan	Train	4,57	Hybrid regime	98727,84	45939,28	6,13	58,50	17,75	15,75
15	Bolivia	Train	5,84	Hybrid regime	1985969,80	1041679,87	6,50	49,35	28,13	24,17
16	Bosnia and Herzegovina	Train	5,24	Hybrid regime	1955277,48	1308125,24	5,90	44,96	13,50	10,50
17	Botswana	Select	7,63	Flawed democracy	120416,70	122176,38	4,70	20,85	17,50	15,50
18	Brazil	Select	7,12	Flawed democracy	79245740,06	64799077,08	7,57	61,01	16,60	15,88
19	Bulgaria	Train	6,78	Flawed democracy	3464629,68	3011477,44	5,47	42,04	19,00	15,61

Для анализа был использован многослойный перцептрон пакета Statistica.

В качестве выходных категорий была использована переменная «Тип политического режима»,

зависимыми переменными считаются данные по пользователям, удовлетворенности жизнью и благополучием, индексы свободы слова.

Было создано 10 нейронных сетей, 5 из них были сохранены. Одна из сетей показала удовлетворительный результат.



Вывод

Предполагается, что обученная сеть способна определять тип политического режима (уровень развития демократии) на основе анализа насколько недовольные пользователи интернета могут выражать свои мысли в сети.

В дальнейшем с помощью математической статистики предполагается обработать полученные результаты и произвести трактовку важности факторов.

ИТОГОВАЯ РАБОТА ПО ТЕМЕ: «ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ В ПРОГНОЗИРОВАНИИ ИНФЛЯЦИИ»

Целью данной работы являлось прогнозирование значений инфляции с помощью нейронных сетей, основываясь на прошлые данные по инфляции. Методы построения инфляционных ожиданий – одна из важнейших тем в современной макроэкономике. От того, как экономические агенты прогнозируют инфляцию, зависят её действительные значения. Инфляционные ожидания входят в значительное количество моделей монетарной экономики. Существует множество методов и подходов для моделирования прогнозирования агентами инфляционной динамики, например, learning-модели, различные поведенческие способы и т.д. Использование нейронных сетей в прогнозировании будущих значений инфляции в перспективе, возможно, позволит более точно моделировать поведение людей, поскольку сами нейронные сети имеют некоторые похожие на человеческое восприятие и мышление способности.

Для данного исследования использовался модуль нейронных сетей в пакете программ «STATISTICA 7». В качестве рассматриваемых данных была взята статистика из

базы МВФ по годовой инфляции (по ценам потребителей) в США, Боливии и России за различные периоды времени.

С помощью нейронных сетей делались и проверялись прогнозы для различных случаев инфляционной динамики, в том числе, и гиперинфляции в Боливии. На основе полученных и проверенных прогнозов делались выводы о качестве работы нейронных сетей и их пригодности для прогнозирования.

Подготовка обучающего набора данных.

- Необходимо составить вопросник из 20 косвенных вопросов, по ответам на которые, с точки зрения студента, возможно провести разделение. Список вопросов утверждается преподавателем.
- Необходимо проанкетировать не менее 20 человек по составленному вопроснику.

    - Обучающий набор данных представляет собой набор *наблюдений*, для которых указаны значения входных и выходных *переменных*.
    - Первый вопрос, который нужно решить, - какие переменные использовать и сколько (и каких) наблюдений собрать. Выбор переменных (по крайней мере первоначальный) осуществляется интуитивно. Ваш опыт работы в данной предметной области поможет определить, какие переменные являются важными. Вы можете произвольно выбирать переменные и отменять предыдущий выбор. Для начала имеет смысл включить все переменные, которые, по Вашему мнению, могут влиять на результат - на последующих этапах можно сократить это множество.
    - Нейронные сети могут работать с числовыми данными, лежащими в определенном ограниченном диапазоне. Это создает проблемы в случаях, когда данные имеют нестандартный масштаб, когда в них имеются пропущенные значения, и когда данные являются нечисловыми. Числовые данные масштабируются в подходящий для сети диапазон, а пропущенные значения можно заменить на среднее значение (или на другую статистику) этой переменной по всем имеющимся обучающим примерам.
    - Более трудной задачей является работа с данными нечислового характера. Чаще всего нечисловые данные бывают представлены в виде номинальных переменных типа *Пол* = {*Муж* , *Жен* }. Переменные с номинальными значениями можно представить в числовом виде. Однако, нейронные сети не дают хороших результатов при работе с номинальными переменными, которые могут принимать много разных значений.

Пусть, например, мы хотим научить нейронную сеть оценивать стоимость объектов недвижимости. Цена дома очень сильно зависит от того, в каком районе города он расположен. Город может быть подразделен на несколько десятков районов, имеющих собственные названия, и кажется естественным ввести для обозначения района переменную с номинальными значениями.

К сожалению, в этом случае обучить нейронную сеть будет очень трудно, и вместо этого лучше присвоить каждому району определенный рейтинг (основываясь на экспертных оценках).

- Во многих реальных задачах приходится иметь дело с не вполне достоверными или недоопределёнными данными. Такие данные могут сопровождаться коэффициентом доверия. Иногда (как в BrainMaker) коэффициент доверия

выражается в виде термометра и может существенно уточнять исходные данные.

- Значения некоторых переменных могут быть искажены шумом или частично отсутствовать. Если у Вас не так много данных, Вы можете включить в рассмотрение случаи с пропущенными значениями (хотя, конечно, лучше этого избегать).
- Нейронные сети в целом устойчивы к шумам. Однако у этой устойчивости есть предел. Например, выбросы, т.е. значения, лежащие очень далеко от области нормальных значений некоторой переменной, могут исказить результат обучения. В таких случаях лучше всего постараться обнаружить и удалить эти выбросы (либо удалив соответствующие наблюдения, либо преобразовав выбросы в пропущенные значения).

Ранжирование входов

При прогнозировании, существенным для качества является учет реального влияния каждого параметра входа $x(x_1, ..., x_n)$ на выходной вектор y. С помощью корреляционного анализа заранее вычисляются коэффициенты парной корреляции между выходом y и каждым из параметров входа $x_1, ..., x_j, ..., x_n$, что позволяет сформировать входную матрицу согласно степени влияния каждого параметра и применить принцип ранжирования входов, что согласовывается со строением биологического нейрона. В нейросеть вводится единый параметр для всех входов сети - коэффициент взвешивания K_f, принимающий значение в диапазоне от 0 до 1.

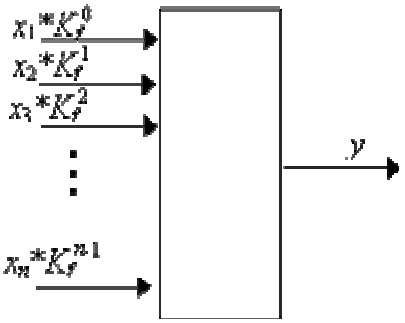


Рис. 1. Влияние коэффициента взвешивания входов

Для 1 входа все значения параметра x_1 не изменяются, для 2 входа уменьшаются в K_f^1 раз, а для последнего n-го входа вес параметра x_n уменьшается в K_f^{n-1} раз (рис. 1). При K_f=1 все входы равнозначны, при K_f=0 учитывается лишь первый вход, остальные входы игнорируется, при $0<K_f<1$ уменьшается влияние несущественных параметров на выходную величину y.

Такой подход требует проведения предварительного анализа информации, но значительное улучшение точности прогноза подтверждает его эффективность.

Выбор данных для обработки

Любая нейронная сеть принимает на входе числовые значения и выдает на выходе также числовые значения. Передаточная функция для каждого элемента сети обычно выбирается таким образом, чтобы ее входной аргумент мог принимать произвольные значения, а выходные значения лежали бы в строго ограниченном диапазоне. При любых входных значениях возникает эффект насыщения, когда элемент чувствителен лишь к входным значениям, лежщим в ограниченной области (например, S - функции). В этом

случае выходное значение всегда будет лежать в интервале (0,1), а область чувствительности для входов едва шире интервала (-1,+1). Данная функция является гладкой, а ее производная легко вычисляется - это обстоятельство важно для работы алгоритма обучения сети (в этом также кроется причина того, что пороговая функция для этой цели практически не используется).

При использовании нейронных сетей могут возникать некоторые проблемы, в частности:

- данные имеют нестандартный масштаб,
- данные являются нечисловыми.
- в данных имеется пропущенное или недостоверное значение.

Числовые данные масштабируются в принятый для сети диапазон. Обычно данные масштабируются по линейной шкале. В пакетах программных нейросетей реализованы алгоритмы, которые автоматически находят масштабирующие параметры для преобразования числовых значений в нужный диапазон.

Более сложной задачей является работа с данными нечислового характера. Пусть, нужно научить нейросеть оценивать стоимость объектов недвижимости. Цена дома зависит от того, в каком районе города он расположен. Город может быть разделен на несколько десятков районов, имеющих собственные названия, и естественно ввести для обозначения района переменную с номинальными значениями. К сожалению, в этом случае научить нейронну сеть будет трудно, и вместо этого лучше присвоить каждому района определенный ранг (базируясь на экспертных оценках).

Чаще всего нечисловые данные представляют в виде номинальных переменных. Номинальные переменные могут быть двузначными (например, Пол ={Мужчина, Женщина}) или многозначными (то есть принимать больше двух значений). Двузначную номинальную переменную легко превратить в числовую (например, Мужчина = 0, Женщина = 1). С многозначными номинальными переменными сложнее. Их тоже можно представить одним числовым значениям (например, Собака = 0, Мышь = 1, Кошка = 2), однако при этом возможно ошибочное присвоение значений номинальной переменной: в рассмотренном примере Мышь окажется чем-то средним между Собакой и Кошкой. Существует более точный способ, известный как кодирование 1-из-N, в котором одна номинальная переменная представляется несколькими числовыми переменными. Количество числовых переменных равняется числу возможных значений номинальной переменной; при этом всякий раз только одна из N переменных принимает ненулевое значение (например, Собака = {1,0,0}, Мышь = {0,1,0}, Кошка = {0,0,1}). К сожалению, номинальная переменная с большим числом возможных состояний требует при кодировании методом 1-из-N очень большого количества числовых переменных, а это приводит к росту размеров сети и создает трудности при ее обучении. В таких ситуациях лучше попробовать найти другой способ представления данных.

Нечисловые данные других типов можно или превратить в числовую форму, или объявить незначительными. Значение дат и времени, если они нужны, можно превратить в числовые, отнимая из них начальную дату (время). Значение денежных сумм превратить совсем несложно. С произвольными текстовыми полями (например, фамилиями людей) работать нельзя и их нужно сделать незначительными.

Во многих реальных задачах приходится иметь дело с не совсем достоверными данными. Значения некоторых переменных могут быть испорчены шумом или частично отсутствовать. Существуют специальные средства работы с пропущенными значениями (они могут быть заменены на среднее значение этой переменной или на другие ее статистики), так что если данных немного, можно включить в рассмотрение случаи с пропущенными

значениями. Нейронные сети в основном стойкие к шумам. Однако у этой стойкости есть граница. Например, выбросы, то есть значения, которые лежат очень далеко от области нормальных значений переменной, могут исказить результат обучения. В таких случаях лучше всего постараться найти и обнаружить эти выбросы (изъять соответствующие примеры или превратить выбросы в пропущенные значения). Если выбросы обнаружить тяжело, то можно воспользоваться возможностями сделать процесс обучения стойким к выбросам, однако такое обучения, как правило, менее эффективно.

Сглаживание данных

Положительный эффект достигается при использовании дополнительной нейросети, функционирующей в режиме сглаживания входных данных обучающего множества. В режиме обучения дополнительной сети каждая реализация обучающего множества приобретает вид: вектор входных значений®вектор входных значений (рис. 3).

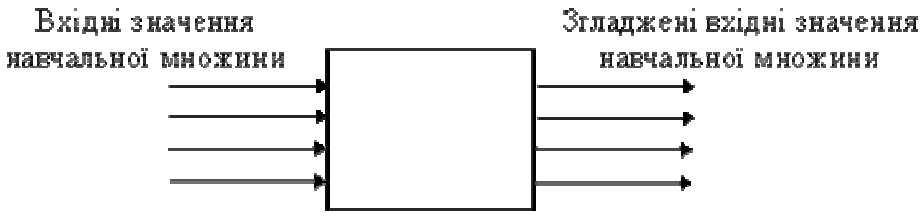


Рис. 3. Пример применения нейросети для сглаживания данных

В режиме функционирования на входы подаются входные значения обучающего множества, на выходе получаем сглаженные значения, без имеющихся выбросов, которые в дальнейшем можно использовать для обработки. Можно дать следующее пояснение эффекта сглаживание данных. Зависимость выходных значений нейросети от входных может быть представлена суммарным степенным полиномом, так как передаточные функции нейронов скрытого слоя - полиномиальные. При незначительном числе нейронов скрытого слоя и невысоких степенях полиномов суммарный полином будет невысокой степени, приводит к сглаживанию.

Вопрос о том, сколько примеров нужно иметь для обучение сети, часто непростой. Известно ряд правил, которые согласовывают число необходимых примеров с размерами сети (простейшее из них говорит, что число примеров должно быть в десять раз больше числа связей в сети). На самом деле, это число зависит также от сложности отображения, которое нейронная сеть стремится воссоздать. С ростом количества параметров количество необходимых примеров растет нелинейно, так что уже при довольно небольшом числе параметров может понадобиться огромное число примеров.

Для большинства реальных задач бывает достаточно нескольких сотен или тысяч примеров. Для сложных задач может понадобиться еще большее количество, однако редко встречается задача, где хватило бы менее сотни примеров. Если данных меньше, то информации для обучение сети недостаточно.

Лекция 4. Архитектура нейросетевых программных систем.

Нейросетевые уровни моделирования и типы решаемых задач.

Структурный подход к моделированию мозга реализуется на нескольких уровнях (этапах).

1. Вначале создается информационная модель отдельной нервной клетки – искусственного нейрона (ИН), что составляет первый уровень нейронного моделирования.
2. Ограниченное число ИН далее могут структурироваться в жесткие необучаемые конфигурации – искусственные нейронные ансамбли (ИНА), что составляет второй уровень нейронного моделирования. В их состав входят ИНА, реализующие функции
   - получения модуля разности двух сигналов,
   - выбора максимального или минимального входного сигнала,
   - эквивалентности (равенства) входных сигналов,
   - классификации
   - ранжирования (сортировки),
   - и др.
3. Наконец, создаются конфигурации из большого числа ИН, которые с помощью специальной процедуры обучения могут гибко изменять свои параметры. Такие конфигурации называются искусственными нейронными сетями (ИНС). Они составляют третий уровень нейронного моделирования.
4. На четвёртом уровне создаются комплексы, содержащие большое количество нейронных сетей различного назначения и оформляются в виде нейросетевых систем управления.

На первом уровне нейронного моделирования действуют модели искусственных нейронов следующих типов:

- формальный нейрон
- нейрон МакКаллока-Питтса
- сигмоидальный нейрон
- нейрон типа "адалайн"
- паде-нейрон
- нейрона Хебба
- нейроны типа WTA (Winner Takes All — "Победитель получает все")
- нейроны с квадратичным сумматором,
- сигма-Пи нейроны,
- стохастические нейроны,
- кубические модели нейронов,
- и др.

Независимо от расположения и функционального назначения, все искусственные нейронные элементы имеют общие компоненты. Среди основных компонентов искусственного нейрона можно выделить следующие:

- Весовые коэффициенты
- Функция сумматора
- Передаточная функция
- Масштабирование
- Выходная функция (соревнование)

- Функция погрешности и распространяемого назад значения (Back Propagation)
- Функция обучения

В нервной системе, особенно в ее периферических отделах, существуют устойчивые, генетически предопределенные конфигурации нервных клеток – *нейронные ансамбли* или *ганглии*, функции которых обычно ограничены и предопределены спецификой периферического отдела в организме.

В практике нейронного моделирования в ряде случаев также оказывается полезным рассматривать ограниченную совокупность ИН как *искусственный нейронный ансамбль* (ИНА), который имеет жесткую необучаемую структуру, определяемую задачей обработки информации.

Понятие ИНА позволяет расширить ограниченный набор вычислительных возможностей одиночного ИН. Переход от одиночного ИН к ИНА можно рассматривать как второй уровень нейронного моделирования.

С точки зрения решения прикладных задач, использование необучаемой «нейронной логики» на основе ИНА вместо традиционной компьютерной логики эквивалентен замене одного функционально полного базиса другим функционально полным базисом. Такая замена не порождает новых уровней функциональности и методов решения задач, и может быть оправдана лишь более эффективной реализацией вычислителя.

В базисе нейронной логики специалистами по нейронному моделированию были предложены решения самых разнообразных задач, которые по эффективности реализации могли конкурировать с вычислителями на обычной логике.

В качестве примера, рассмотрим использование ИНА для реализации специальных логических функций: определения модуля, выделения минимума и максимума, установления факта эквивалентности др.

На рисунке приведена схема ИНА, реализующего функцию получения модуля разности двух сигналов

$$z = z_4 = |x_1 - x_2|$$

В ИНА в качестве нейронов Н1−Н4 используются Градуальные Нейроны в отличие от функционального нейрона (ФН), оперирующего булевыми переменными, в *градуальном нейроне* (ГН) входные и выходные сигналы являются действительными числами, изменяющимися в определенных диапазонах):

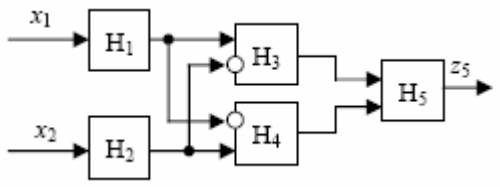


Рис. ИНА, реализующий функцию получения модуля разности двух сигналов

На рис. приведена схема ИНА, реализующего функцию выбора максимального входного сигнала.

$$z = z_4 = \max(x_1, x_2)$$

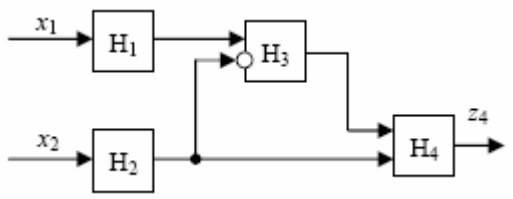


Рис. ИНА, реализующий функцию выбора максимума двух сигналов

На следующем рисунке отображён искусственный нейронный ансамбль, реализующий функцию эквивалентности (равенства) входных сигналов.

$$z = z_4 = \text{eqv}(x_1, x_2)$$

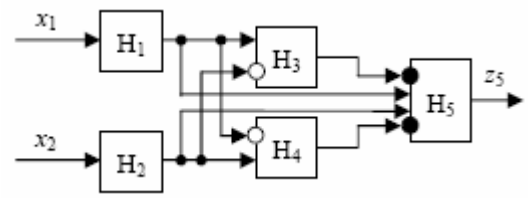


Рис. ИНА, реализующий функцию эквивалентности двух сигналов

Другим примером применения ИНА является реализация функций классификации и ранжирования (сортировки).

Пусть, например, необходимо отслеживать и классифицировать соотношение между двумя сигналами $x1$ и $x2$, а точнее − фиксировать факты: $x1 < x2$; $x1 > x2$; $x1 = x2$.

Логика работы такого ИНА описывается выражением

$$\mathbf{Z} = f(x_1, x_2) = \{z_5, z_6, z_7\} = \begin{cases} \{1\ 0\ 0\}, \text{ если } x_1 > x_2; \\ \{0\ 1\ 0\}, \text{ если } x_1 = x_2; \\ \{0\ 0\ 1\}, \text{ если } x_1 < x_2. \end{cases}$$

В базисе «нейронной логики» такая задача решается с помощью ИНА, показанном на следующем рисунке:

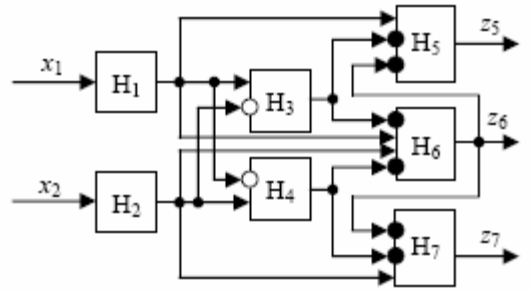


Рис. ИНА, реализующий функцию классификации входных сигналов

Кроме вычислительных ИНА, копирующих элементы ЭВМ, нейронные сети могут реализовать элементы, реализующие функции нейроматематики: элементы для выполнения математических операций, например таких, как

- сложение, вычитание, умножение, деление различных чисел,
- преобразования чисел из одной системы счисления в другую,
- перекодировки текста,
- матричных операций,
- генерации случайных чисел,

- построения гистограмм.

В каждом из этих случаев создаётся небольшая нейросеть и с помощью универсального нейропакета обучается выполнению необходимой операции. Затем обученная нейросеть извлекается из обучавшего её нейросетевого пакета и сохраняется в виде исходного модуля на каком-либо алгоритмическом языке, либо после компиляции сохраняется в виде исполняемого файла, который может быть включён в состав создаваемой программы.

Одной из главных целей нейронного моделирования является использование принципов построения и функционирования мозга для решения практических задач по обработке информации, трудно поддающихся решению другими средствами.

Эта цель реализуется путем создания и использования нейронных конфигураций, которые имитируют некоторые важные свойства, присущие естественному интеллекту, такие как

- обобщение,
- обучение,
- распознавание,
- принятие решений
- и др.

Объединение ИН в такие конфигурации фактически порождает новый уровень функциональности программирования, отличный от использующихся возможностей традиционных компьютеров.

Новый (третий) уровень функциональности нейронных моделей возникает на больших нейронных конфигурациях, в которых воспроизводится свойство обучения. Они называются *искусственными нейронными сетями* (ИНС).

Специалистами по нейронному моделированию предложено множество типов ИНС, отличающихся типом ИН, структурой связей, методами обучения, функциональным назначением.

Такие сети не являются специализированными, имеют универсальный характер и предназначены для решения различных задач. Примером простейшей нейросети, обладающей указанными возможностями, является однослойная нейронная сеть с одним рабочим слоем:

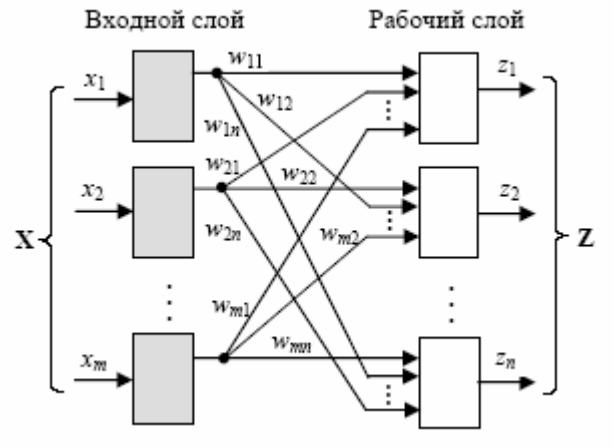


Рис. Однослойная ИНС

В процессе функционирования сети происходит преобразование входного вектора сигналов в выходной. Конкретный вид преобразования определяется

- архитектурой нейросети

- характеристиками нейронных элементов,
- средствами управления
- синхронизацией информационных потоков между нейронами.

Важным фактором эффективности сети является установление оптимального количества нейронов и типов связей между ними.

При описании нейросетей используют несколько терминов, которые в разных источниках могут иметь разное трактование, в частности:

- структура нейросети - способ связей нейронов в нейросети;
- архитектура нейросети - структура нейросети и типы нейронов;
- парадигма нейросети - способ обучения и использования; иногда содержит и понятие архитектуры.

На основе одной архитектуры могут быть реализованы разные парадигмы нейросети и наоборот.

Среди известных архитектурных решений выделяют группу слабосвязанных нейронных сетей, когда каждый нейрон сети связан лишь с соседними.

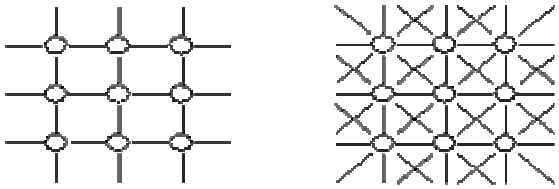


Рис. Слабосвязанные нейросети

Наоборот, если входы каждого нейрона связаны с выходами всех остальных нейронов, тогда речь идет про полносвязанные нейросети.

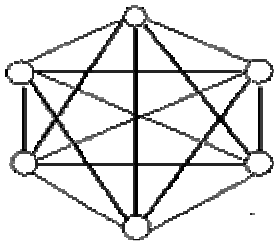


Рис. Полносвязанная сеть

Понятно, что такое деление носит теоретический характер. Анализируя наиболее известные на данное время разработки нейросетей, следует отметить, что самым распространенным вариантом архитектуры являются многослойные сети. Нейроны в данном случае объединяются в слои с единым вектором входных сигналов.

Внешний входной вектор подается на входной слой нейронной сети (рецепторы). Выходами нейронной сети являются выходные сигналы последнего слоя (эффекторы). Кроме входного и выходного слоев, нейросеть имеет один или несколько скрытых слоев нейронов, не имеющих контактов с внешней средой.

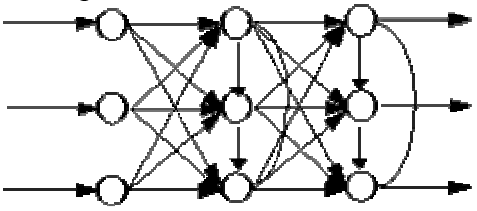


Рис. Многослойный тип соединения нейронов

Для многослойных сетей характерно:

- Связи между нейронами разных слоев называют проективными.
- Связи направленные от входных слоев к выходным называются афферентными.
- При обратном направлении связей - эфферентными.
- Связи между нейронами одного слоя - боковыми (латеральными).

По архитектуре связей, большинство известных нейросетей можно сгруппировать в два больших класса:

- Сети прямого распространения (с однонаправленными последовательными связями).
- Сети обратного распространения (с рекурентными связями).

Классификация известных нейросетей по основным категориям применения

- Перцептрон Розенблатта
- Нейросеть обратного распространение погрешности (Back Propagation)
- Delta Bar Delta
- Extended Delta Bar Delta
- Направленный случайный поиск
- Нейронная сеть высшего порядка или функционально - связанная нейронная сеть
- Сеть Кохонена
- Квантование обучающего вектора (Learning VectorQuantization)
- Сеть встречного распространения (CounterPropagation)
- Вероятностная нейронная сеть.
- Сеть Хопфилда
- Машина Больцмана
- Сеть Хемминга
- Двунаправленная ассоциативная память
- Сеть адаптивной резонансной теории

Сети прямого распространения относят к статическим, так как на заданные входы нейронов поступает независимый от предыдущего состояния сети вектор входных сигналов.

Рекурентные сети считаются динамическими, так как за счет обратных связей (петель) входы нейронов модифицируются во времени, что приводит к изменению состояния сети.

Оригинальность нейросетей, как аналога биологического мозга, состоит в способности к обучению по примерам, составляющих обучающее множество.

Процесс обучения нейросетей рассматривается как настраивание архитектуры и весовых коэффициентов синаптических связей в соответствии с данными обучающего множества так, чтобы эффективно решать поставленную задачу.

Типы задач, решаемых с помощью нейропакетов и средств для проведения нейросетевых исследований.

В процессе работы нейросети осуществляют оценивание и предсказание поведения объектов, в том числе систем и процессов, заданных определенными законами и совокупностью своих реализаций.

Каждая реализация должна содержать набор признаков, определяющих основное содержание объекта. Если одним из признаков объекта исследования является время, тогда реализации могут быть представлены в виде временных рядов.

У большинства реальных объектов исследования можно выделить их основные составляющие:

- детерминированная составляющая, которая в принципе подлежит точному предсказанию;
- вероятностная составляющая, которую можно предсказать с заданной степенью вероятности;
- чисто случайная составляющая, которую невозможно ни учесть, ни предусмотреть.

В зависимости от степени влияния той или другой составляющей, можно говорить об определенном типе множеств данных, используемых для обучения нейросетей:

- детерминированное множество данных, со всеми учтенными основными параметрами, вызванными действием известных причин. Оно характеризуется малым уровнем шумов;
- множество данных с наличием вероятностной составляющей, вытекающей из экспериментальной постановки задачи, с разной степенью учета действующих факторов и с влиянием погрешностей оценивания;
- множество данных с наличием случайной составляющей, вследствие неучтенных признаков явления.

Решаемые задачи можно разбить на два основных класса:

- классификация;
- регрессия.

В задачах классификации нужно определить, к какому из имеющихся классов принадлежит входной набор. Примерами могут служить предоставление кредита, диагностика заболеваний, распознавание образов.

В задачах регрессии пытаются предвидеть значение переменной, принимающей непрерывные числовые значения: цена акций, затраты топлива в автомобиле, прибыль кампании и т.п.

Предвидение явлений можно поделить на:

- предвидение выходов для множества дискретных входных данных, не связанных со временем (экономические, социологические оценки и др.); данные представлены таблично;
- прогнозирование явлений, непрерывно изменяющихся во времени (физические процессы, естественные явления, и т.п.); данные представлены в виде временных рядов.

Задачи регрессии

Особое значение имеют задачи предвидения и прогнозирования временных рядов, среди которых выделяются задачи с набором определенных специфических признаков, поэтому следует провести их классификацию. Задачи исследования явлений, развитие которых связано со временем, можно поделить на несколько классов:

По характеру основных признаков объекта:

- прогнозирование явлений, реализации которых представленные в виде детерминированных временных рядов. Такие задачи, в частности, можно решить путем применения методов математического анализа;

- прогнозирование явлений, реализации которых представлены в виде индетерминированных временных рядов. Решение этих задач традиционно осуществляется путем применения методов теории вероятностей и математической статистики. В частности, реализации таких явлений, могут иметь вид:

а) стационарного временного ряда, который характеризуется однородностью во времени, без существенных изменений характера колебаний и их средней амплитуды; выбор промежутка для формирования обучающего множества произвольный; в качестве примера такого ряда на рис. приведен график суммарного годового стока Днепра за период с 1810 до 1964 года;

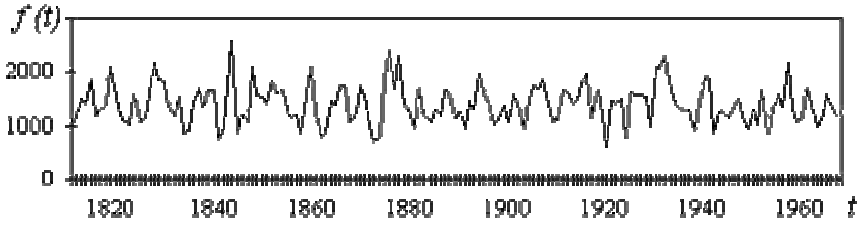


Рис. График суммарного годового стока Днепра за период с 1810 до 1964 года

б) нестационарного временного ряда, который характеризуется определенной тенденцией развития в времени; при исследовании нестационарных процессов можно выделить участки, на которых процесс можно считать стационарным; выбор промежутка для формирования обучающего множества в таком случае избирается согласно задаче прогнозирования.
На рис. приведены ежедневные нормированные данные микросейсмических колебаний Земли за определенный период времени.

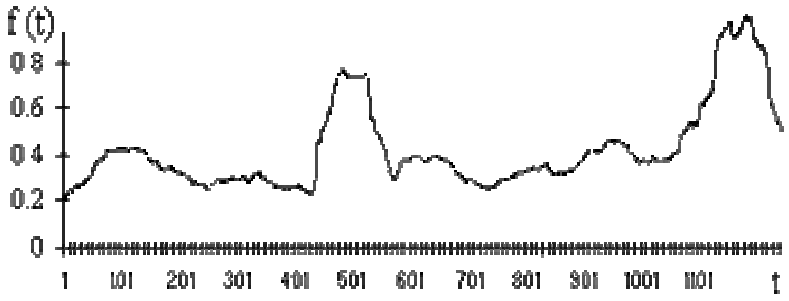


Рис. Ежедневные нормированные данные микросейсмических колебаний Земли за 1000 лет

По числу признаков объекта исследований:

- одномерная задача; явление представлено лишь одним признаком, изменения которого происходят во времени; (на рис. изображены данные наблюдений относительных чисел Вольфа, усредненные за месяц, за период с 1900 по 1924 год);

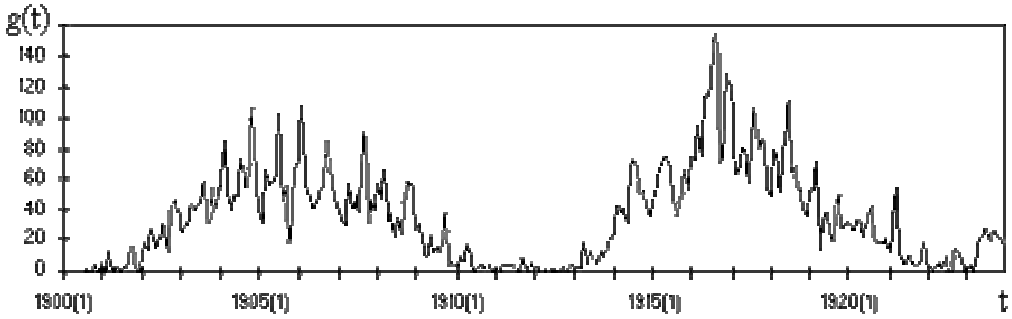


Рис. Распределение чисел Вольфа во времени

- многомерная задача; объект или явление представлены несколькими признаками; задача прогнозирования может быть расширена благодаря представлению данных в пространстве.

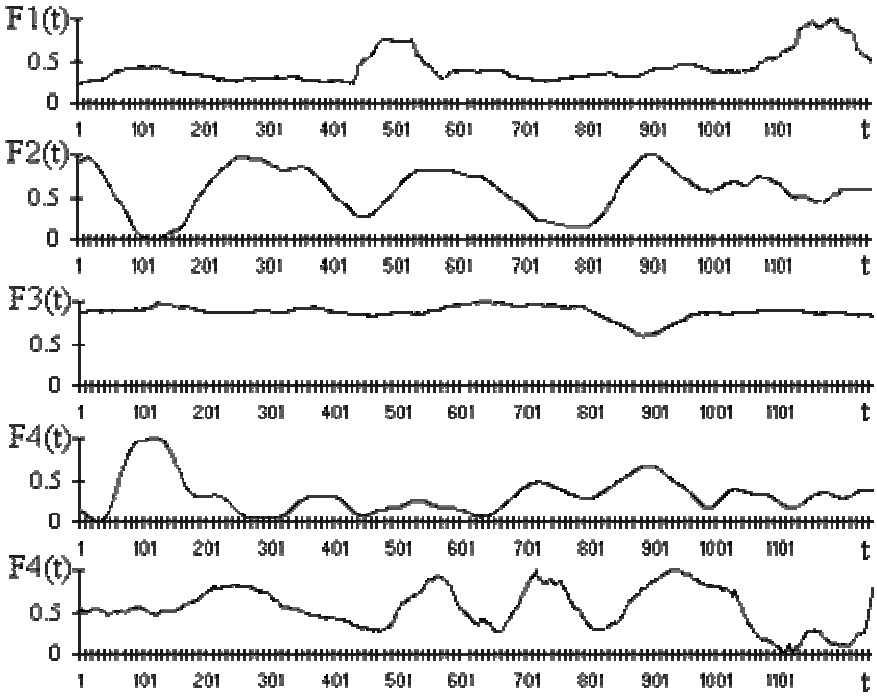


Рис. Основные предвестники сейсмической активности

На рис. представлены ежедневные данные некоторых из основных признаков, определяющих землетрясение. Показанные данные, в частности:

- F1 - акоистика,
- F2 - деформации земной поверхности,
- F3 - микросейсмические колебания Земли,
- F4 - суммарная энергия землетрясения,
- F5 - температура Земли,

составляют признаки, измеренные в одинаковые отсчеты времени за определенный период и в дальнейшем использовались для экспериментов.

В качестве примера многомерной задачи покажем оценивания числа групп солнечных пятен, усредненных за год, которые определяются двумерным распределением в области двух аргументов - широта-время (рис.).

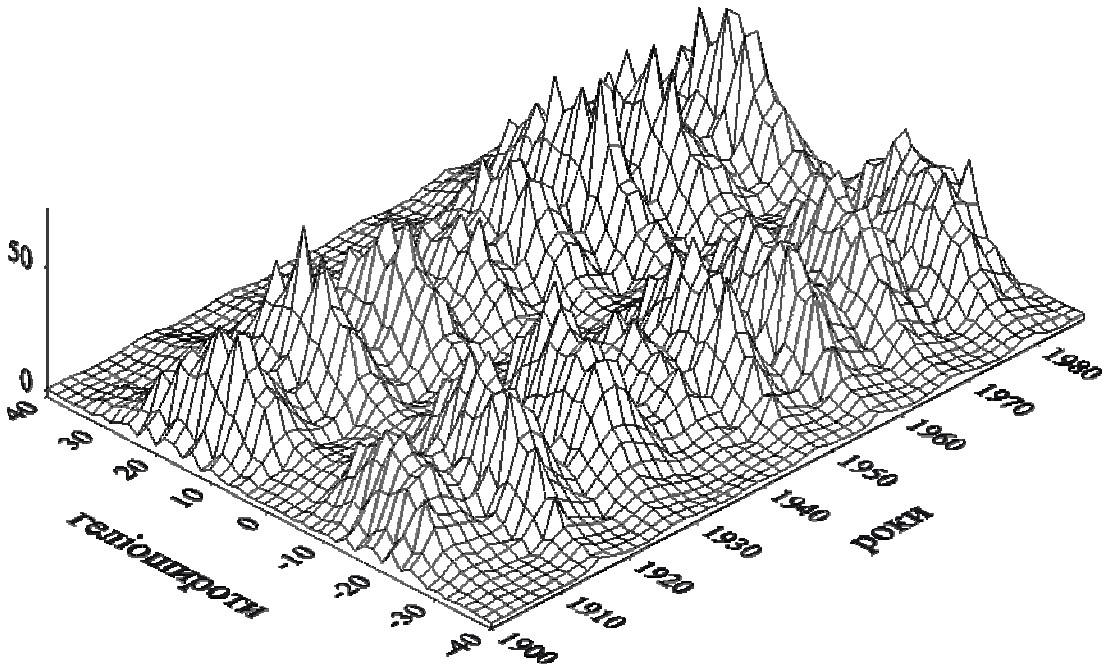


Рис. Пространственное отображение количества солнечных пятен в координатах гелиоширота-время

Учитывая специфический характер прогнозирования временных рядов и определенный разнобой в терминологии, будем придерживаться ряда определений.

Предысторией ряда назовем набор элементов временного ряда, который учитывается для одного шага прогнозирования следующих элементов временного ряда.

Одношаговое прогнозирование сводится к задачам отображения в случае, когда значение элементов предыстории могут определять лишь один дискретный отсчет выходных величин.

Многошаговое прогнозирование характеризуется увеличением дискретных отсчетов выходной величины и, соответственно, увеличением времени, на который осуществляется прогноз (время опережения T_{on}). При многошаговом прогнозировании $T_{on}=a*R$, где R - количество шагов вычисления прогнозирования; а - шаг дискретизации выходного параметра (например, год, месяц, день, и т.п.).

По времени опережения различают виды прогнозов:

- сглаживание, R= 0;
- краткосрочный прогноз, R= 1 - 2;
- середнесрочный прогноз, R= 3 - 7;
- долгосрочный прогноз, R= 10 - 15.

Очевидно, что вид прогноза существенно влияет на выбор средств и методику его реализации.

Лекция 5. Классификация нейропакетов и средств для проведения нейросетевых исследований.

Нейроматематика имеет дело со сложными и разнообразными конструкциями:

1. Нейроны
    - формальный нейрон
    - нейрон МакКаллока-Питтса
    - сигмоидальный нейрон
    - нейрон типа "адалайн"
    - паде-нейрон
    - нейрона Хебба
    - нейроны типа WTA (Winner Takes All — "Победитель получает все")
    - нейроны с квадратичным сумматором,
    - сигма-Пи нейроны,
    - стохастические нейроны,
    - кубические модели нейронов,
    - и др.

    Среди основных компонентов искусственного нейрона можно выделить следующие:
    - Весовые коэффициенты
    - Функция сумматора
    - Передаточная функция
    - Масштабирование
    - Выходная функция (соревнование)
    - Функция погрешности и распространяемого назад значения (Back Propagation)
    - Функция обучения

2. Нейронные ансамбли (аппаратные или программные).
    - сложение, вычитание, умножение, деление различных чисел,
    - преобразования чисел из одной системы счисления в другую,
    - перекодировки текста,
    - матричных операций,
    - генерации случайных чисел,
    - построения гистограмм.

3. Нейронные сети.

    Необходимо различать процессы обучения нейронной сети и использования обученной сети.

    При использовании обученной сети происходит только решение сетью определенной задачи. При этом синаптическая карта сети остается неизменной.

    Работу сети при решении задачи будем далее называть **прямым функционированием.**

    Нейронные сети характеризуются:
    - архитектурой нейросети

- характеристиками нейронных элементов,
- средствами управления
- синхронизацией информационных потоков между нейронами.

Классификация нейронных сетей:

- Сети прямого распространения (с однонаправленными последовательными связями).
- Сети обратного распространения (с рекурентными связями).

Разновидности нейронных сетей

- Перцептрон Розенблатта
- Нейросеть обратного распространение погрешности (Back Propagation)
- Delta Bar Delta
- Extended Delta Bar Delta
- Направленный случайный поиск
- Нейронная сеть высшего порядка или функционально - связанная нейронная сеть
- Сеть Кохонена
- Сеть с квантованием обучающего вектора (Learning VectorQuantization)
- Сеть встречного распространения (CounterPropagation)
- Вероятностная нейронная сеть.
- Сеть Хопфилда
- Машина Больцмана
- Сеть Хемминга
- Двунаправленная ассоциативная память
- Сеть адаптивной резонансной теории

4. Комплексы нейронных сетей:

- управляющие нейросети
- прогнозирующие системы оптимизации
- специализированные конструкции для распознавания графических объектов
- специализированные конструкции для распознавания звуковых объектов

Все эти конструкции обслуживаются такими программными средствами:

1. Комплексы программ для конструирования и изучения свойств нейронов
2. Программные средства для изготовления нейронных ансамблей (вычислительные модули; экспертные системы, анализаторы причинно-следственных отношений; и др.)
3. Нейромодели
4. Нейросетевые имитаторы
5. Нейросетевой конструктор
6. Нейросетевые комплексы (системы оптимизации)

Все они входят в группу нейропакетов. Первые два типа имеют вспомогательное значение и используются лишь при изготовлении элементов нейросетевых комплексов.

Среди доступных (так называемых «свободнораспространяемых – freeware») инструментов к числу нейромоделей можно отнести модель многослойного перцептрона (Sharky: http://www.shatktime.com) и модель нейросети Кохонена; к числу нейросетевых имитаторов относятся «Пермский нейросетевой имитатор» (http://www.lbai.ru/#;show;course)

и нейроимитатор Штудгартского университета. Нейросетевым конструктором можно считать пакет MemBrain и специально создаваемые для этого на каком-либо алгоритмическом языке пространства имён (например, Neural Networks for C#). К нейросетевым комплексам относятся специализированные комплексы, в состав которых «Свёрточные сети», нейронные сети «Deep learning», «Azur», «Caffe» и др.

К условно доступным нейросетевым имитаторам можно отнести демоверсии Deductor и Neurosolutions.

У перечисленных программных средств (нейропакетов) иногда можно выделить специфические для нейропакетов модули

1. Генераторы Нейронных Сетей - редакторы визуального проектирования.
2. Генераторы исходного кода обученной нейросети, допускающие выделение созданного кода из нейропакета.
3. Средства подготовки данных для нейросетевых исследований (конверторы данных, пакеты математической статистики).
4. Внешние модули для расширения возможностей нейропакета.
5. Ансамбли нейронных сетей, такие, которые создаются пакетом «Статистика neural networks – SNN».

Возможности пакета Neural Bench.

Программный пакет Neural Bench для Windows 95 (http://neuralbench.com/RUS/INDEX.HTM) состоит из следующих модулей:

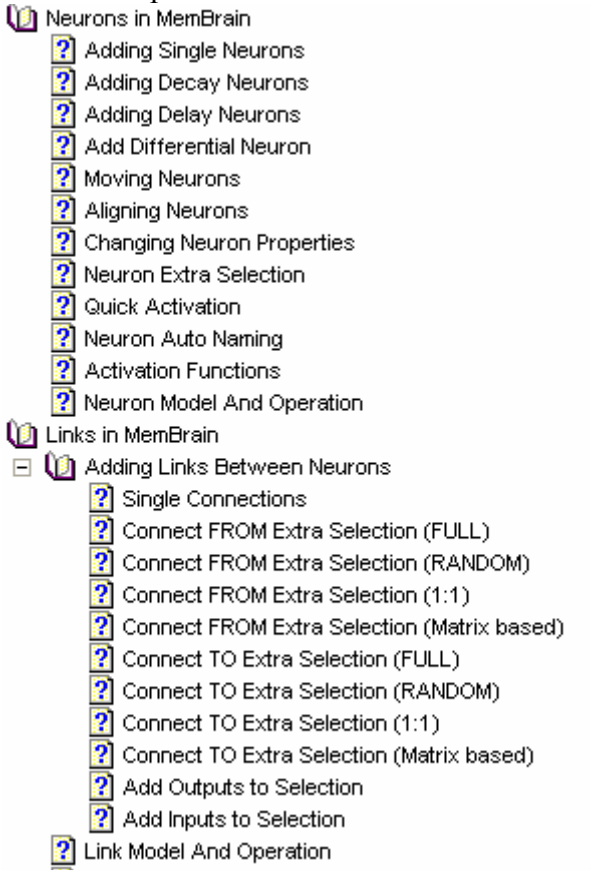
1. **нейросетевой эмулятор Neural Bench** - мощный математический интуитивный в использовании пакет.
   В него входят:
   - Генерация топологий и выбор нейроматематики;
   - Масштабирование;
   - Рандомизация;
   - Редактирование свойств на уровнях отдельных связей, нейронов и слоев;
   - Оптимизация топологии;
   - Фоновый режим обучения,
   - Тестирование,
   - Пользовательский интерфейс с нейросетью,
   - Генератор исходного кода
   - И др.
2. Мастер обработки исходной и результирующей информации **Data Processor**, обеспечивающий множество вариантов анализа и тестирования данных. В него входит
   - Мастер Данных (Data Wizard), позволяющий создавать обучающие и тестирующие выборки,
   - Графический редактор, обеспечивающий возможность построения графиков,
   - Блок, обеспечивающий совместимость с MS Excel , и др.
3. Специальное приложение, позволяющее пользователю создавать собственные окна диалога для взаимодействия с нейронной сетью **Dialog Editor**, которое обеспечивает редактирование и просмотр данных.

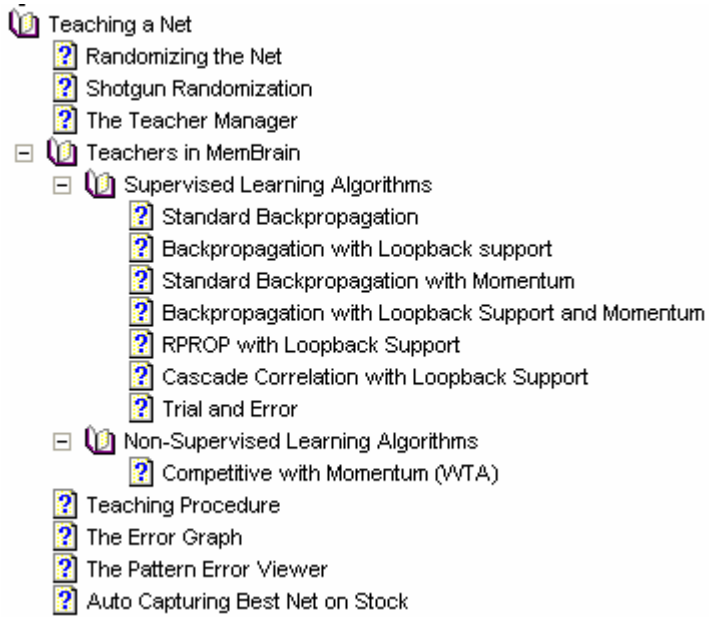
Neural Bench существует только в 32 разрядной версии для Microsoft Windows 95 или Microsoft Windows NT.

Нейропакет MemBrain.

MemBrain - мощный графический нейросетевой редактор - имитатор для Microsoft Windows, поддерживающий искусственные нейросети произвольного размера и архитектуры. Его адрес: http://www.membrain-nn.de/english/download_en.htm .

Нейропакет позволяет использовать:





Основные характеристики пакета MemBrain позволяют:

- Передачу подготовленных нейросетей в промышленные программные системы (динамические библиотеки dll) или автоматическую генерацию C-Code обученной нейросети.
- Формирование «на-лету» результатов обучения нейросети на заранее подготовленных данных;
- Обучение «с учителем» и «без учителя»;
- Моделирование сложных нейросетей с учётом временных связей и динамики их изменения;
- Внедрение объектов высокой сложности с учётом языка сценариев.

MemBrain обеспечивает максимальную гибкость разработки и исследования искусственных нейросетей, созданных с помощью интуитивного графического интерфейса пользователя.

К настоящему времени пакет используется многими университетами для исследования и обучения. В то же время, в течение последних лет наблюдается повышение интереса к использованию его в промышленном производстве и технических управляющих приложениях.

Большинство приложений связано с отображением корреляции между входами и выходами неизвестных или недостаточно известных данных в искусственных нейронных сетях. Целью является способность предсказания выходов системы на основе тестирования входов без знания передаточной функции. Знание этих функций часто невозможно совсем, или по крайней мере недоступно для получения законными средствами.

Исключительно важное требование для достижения этой цели – достаточное количество вводимых и выходных пар данных, которые могли быть получены из реальной системы или могли уже быть доступны из исторических записей. Эти данные могут быть импортированы в MemBrain и могут быть использованы для обучения искусственных нейронных сетей.

Как только нейронная сеть разработана и успешно обучена, она может быть использована, чтобы аппроксимировать реальную систему для использования в прогнозах или других вычислениях: новые величины прикладываются ко входам сети - целевые выходные величины вычисляются сетью и доступны через выходные порты сети.

Используя MemBrain, такая сеть может легко быть преобразована в промышленные системы, независимо от того, являются они основными или вложенными системами.

Основные системы могут использовать dll - версию MemBrain для загрузки и обучения нейросетей из файлов. Dll так же позволяет проводить обучение сетей вместе с пользовательскими приложениями.

Для вложенных систем MemBrain даёт возможность использовать встроенный исходный кодовый генератор C, который позволяет генерировать код, ограниченый типичными требованиями вложенных систем: небольшой объём используемой памяти и отказ от динамического распределения памяти.

Из-за своих гибких дисплейных свойств MemBrain применим как для больших сетей, использующих много нейронов и связей, так же и для небольших сетей, состоящих только из нескольких нейронов и может быть использован для подробного обследования, которое производится на самом низком уровне.

MemBrain использует мощный язык C++, как язык описания, который позволяет описать всё - от простых пакетных процессов до сложных управляющих программ для того, чтобы автоматически запускать часто используемые последовательности операторов или чтобы достигать определённого потребителем поведения.

Программа MemBrain и все связанное с ней вспомогательное программное обеспечение включая MemBrain DLL, распространяются по принципу "как есть". Никакая гарантия, любого типа, ни - явная или неявная, не даётся.

Вы используете пакет на свой риск. Томас Jetter не будет ответственным за потерю данных, убытки и упущенную выгоду если убыток вызывался использованием или неправильным употреблением этого программного обеспечения.

Что представляет собой SNNS.

SNNS (Stuttgart Neural Network Simulator) – это программный имитатор нейросетей для работы под Unix, разработанный в институте параллельных и распределённых высокоуровневых систем (IPVR) Штудгардского университета. Цель проекта SNNS – создание эффективной и гибкой моделирующей среды для научных исследований в области нейронных сетей. Его адрес: http://www.ra.cs.uni-tuebingen.de/SNNS/

Имитатор SNNS состоит из двух основных компонент:

1) ядро имитатора, написанное на C
2) графический интерфейс пользователя под X11R4 или X11R5.

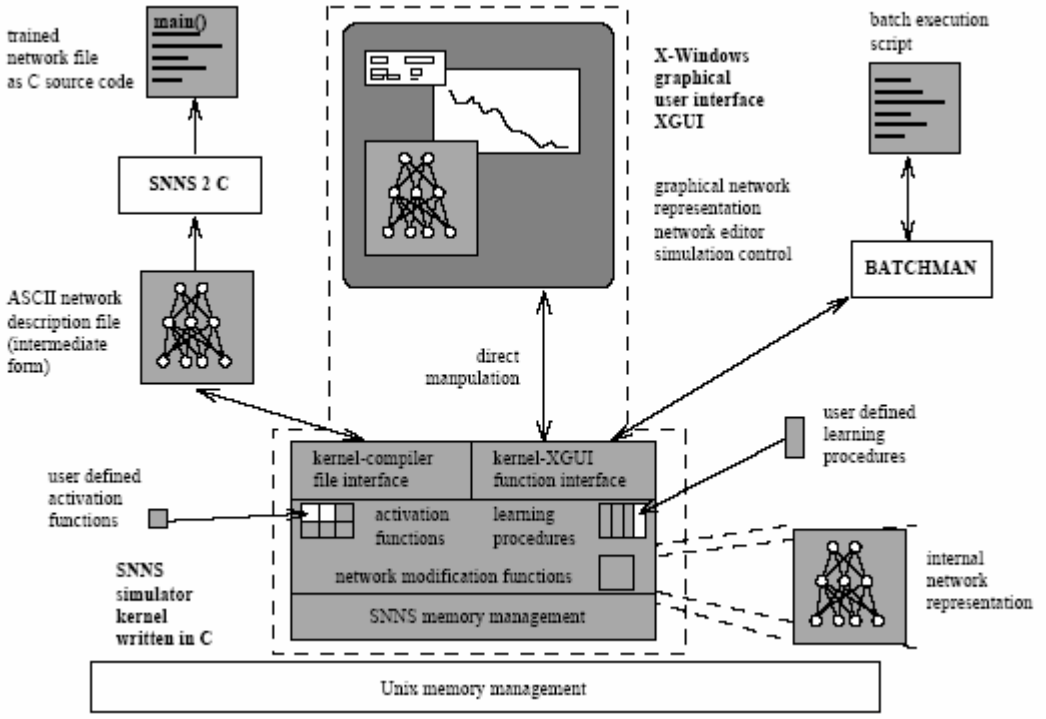


Figure 1.1: SNNS components: simulator kernel, graphical user interface xgui, batchman, and network compiler snns2c

Ядро симулятора действует на внутрисетевых структурах данных нейронной сети и выполняет все операции обучения и вспоминания (восстановления). Оно может так же использоваться без другой части, как C-программа, вставленная в обычное приложение. Оно поддерживает произвольную сетевую топологию, и поддерживает концепцию сайта.

SNNS может быть расширен пользователем определёнными им функциями, внешними функциями, определёнными сайтом функциями и обучающими процедурами, которые написаны в виде простых C программ и внедрёнными в ядро имитатора.

В настоящее время в имитаторе используются следующие нейросетевые архитектуры и обучающие процедуры:

- Backpropagation (BP) для нейронных сетей прямого распространения
- Counterpropagation
- Quickprop
- Backpercolation 1
- RProp
- Generalized radial basis functions (RBF)
- ART1
- ART2
- ARTMAP
- Cascade Correlation
- Recurrent Cascade Correlation
- Dynamic LVQ
- Backpropagation through time (for recurrent networks)
- Quickprop through time (for recurrent networks)
- Self-organizing maps (Kohonen maps)
- TDNN (time-delay networks) with Backpropagation
- Jordan networks
- Elman networks and extended hierarchical Elman networks
- Associative Memory

Графический интерфейс пользователя XGUI (X Graphical User Interface), смонтированный в корне ядра, позволяет реализовать 2D и 3D графические представления нейронных сетей и регулировать ядро в течение работы имитатора.

В дополнение, 2D – интерфейс имеет встроенный нейросетевой редактор, который может быть использован непосредственно для создания, манипулирования и визуализацией нейронной сети в различных ситуациях.

Возможности SNNS v4.1

Версия SNNS v. 4.1 представляет следующие улучшения и расширения:

- Новый пакетный язык выполнения batchman
- Интеграция генетического алгоритма инструмента Enzo. Этот инструмент допускает развивающую оптимизацию нейронной сети. Enzo доступен как особый tar-файл на ftp сервере в той же позиции, что и SNNS. По всем исследованиям Enzo обращайтесь к enzo-request@ira.uka.de.
- Улучшенная обработка окна. При переоткрытии окна SNNS, оно размещается наверху взамен окна с сообщением об ошибке.
- Новый обучающий сетевой алгоритм 'scaled conjugated gradient' (SCG) (масштабированный производный градиент)
- Список параметров на панели управления, приспособленный к работе с числовыми параметрами.
- WinSNNS, MS-Windows - пакетное выполнение, не являющееся основным для рабочей станции unix. WinSNNS можно найти на http://www.lans.ece.utexas.edu/winsnns.html. Обратитесь к ghosh@pine.ece.utexas.edu для получения более полной информации.
- Представлен дополнительный параметр принудительного обучения (`Teacher-Forcing') для нейронной сети Jordan.
- Расширенная отладка имее ряд дополнительных опций:
    - Seg-fault при попытке тренировать нейронную сеть без внешнего устройства

- Seg-fault, когда нажимается клавиша info без загрузки образца
- Нарушение петли обратной связи при удалении сети в ассоциативной памяти
- Неправильная активация функции в ассоциативной памяти при создании большой сети
- Неправильная инициализация функции в сети Кохонена при создании большой сети
- Вычислительная ошибка при анализе комплекса функций
- Лучшее шрифтовое приспособление размера предметов этикетки
- Неправильный просмотр параметров коррекции в snnsbat
- Ошибка в руководстве: Потеря параметра BackpropWeightDecay

NeuroSolutions

Пакет NeuroSolutions – один из лучших современных пакетов, предназначен для моделирования широкого круга искусственных нейронных сетей. Его распространителем является сайт http://www.neurosolutions.com/ .

Пакет доступен в демоверсии.

Характеристики пакета:

Operating System: Windows XP (x86/x64), Vista (x86/x64), 7 (x86/x64), Windows 8.1 (x86/x64), or Windows 10 (x86/x64)
Processor: Intel Core Duo or higher (or AMD Compatible)
Memory: 3.00 GB System Memory
Storage: 650 MB (NeuroSolutions Installation)

Фирма поставляет важные расширения к пакету: NeuroSolutions for MATLAB и C++ Code Generations.

Современные направления развития нейрокомпьютерных технологий.

В настоящее время в отрасли ИТ происходит переход к новой технологической парадигме. В основе всех существующих компьютеров с 1940-х гг. лежала архитектура с разделенным процессингом и памятью. Ее принципиальным ограничением является неспособность к самостоятельному ассоциированию и синтезу нового знания.

Сегодня на рынке представлены сотни нейральных симуляторов самого разного уровня исполнения и возможностей. В сети представлено множество их компаративных обзоров.

Большинство из них поддерживает лишь ограниченное число возможных для построения стандартных архитектур классических нейросетей и методов при очень небольшом числе (менее тысячи) нейронов, которые можно включить в сеть. К таким, довольно слабым, инструментам относятся и Neural Network Toolbox MATLAB и SNNS. Подобные средства вполне хороши для решения задач анализа временных рядов или прямого персептронного распознавания для моделирования небольших зрительных полей, но совершенно не годятся для построения нейроморфных когнитивных систем, способных к обработке данных размерности реального мира.

В то же время есть и пакеты, в которых можно строить большие нейросети. Например, пакет NEST (NEural Simulation Tool), представляющий из себя среду для моделирования больших гетерогенных сетей точечных нейронов.

NEST фокусируется на динамике, размерах и структуре нейронных систем, а не на точном представлении морфологии отдельных нейронов. Поэтому NEST подходит для моделирования спайковых сетей любого размера и построения моделей обработки информации (например, в визуальной или слуховой коре, моделирования ламинарных кортикальных сетей, моделирования синаптической пластичности и обучения .

Используя пакет NEST и суперкомпьютер K исследователи из института RIKEN летом 2013 г. построили искусственную сеть из 1.73 млрд. нейронов и 10.4 триллионов синапсов http://www.riken.jp/en/pr/press/2013/20130802_1 .

Пакет NEST, наряду с программными комплексами Neuron и STEPS является одним из трех основных пакетов, используемых уже сегодня для построения Brain Simulation Platform создаваемой в рамках Human Brain Project (Neuron и STEPS в большей мере есть пакеты портретного моделирования нейронов, нежели построения больших сетей для нейрональных вычислений).

Но и возможности этого пакета все же ограничены как по числу синапсов, так и по функционалу моделей. Именно поэтому в рамках первой фазы HBP, которая продлится с октября 2013 по март 2016 в качестве одной из важнейших задач предусмотрено создание программной среды, способной обеспечить процедуры создания и алгоритмы будущих моделей мозга, а также расширение возможностей существующих нейросимуляторов для поддержки очень больших нейросетей, в том числе моделей всего мозга.

В качестве задач для разработки софта также указаны необходимость реализации диффузного распространения сигналов, моделирования биохимических процессов внутри отдельных клеток, прогнозирования кинетики и молекулярных взаимодействий.

Идеальный нейропакет для биоморфного моделирования сигнальных процессов и построения ИКС должен позволять:

- строить сети с максимально возможным числом нейронов на каждый узел кластера (как минимум, это сотни миллионов синапсов на узел);
- обеспечивать высокую скорость кортикальных вычислений, так как обработка даже простого предложения от входа до выхода ассоциативной реакции составляет по кратчайшему пути тысячи синапсов (как минимум, скорость обработки пакетом должна составлять несколько миллионов синапсов в секунду на Ггц тактовой частоты доступных процессоров);
- обеспечивать создание сотен типов произвольных биоморфных моделей нейронов, в т.ч. с многофазным формированием следа (от ионного переноса до структурной перестройки);
- обеспечивать ростовые процессы и развитие создаваемых нейросистем (ибо вручную сети из миллиардов нейронов создать невозможно, даже путем копирования, так как их структура нерегулярна);
- обеспечивать эффективные интерфейсы ввода-вывода основных типов сенсорной информации и моторных реакций (т.е. конструкцию зрительных полей, ввода звука, телеметрии, вывода движений и т.д. - при том, что требования промышленности сегодня в ряде случаев - это микросекунды);
- обеспечивать облачный доступ исследователей и разработчиков к ресурсам суперкомпьютера (например, для ввода отраслевого стандарта представления результатов нейробиологических исследований коннектома, а равно для технических разработок).

Сегодня, в результате прогресса нейронаук, роста производительности суперкопьютеров и появления обучаемых наноматериалов начался переход к новой архитектурной парадигме - ассоциативным искусственным когнитивным системам, способным к самообучению и синтезу нового знания путем ассоциативной рекомбинации полученной информации.

Под "искусственными когнитивными системами" (ИКС) здесь понимаются технические системы, способные к познанию и поведению для решения существующих проблем в условиях реального мира, включая распознавание образов, понимание смысла и контекстуального значения информации, мышление, самостоятельное усвоение знаний из различных источников, обучение, а в необходимых случаях – осуществляющие субъективную оценку информации.

Базовой технологией реализации ИКС являются алгоритмы управления развитием и размножением систем трёх типов:

- нейроморфные (подобные организации живой нервной системы),
- кортикоморфные (подобные организации коры головного мозга) искусственные нейросети и
- генноморфные (подобные генетическим и эпигенетическим механизмам организмов).

ИКС будут созданы на всех видах платформ:

- в виде специализированных систем на базе суперкомпьютеров,
- в виде распределенных систем в глобальных и корпоративных компьютерных сетях,
- в виде автономных технических устройств и роботов,
- в виде систем управления крупными технологическими комплексами и соединениями,
- в виде микросистем и нанокомплексов,
- в виде киберорганических систем.

Для всех этих платформ существуют два основных способа реализации ИКС –

- в виде программ на базе суперкомпьютеров и
- в виде аппаратных устройств и роботов на основе обучаемых наноматериалов.

Важнейшее значение для создания ИКС имеет построение в параллельной программной среде моделей кортикальных колонок и коры мозга - решение задачи кортикального процессинга.

Основными зарубежными проектами создания подобных ИКС являются

- европейские проекты BBP/HBP,
- американская инициатива BRAIN,
- проект IBM DeepQA "Watson",
- проект "Siri" корпорации Apple,
- проект нейросетевого искусственного интеллекта и использующих его роботов компании Google,
- японские проекты JST,
- канадский проект "Spaun" и др.

Создание элементной базы ИКС на основе обучаемых наноматериалов ведется

- по программе DARPA "SyNAPSE" и
- в рамках проекта IBM "Создание когнитивного компьютера на базе синаптроники и суперкомпьютеров" ("C2S2") в результате которого по

прогнозу IBM появятся новый класс технологий и новые отрасли промышленности.

Согласно докладу "Emerging Cognitive Neuroscience and Related Technologies" Национального исследовательского совета (NRC) США по-настоящему интеллектуальные ИКС, способные к самостоятельному обучению и познанию, могут быть созданы в любой из развитых стран Америки, Европы или Азии в ближайшие десять лет (начиная с 2008г.).

Российский проект "Искусственные когнитивные системы (ИКС)"

В 2012г. Министерством образования и науки совместно с Министерством связи и массовых коммуникаций Российской Федерации был проведен конкурс на право создания центров прорывных исследований мирового уровня в области информационных технологий.

В состав победителей конкурса вошли МГУ им. М.В. Ломоносова, Московский физико-технический институт, Высшая школа экономики, Национальный исследовательский ядерный университет - МИФИ, Сколковский институт науки и технологий, Санкт-Петербургский государственный университет, Тюменский государственный университет, Казанский федеральный университет, Институт проблем передачи информации РАН им. А.А. Харкевича и др.

По итогам конкурса Минкомсвязи и Минобрнауки в России создан Центр прорывных ИТ-исследований в сфере разработки искусственных когнитивных систем.

Стратегическая программа создания Центра прорывных исследований в области информационных технологий "Искусственные когнитивные системы" была подготовлена ТюмГУ совместно с высокотехнологичной компанией ООО "Тюменских ассоциативных систем объединение" (ТАСО), учрежденной университетом во исполнение Федерального Закона № 217 от 2 августа 2009 г.

Под "искусственными когнитивными системами" понимаются технические системы, способные к

- познанию, распознаванию образов и самостоятельному усвоению новых знаний из различных источников,
- продолжительному обучению, пониманию контекстуального значения и субъективной оценке получаемой информации,
- синтезу нового знания,
- мышлению и поведению для успешного решения существующих проблем в условиях реального мира.

В основе российского проекта "Искусственные когнитивные системы" лежат разработанные компанией ТАСО

- технологии кортикоморфных нейросетей,
- кибергеномики и программные средства построения сверхбольших нейрогенетических сетей, а также
- технологии создания твердотельных мемристорных микросхем.

Они представлены, в частности, в опубликованных работах (2008) и на сайте taso.pro.

На первом этапе реализации проекта (2014-2016 гг.) идет создание программно-реализованной искусственной коры и развитие программного комплекса интегрированной среды разработки нейрогенетических сетей, динамических моделей сложных объектов и искусственных когнитивных систем.

На первом этапе реализации Стратегической программы (2014-2016 гг.) должно быть обеспечено создание программно-реализованной модели искусственной коры мозга, способной к обработке всех существующих в сигнальных системах типов ассоциативных оснований и решению различных когнитивных задач: от ответов на вопросы до синтеза нового знания.

Создание программно-реализованной искусственной коры обеспечивается через разработку модулей кортикальной колонки для обработки упорядоченных ассоциативных оснований, неупорядоченных ассоциативных оснований, частично-упорядоченных ассоциативных оснований, встроенных ассоциативных оснований, разорванных ассоциативных оснований, а также модулей системы фильтрации существенного и системы торможения неправильного.

Именно эти задачи и позволяет решать программный комплекс - "ТАСО Нейроконструктор" и идущая ему сегодня на смену IDE AI (последовательно создаются версии Parallel IDE AI NeuroStandard, NeuroSensorics, NeuroGenomics , NeuroCloud и др.).

На втором этапе проекта (2017-2018 гг.) создается базовая программно-реализованная искусственная когнитивная система. Для этого к программно-реализованной искусственной коре добавляются модели сетчатки, наружного коленчатого тела, зон V1-MT коры, т.е. зрительная подсистема, слуховая подсистема, подсистема формирования пространственно-временного континуума личности - модель энторинальной коры, субикулума, гиппокампа и зубчатой извилины, подсистема субкортикального управления моторными движениями и модель мозжечка, модуль произвольного и непроизвольного внимания, подсистема многоактного анализа, подсистема целенаправленного поведения, модуль искусственной эмоциональности и субъективности. На этом же этапе путем реализации образовательных и воспитательных программ из ранее созданных программно-реализованных ИКС формируются специализированные ИКС.

В целом в рамках программы выделяются три основных направления. Важнейшим является проведение исследований и разработок в сфере создания искусственных когнитивных систем.

На втором этапе реализации программы (2017-2018 гг.) создается базовая программно-реализованная искусственная когнитивная система, в рамках которой к искусственному кортексу добавляются действующие программныые модели сетчатки, наружного коленчатого тела, стриарной коры (зрительной подсистемы), слуховой подсистемы, модель энторинальной коры, субикулума, гиппокампа и зубчатой извилины (подсистемы формирования пространственно-временного континуума искусственной личности), подсистема двигательного обучения, система фильтрации существенного, модуль произвольного и непроизвольного внимания, модуль машинной субъективности и др. необходимые подсистемы.

На втором этапе также начинаются работы по созданию специализированных искусственных когнитивных систем в интересах предприятий реального сектора экономики и населения.

И на первом, и на втором этапах осуществляется разработка нейроморфных электронных устройств на основе твердотельных неорганических полупроводниковых мемристорных обучаемых наноматериалов (мемристоры на основе тонких пленок диоксида титана были получены в ТюмГУ в 2012 г.).

Решение всех этих задач представляется возможным с использованием уникальных технологий, разработанных компанией ТАСО. Прежде всего - биоморфных моделей нейронов с многофазной консолидацией следа памяти, моделей кортикальных колонок,

кортикоморфных (подобных коре головного мозга) нейронных сетей и кибергеномики - технологии управления ростом и развитием сверхбольших искусственных нейросетей.

Важное значение имеет то, что ТАСО создан программный комплекс "ТАСО-Нейроконструктор" для разработки нейронных сетей.

В рамках программы будут использоваться имеющиеся в ТюмГУ суперкомпьютеры, включая известный "Менделеев", а также технологическая платформа для производства наноэлектроники НТ-МДТ "Нанофаб-100" и другое оборудование ЦКП университета. При этом ТюмГУ осуществляет в рамках Стратегической программы развития Центра теоретические исследования, а ТАСО и вновь создаваемые ИТ-компании ведут прикладные разработки.

Суперкомпьютеры для построения искусственных когнитивных систем

17 декабря 2013 г. в Москве запущен новый суперкомпьютер "Т-Нано", который будет использоваться для размещения программно-реализованных моделей искусственной коры мозга в рамках проектов тюменского Центра прорывных ИТ-исследований "Искусственные когнитивные системы".

Пиковая производительность кластера "Т-Нано" производства российской компании "Т-Платформы" составила 220 терафлопс. Суперкомпьютер займет 7 место в списке самых мощных российских суперкомпьютеров и войдет в список 500 самых мощных суперкомпьютеров мира.

Запуск кластера состоялся в рамках открытия нанотехнологического центра "Т-Нано".

Кроме суперкомпьютера "Т-Нано", созданные нейрогенетические искусственные когнитивные системы будут размещаться на суперкомпьютере "Менделеев" Тюменского государственного университета (11 терафлопс), также построенном компанией "Т-Платформы". Разработка систем ведется на кластере ООО "ТАСО".

На снимках:

1. Суперкомпьютер "Т-Нано".
2. Суперкомпьютер "Менделеев".

Лекция 6. Нейроматематика.

Нейроматематика – раздел вычислительной математики, связанный с разработкой методов и алгоритмов решения задач в нейросетевом логическом базисе.

Нейронным (или *нейросетевым) алгоритмом* будем называть вычислительную процедуру, основная часть которой может быть реализована в виде нейронной сети той или иной структуры.

Нейронные сети – наилучший аппроксиматор функций.
Для классической теории аппроксимации, являющейся основой вычислительной математики для ЭВМ с архитектурой фон-Неймана, аппроксимируемая функция представляется в следующем виде:

$$y(x) = \sum_i a_i \psi_i(x) .$$

Здесь набор функций Ψ_i, выбирается исходя из конкретного метода аппроксимации и некоторых свойств данных функций.
Для нейронных сетей базовое выражение для аппроксимируемой функции выглядит несколько иначе, например, для трехслойной нейронной сети с последовательными связями:

$$y(x) = \psi\left(\sum_i a_i \psi\left(\sum_j a_j \psi\left(\sum_k a_{kj} x_k\right)\right)\right).$$

В последнем выражении структура эквивалентных базовых функций Ψ_i представлена в нейросетевом логическом базисе с множеством коэффициентов, которые являются настраиваемыми в процессе поиска наилучшей аппроксимации.
Подобное представление аппроксимированной функции может быть при необходимости усложнено как увеличением числа слоев, изменением числа нейронов в каждом слое нейронной сети, введением перекрестных связей в структуре нейронной сети, так и таким широко используемым в последние годы способом, как введение обратных связей в структуре нейронной сети (или, по определению зарубежных авторов, введение понятия рекуррентных нейронных сетей).
Преимущества нейросетевого представления аппроксимируемой функции заключается в следующем:

- большая гибкость базовых функций $\Psi_i(x)$, связанная с адаптацией ко входным данным;
- высокая потенциальная параллельность выполняемых операций на нейронных сетях;
- однородность базовых операций на нейронных сетях (умножение, сложение, вычисление простейшей нелинейной функции Ψ_i);
- технологическая простота выполнения данных простейших операций при различных способах физической реализации;
- возможность управления числами элементов суммирования по j и k для каждого i, которые также являются предметом адаптации в процессе поиска наилучшей аппроксимации;
- возможность использования нейросетевых структур для априорного представления функции $y(x)$.

Первыми попытками в 60-70-е годы решения задач на нейрокомпьютерах были задачи распознавания образов, которые можно сформулировать как задачи аппроксимации функций (K классов образов), существующих в многомерном пространстве признаков.
Затем следовали некоторые попытки решения других классических задач, например, обращения матриц.
В конце 80-х годов в связи с активизацией разработки нейрокомпьютеров резко расширился круг задач, решаемых с их помощью.

В настоящее время круг таких задач настолько широк, что можно открыто говорить о потенциальной универсальности нейрокомпьютеров, а также о том, что любую математическую задачу можно решить в нейросетевом логическом базисе.

Даже такие, казалось бы, простые задачи, как сложение чисел, умножение, деление, извлечение корня, обращение чисел и подобные им многие пытаются решить с помощью нейрокомпьютеров, так как при ориентации на нейросетевую физическую реализацию алгоритмов эти операции можно реализовать значительно эффективнее, чем на известных булевых элементах.

Реализуются в нейросетевом логическом базисе и применяются на практике различные функциональные преобразования, для выполнения которых на классических ЭВМ необходима разработка специальных алгоритмов, требующих для реализации значительного времени.
Это – прямые и обратные тригонометрические, экспоненциальные функции, в нейронной сети – функции активации.

Во многих научных работах разрабатываются и исследуются нейросетевые алгоритмы решения следующих задач:

1. системы линейных уравнений и неравенств;
2. обращение матриц;
3. задачи оптимизации (линейное и нелинейное программирование);
4. сортировка;
5. решение задач общей нейроматематики (обыкновенных дифференциальных уравнений с произвольной нелинейной правой частью, дифференциальных уравнений в частных производных).

Отдельным важным разделом общей нейроматематики является комплекс задач, связанных с формализацией при помощи графов, в частности

1. задачи поиска и подсчета путей, циклов и разрезов в графах,
2. задачи разбиения,
3. задачи об укладке и раскраске графов.

Обозначения

Множество исходных данных	$\mathbf{D}$
Множество величин, подлежащих определению	$\mathbf{R}$
Входной сигнал нейронной сети	$\mathbf{x}$
Его размерность	N
Входной сигнал j-го слоя[1]) нейронной сети	$\mathbf{x}^j$
Его размерность	N^j
Число слоев	J
Число ЛПЭ в j-м слое	M^j
Матрица весовых коэффициентов j-го слоя	$\mathbf{A}^j$
Вектор смещения ЛПЭ j-го слоя	$\mathbf{b}^j$
Аналоговый выход j-го слоя	$\mathbf{g}^j$
Функция активации m-го ЛПЭ j-го слоя (в случае однородной сети нижний индекс обозначает собственный номер функции в статье)	$f_m^j(\mathbf{g}^j)$
Выход j-го слоя	$\mathbf{y}^j$
Требуемый (желаемый, идеальный) выход системы	$\overline{\mathbf{y}}$

[1]) Если сеть однослойная, то индексы номера слоя (j) не пишутся.

1. Классы задач, решаемых нейрокомпьютером.

Целью создания нейроматематики является создание алгоритмов с высокой степенью распараллеленности как для формализуемых, так и для трудно- и не- формализуемых задач.
Критерием эффективности нейронных алгоритмов будет ускорение решения задач по сравнению с традиционными методами. Сравнение алгоритмов по эффективности для различных способов реализации нейрокомпьютеров является предметом самостоятельного рассмотрения и отдельной работы.

2. Нейронные сети.

При разработке нейронных алгоритмов решения задач используются положения теории нейронных сетей, основы которой были заложены в работах [1-6],
Определим типовые вычислительные алгоритмические процедуры, которые будут использоваться для построения нейронных алгоритмов.
В качестве аналога нейрона рассмотрим вычислительный элемент, в дальнейшем называемый *линейным пороговым элементом* (ЛПЭ) и определяемый соотношением

$$y = f\left(\sum_{n=1}^{N} a_n x_n + b\right).$$

Дадим описание входящих в это соотношение величин:
x = *(xu x2,. . . , xN)* — входной сигнал (входной вектор) ЛПЭ;
a = *(ar, a2,. . ., aN)* — вектор (набор) весовых коэффициентов ЛПЭ;
N — размерность входного сигнала;
b - значение смещения (эквивалентное *a0* ,если принять, что *x0* = 1) ;
/ (•) — функция активации, являющаяся нелинейным преобразованием своего аргумента; иногда в литературе применяют термины "пороговая функция" или просто "нелинейное преобразование";
y — выходной сигнал ЛПЭ.
Схема функционирования ЛПЭ изображена на рис. 1.
ЛПЭ объединяются в слои. *Слой* представляет собой совокупность ЛПЭ с единым входным сигналом, не имеющих связей между собой.

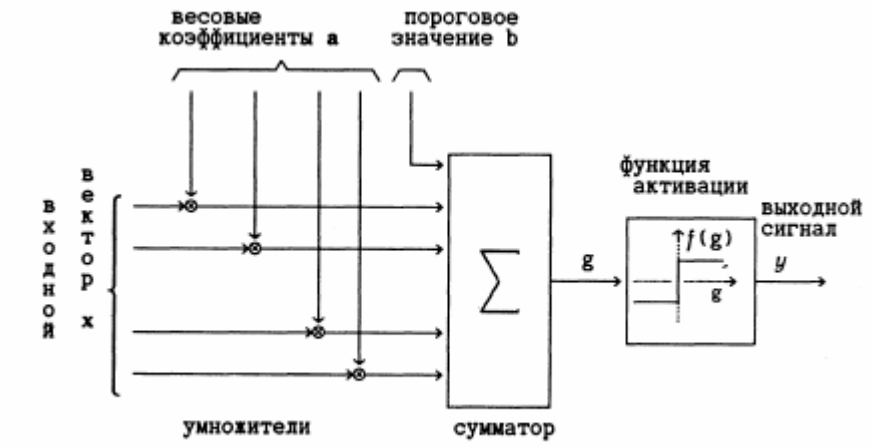


Рис. 1. Схема линейного порогового элемента

Соотношение, связывающее вход и выход m-го ЛПЭ в слое, записывается в следующем виде:

$$y_m = f_m\left(\sum_{n=1}^{N} a_{mn}x_n + b_m\right), \quad m = \overline{1,M}, \tag{2}$$

где

M – число ЛПЭ в слое,

m – номер ЛПЭ в слое,

$\mathbf{x} = (x_1, x_2, \ldots, x_n)$ – входной сигнал слоя ЛПЭ,

$\mathbf{A} = (a_{mn})_{n=\overline{1,N}}^{m=\overline{1,M}}$ – матрица весовых коэффициентов слоя ЛПЭ,

$\mathbf{b} = (b_1, b_2, \ldots, b_M)$ – вектор смещения,

$f_m(\cdot)$ – функция активации m-го ЛПЭ в слое,

y_m – выходной сигнал m-го ЛПЭ в слое.

Слои ЛПЭ могут объединяться в сеть. Соотношение вход–выход m-го ЛПЭ j-го слоя сети запищется в следующем виде:

$$y^j_{m_j} = f^j_{m_j}\left(\sum_{n_j=1}^{N_j} a^j{}_{m_j n_j} x^j_{n_j} + b^j_{m_j}\right), \quad m_j = \overline{1,M_j}, \quad j = \overline{1,J}, \tag{3}$$

где J – число слоев, j – номер слоя.

Остальные символы имеют то же значение, что и в соотношении (2), и относятся к j-му слою.

Между слоями могут быть установлены *связи* различного вида. В общем случае выход слоя с номером j подается на вход слоя с номером $j + s$. (Вход нейронной сети можно рассматривать как выход "нулевого слоя".)

Связи будем называть:

– *последовательными*, если $s = 1$;

– *перекрестными*, если $s > 1$;

– *обратными*, если $s < 1$.

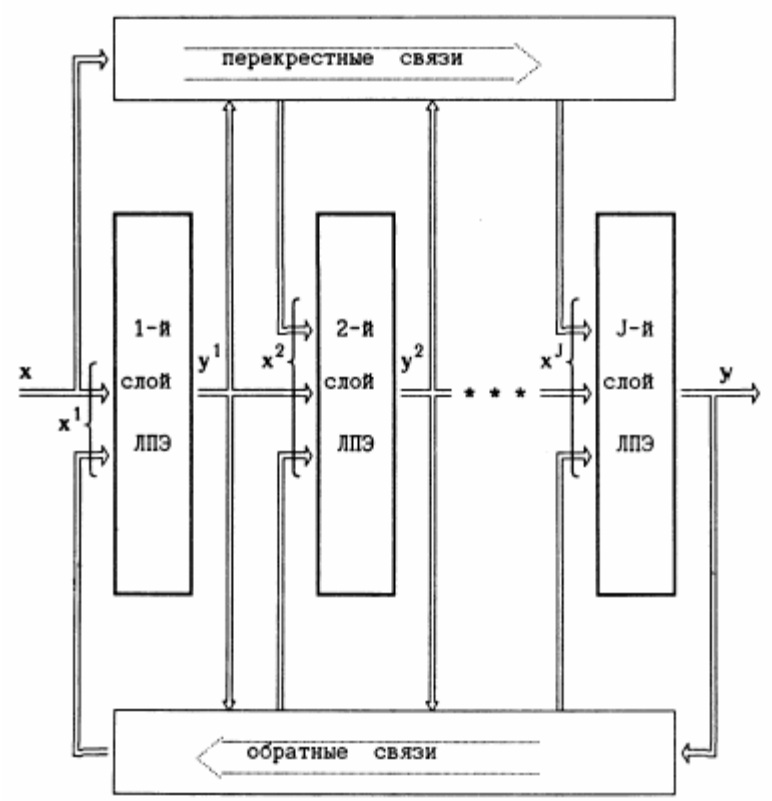


Рис. 2. Многослойная нейронная сеть

Нейронной сетью (НС) назовем структуру (см. рис. 2), состоящую из связанных между собой слоев ЛПЭ. НС может быть реализована с помощью различных технических устройств — цифровых, аналоговых, оптических — или смоделирована (эмулирована) программно.
Если ЛПЭ каждого слоя нейронной сети имеет единую функцию активации

$$f^j_{m_j}(\cdot) = f^j(\cdot) \quad \forall j,$$

то такую НС будем назьюать *однородной.*
Шагом работы НС будем называть промежуток времени между подачей входного сигнала на 1-й слой и получением выходного сигнала последнего слоя.

3. Нейронные алгоритмы решения задач.

Нейронным алгоритмом (НА) будем называть вычислительную процедуру, основная часть которой может быть реализована на нейронной сети.

П о с т р о е н и е н е й р о н н о г о а л г о р и т м а .

Пусть имеется P — формальная постановка задачи. Она включает множества исходных данных D и множество объектов, подлежащих определению, R.
Основой разработки нейронного алгоритма решения задачи является системный подход, при котором процесс решения задачи представляется как функционирование во времени некоторой динамической системы, на вход которой подается множество D, а на выходе снимается множество R (объекты, подлежащие определению и получившие свои значения) .
Для построения динамической системы, решающей задачу, необходимо определить:

- объект, выступающий в роли входного сигнала нейросети (это может быть элемент исходных данных, начальное значение определяемых величин и т.д.);

- объект, выступающий в роли выходного сигнала нейросети (это может быть само решение или некоторая его характеристика) ;
- желаемый (требуемый) выходной сигнал нейросети;
- структуру нейросети:
  а) число слоев,
  б) связи между слоями,
  в) объекты, являющиеся весовыми коэффициентами;

- функцию ошибки системы, т.е. функцию, характеризующую отклонение желаемого выходного сигнала нейросети от реального выходного сигнала;
- критерий качества системы и функционал ее оптимизации, зависящий от ошибки;
- значения весовых коэффициентов — в зависимости от задачи это возможно сделать различными способами:
  а) аналитически непосредственно из постановки задачи,
  б) с помощью некоторых численных методов,
  в) применив процедуру *настройки* весовых коэффициентов нейронной сети.

3.2. Р е ш е н и е з а д а ч и на н е й р о к о м п ь ю т е р е .
Решение задачи с помощью нейронного алгоритма заключается в применении (функционировании в некотором режиме) построенной вычислительной процедуры с конкретными значениями числовых данных.

Процесс решения:
Ш а г 1 - получение конкретной структуры нейросети, соответствующей применяемому алгоритму;
Шаг 2 - нахождение значений весовых коэффициентов либо выбор их из памяти, если они были найдены ранее;
Шаг 3—генерация начальных приближений параметров, если это необходимо;
Ш а г 4 — передача всех численных значений в нейросеть и ее запуск;
Шаг 5 - функционирование сети в соответствии с режимом:
а) за один шаг или фиксированное число шагов,
б) за переменное число шагов, зависящее от требуемой точности и/или конкретных числовых значений параметров (в этом случае, происходит процесс *настройки входного сигнала)* ;
Ш а г 6 — получение решения.

При многократном использовании пп. 1 и 2 могут быть выполнены однократно.
Таким образом, *нейрокомпьютером* будем называть вычислительную систему с архитектурой, обеспечивающей выполнение шагов 1—6.

4. Примеры построения нейронных алгоритмов.

Ниже представлены нейронные алгоритмы решения некоторых задач в соответствии с определениями и общим подходом, изложенным выше.

4.1. О б р а щ е н и е числа.
Рассмотрим вычисление значения функции
$y = 1/x$, где $x > 0$, x6R.

Эта функция обладает свойствами положительности, монотонного убывания, выпуклости и симметричности относительно прямой *y = x*. Ее графиком является гипербола.
Требуется построить нейронный алгоритм, находящий с некоторой точностью по заданному положительному вещественному числу *x* обратное ему число.
Заметим, что операция деления легко сводится к операции обращения умножением результата обращения на делимое.
4.1.1. *Первый алгоритм обращения числа*.
Гиперболу *y = 1/x* аппроксимируем кусочно-линейной функцией
$1/x \approx a_i x + b_i$ при $x_i < x < x_{i+1}$, $i = \overline{0, N}$
(см. рис. 3), проходящей через точки разбиения *\(x(, l/Xf)]* , принадлежащие гиперболе.
Здесь *N* - число точек разбиения, которое для определенности будем считать нечетным; *at, bt* - коэффициенты прямой /-го участка (определяются через координаты точек разбиения);
за (х0, *1/x0)* примем точку (е,М), *М*= 1/е,
з а (Л*лг+1? *1IXN+I)* пРи м ем точку (М, б) ;
будем считать, что *xN+l/2* = 1.

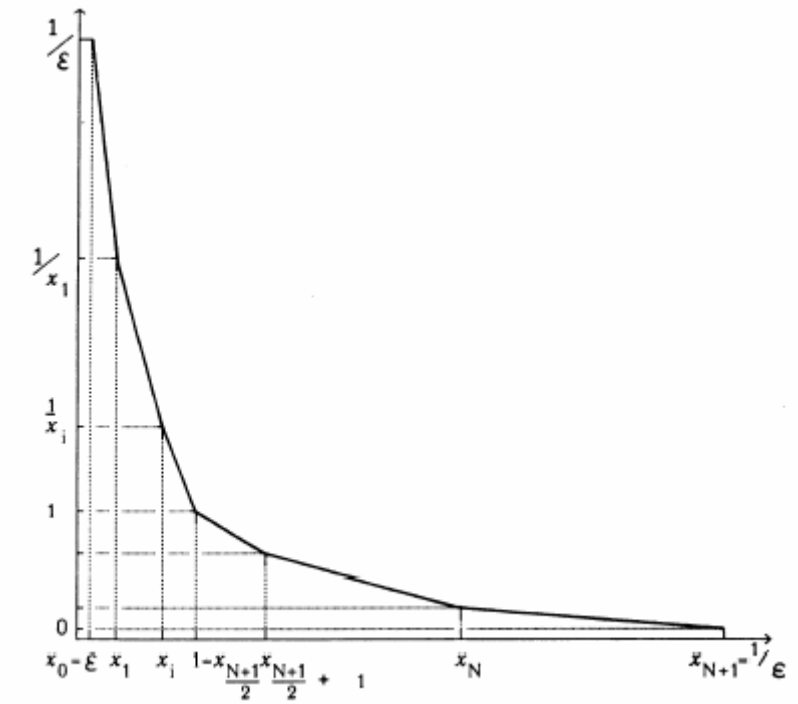


Рис. 3. Кусочно-линейная аппроксимация гиперболы $y = 1/x$

Была решена задача о нахождении разбиения, удовлетворяющего этим требованиям, и оптимального в смысле минимума функционала

$$F = \sum_i \int_{x_i}^{x_{i+1}} \left(a_i x + b_i - \frac{1}{x} \right) dx \to \min_{x_i}.$$

Ее решением является следующее разбиение:

$$x_i = \epsilon^{1-2i/(N+1)}, \quad i = \overline{0, N+1},$$

или, соответственно, $y = 1/x_i = \epsilon^{2i/(N+1)-1}$.

Приступим к построению алгоритма.
Входным сигналом нейронной сети, решающей поставленную задачу, является исходно вещественный аргумент *x*.

Выходным сигналом является результат — число y.

Желаемым выходным сигналом является значение числа, обратного входному, найденное с некоторой точностью.

Структуру нейронной сети определим следующим образом:

— первый слой вычисляет аппроксимирующее линейное преобразование

$$y_i^1 = a_i x + b_i,$$

где a_i, b_i определяются по формулам коэффициентов прямой, проходящей через две точки с координатами $(x_i, 1/x_i)$, $(x_{i+1}, 1/x_{i+1})$, и получаются в следующем виде:

$$a_i = -\frac{1}{x_i x_{i+1}}, \quad b_i = \frac{1}{x_i} + \frac{1}{x_{i+1}};$$

– второй слой определяет принадлежность выходного сигнала первого слоя к интервалу $[x_i, x_{i+1}]$:

$$y_i^2 = f(c_i x + d_i), \quad i = \overline{0, N},$$

c_i и d_i определяются по формулам для весовых коэффициентов ЛПЭ, определяющего принадлежность к интервалу (x_i, x_{i+1}):

$$c_i = \frac{1}{x_{i+1} - x_i}, \quad d_i = -\frac{x_i}{x_{i+1} - x_i}, \quad f(g) = \begin{cases} 1, & 0 < g < 1, \\ 0, & \text{иначе;} \end{cases}$$

– третий слой состоит из одного ЛПЭ:

$$y = \sum_i (y_i^1 y_i^2);$$

здесь y_i^1, y_i^2 – выходные сигналы, соответственно, 1-го и 2-го слоя.

Таким образом, гипербола оказалась представлена в виде ломаной, имеющей $N + 1$ звено. Каждый ЛПЭ в сети будет реализовывать одно звено. Число точек разбиения N является параметром сети и определяет точность аппроксимации. Для повышения точности требуется только увеличить N.

Структура полученной нейронной сети показана на рис. 4.

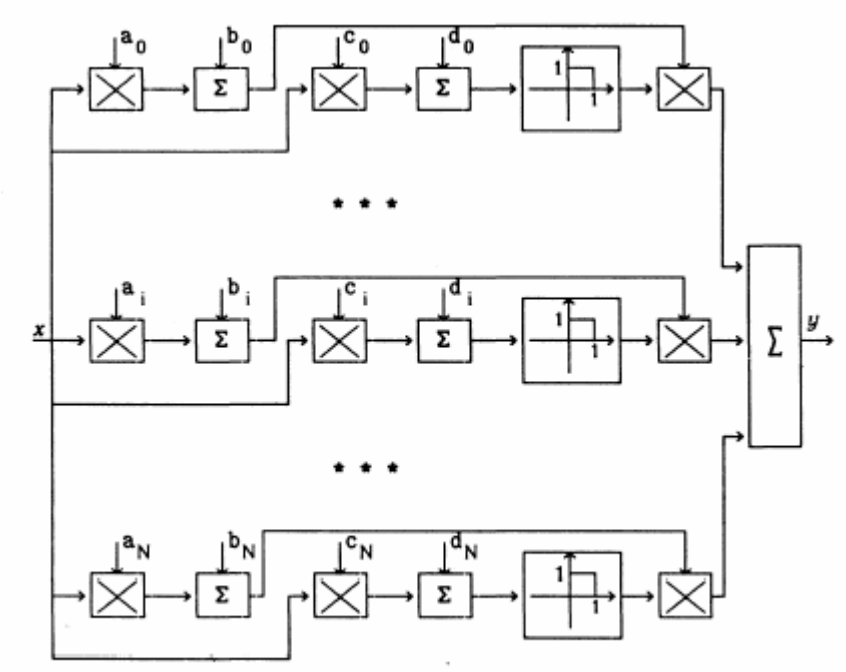


Рис. 4. Нейронная сеть для обращения числа (первый алгоритм)

СПИСОК ЛИТЕРАТУРЫ
1. *Розенбаатт Ф.* Принципы нейродинамики. - М.: Мир, 1964.
2. *Минский М., ПайпертС.* Персептроны. - М.: Мир, 1965.
3. *Нильсон* Я. Обучающиеся машины. - М.: Мир, 1967.
4. *GrossbergS.* The adaptive brain. V. 1, 2. Advances in psychology. - 1987.
5. *ДертоузосМ,* Пороговая логика. - М.: Мир, 1967.
6. *Галушкин А.И.* Синтез многослойных систем распознавания образов. - М.: Энергия, 1974,
7. А. И. Галушкин, В. А. Судариков, Е. В. Шабанов, Нейроматематика.pdf

Лекция 7. Кластеризация.

Задача кластеризации

Министерством образования и науки Российской федерации совместно с Министерством связи и массовых коммуникаций РФ реализуется с 2012 года стратегическая программа прорывных научных исследований в области информационных технологий **"Искусственные когнитивные системы".**

Под "искусственными **когнитивными системами**" понимаются технические системы, способные к

- **познанию,**
- **распознаванию образов,**
- **самостоятельному усвоению новых знаний из различных источников,**
- **продолжительному обучению,**
- **пониманию контекстуального значения,**
- **субъективной оценке получаемой информации,**
- **синтезу нового знания,**
- **мышлению,**
- **поведению**

для успешного решения существующих проблем в условиях реального мира.

Перечисленный состав задач объясняется в разделе «теория познания» науки, называющейся «Философия».

Познание начинается с созерцания, в результате которого создаются образы, которые являются элементами знания. Образовавшиеся в результате созерцания образы (множества знаний) в результате анализа разделяются на сходные группы.

Назовём образовавшиеся группы кластерами, а аналитическую операцию по разделению множества знаний на сходные группы - кластеризацией.

Изучение кластеров позволяет превратить образы в знания, сформировать их типовой образ: размеры, типовые сходные черты, отличительные черты кластеров.

Анализ типовых сходных черт и отличительных черт кластеров позволяет выявить их иерархию, выделить род, видовые отличия и взаимосвязи, определить их взаимосвязи и взаимозависимости.

Типовая структура кластеров (сходные и отличительные черты, иерархия, взаимосвязи и взаимозависимости) дают необходимую информацию для следующего этапа познания - распознавания образов.

Для нейроматематики важно, что кластеризация выполняется в условиях отсутствия каких-либо знаний об окружающем мире, неизвестном составе сформировавшихся при созерцании образов и неизвестном их количестве. Другими словами, аналитическая операция

«кластеризация» выполняется “без учителя”. В то же время, разделить сформировавшиеся образы на однотипные группы можно и под руководством учителя, но это уже не задача кластеризации. Её обычно относят к группе задач распознавания образов или автоматической классификации.

Кластеризация предназначена для разбиения совокупности объектов на однородные группы (кластеры, классы или таксоны). Если данные однородные группы представить как точки в признаковом пространстве, то задача кластеризации сводится к определению "сгущений точек".

Кластеризация является описательной процедурой, она не делает никаких статистических выводов, но дает возможность провести разведочный анализ и изучить "структуру данных".

Переводится понятие кластер (cluster) как "скопление", "гроздь". Само понятие "кластер" определено неоднозначно: в каждом исследовании свои "кластеры".

Кластер можно охарактеризовать как группу объектов, имеющих общие свойства.

Характеристиками кластера можно назвать два признака:

- внутренняя однородность (минимальная внутригрупповая дисперсия);
- внешняя изолированность.

Схематически сходство и различие задач классификации и кластеризации можно представить графически:

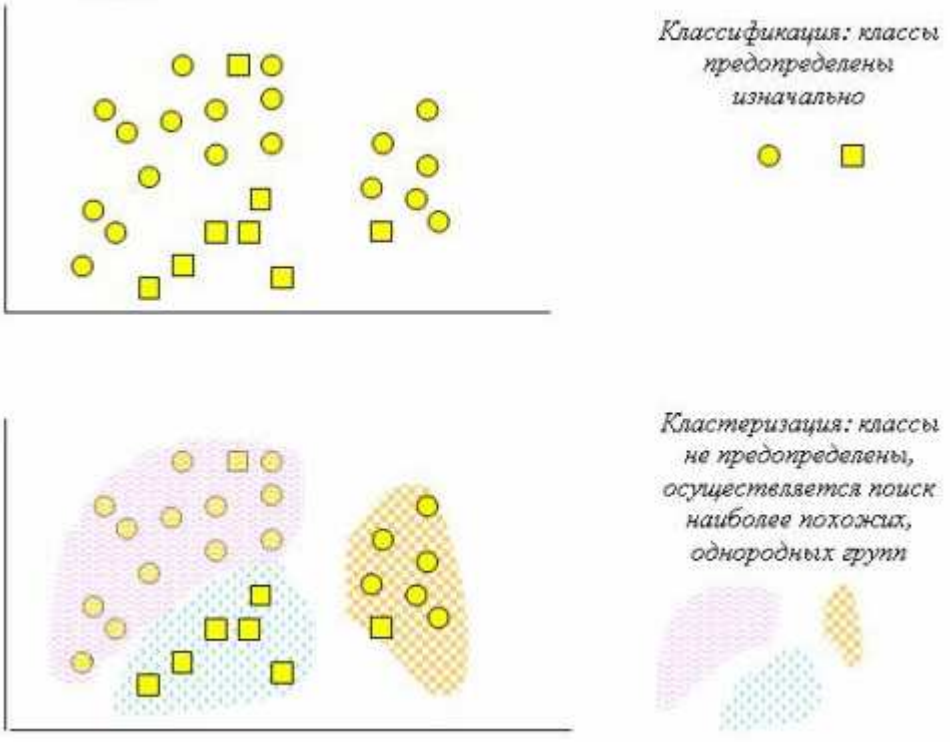


Рис. Сравнение задач классификации и кластеризации

Кластеры могут быть непересекающимися - или эксклюзивными (non-overlapping, exclusive), и пересекающимися (overlapping). Схематическое изображение непересекающихся и пересекающихся кластеров дано на рис.

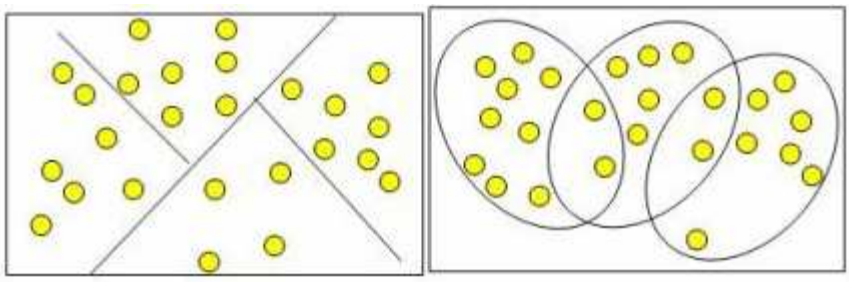


Рис. Непересекающиеся и пересекающиеся кластеры

В результате применения различных методов кластерного анализа могут быть получены кластеры различной формы.

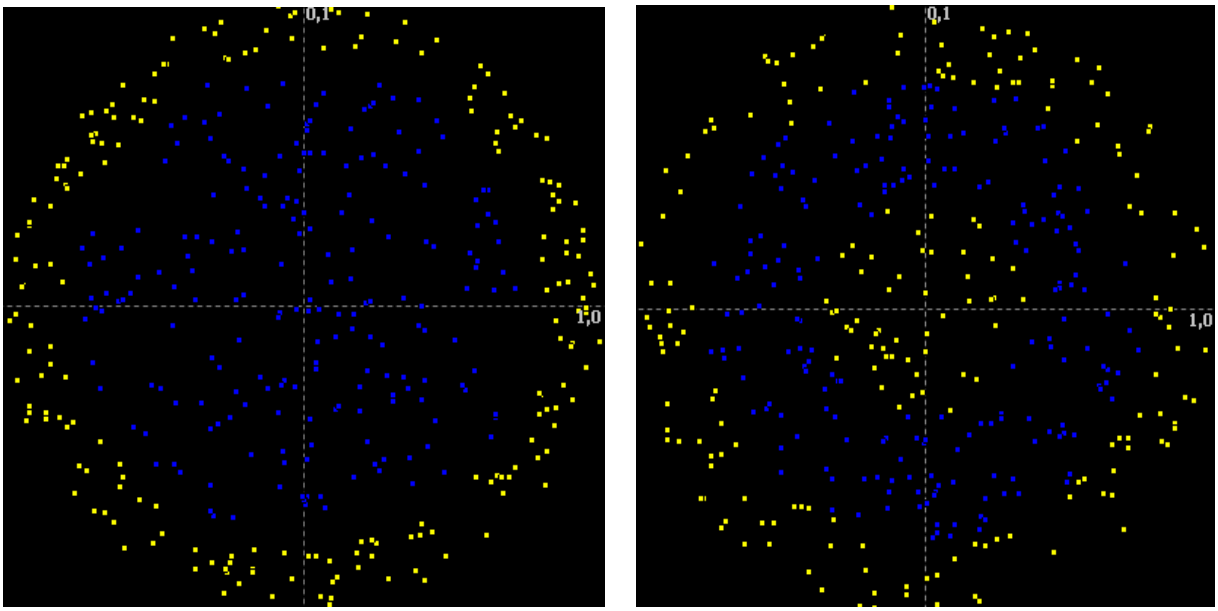
Например, возможны кластеры "цепочечного" типа, когда кластеры представлены длинными "цепочками", кластеры удлиненной формы и т.д., а некоторые методы могут создавать кластеры произвольной формы.

Кроме того, кластеры могут быть компактными, легко разделимыми, и - нечёткими, границы между которыми размыты.

Классы, возникающие при автоматической классификации, могут быть

дискретными (когда объект либо является, либо не является элементом некоторого класса. И уж если он вошел в один класс, то в других он отсутствует), или

нечеткими (когда объектам ставятся в соответствие функции принадлежности каждому классу, например - вероятность).



Различные методы могут стремиться создавать кластеры определенных размеров (например, малых или крупных) либо предполагать в наборе данных наличие кластеров различного размера.

Некоторые методы кластерного анализа особенно чувствительны к шумам или выбросам, другие - менее.

В результате применения различных методов кластеризации могут быть получены неодинаковые результаты, это нормально и является особенностью работы того или иного алгоритма. Данные особенности следует учитывать при выборе метода кластеризации.

Процесс кластеризации

Конечной целью кластеризации является получение наиболее похожих однородных групп объектов, и желательно - содержательных сведений о структуре исследуемых данных (мера расстояния, тип стандартизации переменных, количество кластеров, и т.д.).

Полученные при кластеризации результаты требуют дальнейшей интерпретации, исследования и изучения свойств и характеристик объектов для возможности точного описания сформированных кластеров.

Первым вопросом при решении всех этих задач является вопрос о том, как определить похожесть, родственность, близость образов.

Один из методов определения схожести объектов основан на том, что сходные объекты имеют похожие, (т.е. близкие по значению) характеристики. Если в системе координат на плоскость нанести точки, характеризующие эти объекты, такие точки будут находиться недалеко друг от друга. И наоборот, у сильно различающихся объектов точки будут находиться на большом расстоянии.

Расстояние между точками можно определить по их координатам, в качестве которых могут выступать свойства исследуемых объектов. Для двух объектов мерой близости может являться Эвклидово расстояние:

$$d(X,Y) = \sqrt{\sum_{i=1}^{m} (X_i - Y_i)^2}$$

где X_i – значение свойства одного объекта, Y_i – значение этого же свойства другого объекта, m – количество сопоставляемых свойств; i – порядковый номер свойства.

Условимся значения свойств каждого изучаемого объекта записывать в виде отдельной строки прямоугольной таблицы. Такая таблица фиксирует результаты наблюдения за "m" объектами в виде "объект-свойство".

Строки этой таблицы (называемые "реализации"), характеризуют объекты. Обозначим их - x(1), x(2),..., x(j),..., x(n).

Столбцы таблицы (называемые “переменные"), характеризуют свойства. Свойства объектов обозначим a(1), a(2),...,a(i),...,a(m).

Таблица "объект-свойство" размером m*n может быть использована для анализа представленных в ней наблюдений с целью обнаружения эмпирических закономерностей, т.е. устойчивых зависимостей между разными частями таблицы.

№ п/п	**Наименование объекта**	**a(1)**	**a(2)**	**…**	**a(i)**	**…**	**a(m)**
1	x(1)						
2	x(2)						
…	…						
j	x(j)						
…	…						
n	x(n)						

Допустим, у нас есть три объекта: A, B, C. Свойства этих объектов отображены в таблице:

Объект	a(1)	a(2)	a(3)	a(4)	a(5)	a(6)
A	1	2	3	4	5	6
B	9	8	7	6	5	4
C	2	3	4	5	6	7

Какие из этих объектов ближе друг к другу? Вычислим эвклидовы расстояния объектов A от B, B от C и A от C. Результат разместим в другой таблице:

Объекты	B	C
A	12	~2,45
B	x	~9,6

d(A,B) =12;
d(A,C) =2,45;
d(C,B) =9,6;

Таким образом, наименьшее расстояние – между объектами A и C. Значит, они наиболее похожи. Это видно и из простого попарного сопоставления свойств объектов.

Нужно отметить, что результаты в сильной степени зависят от того, что представляют собой свойства объектов. Например, если свойства характеризуют форму, то может существовать два очень похожих объекта различной величины. Эвклидово расстояние похожесть этих объектов не покажет. Но если свойства перед вычислением эвклидова расстояния нормировать, результат будет соответствовать действительности. Аналогичный результат может быть получен и при центрировании объектов.

Мер близости разработано очень много. Каждая мера имеет свои специфические особенности и представляет интерес только для определённых условий.

Перечислим некоторые меры близости:

1. Квадрат евклидова расстояния (Squared Euclidian distance); $d(X,Y) = \sum_{i=1}^{m} (X_i - Y_i)^2$
2. Мера близости — коэффициент корреляции $S(X,Y) = (\sum_{i=1}^{m} Z_{X_i} Z_{Y_i})/(m-1)$, где Z_{X_i} и Z_{Y_i} — компоненты стандартизованных векторов X и Y. Эту меру целесообразно использовать для выявления кластеров переменных, а не объектов.
3. Бинарное евклидово расстояние представляет собой корень из числа наблюдений, для которых, по крайней мере, один из критериев присутствует и один отсутствует.
$$dist = \sqrt{b+c}$$
4. Разность длин. Эта мера имеет минимальное значение равное 0 и не имеет верхнего предела. $$dist = \frac{(b-c)^2}{(a+b+c+d)^2}$$
5. **Дисперсия** имеет минимальное значение равное 0 и не имеет верхнего предела.
$$dist = \frac{b+c}{4(a+b+c+d)}$$
6. Форма. У этой дистанционной меры нет ни нижнего ни верхнего предела
$$dist = \frac{(a+b+c+d)(b+c)-(b-c)^2}{(a+b+c+d)^2}$$

7. Мера Ланса и Уильямса (Lance and Williams). Эта мера может принимать значения от 0 до 1. $dist = \frac{b+c}{2a+b+c}$
8. **Расстояние городских кварталов (манхэттенское расстояние),** также называемое "хэмминговым"**.** Это расстояние является просто средним от разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако отметим, что для этой меры влияние отдельных больших разностей (выбросов) уменьшается (так как они не возводятся в квадрат).
Манхэттенское расстояние вычисляется по формуле:

$$расстояние(x,y) = \Sigma_i |x_i - y_i|$$

9. **Расстояние Чебышева.** Это расстояние может оказаться полезным, когда желают определить два объекта как "различные", если они различаются по какой-либо одной координате (каким-либо одним измерением). Расстояние Чебышева вычисляется по формуле:

$$расстояние(x,y) = Максимум|x_i - y_i|$$

10. **Степенное расстояние.** Иногда желают прогрессивно увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Это может быть достигнуто с использованием *степенного расстояния*. Степенное расстояние вычисляется по формуле:

$$расстояние(x,y)=(\Sigma_i |x_i - y_i|^p)^{1/r}$$

где r и p - параметры, определяемые пользователем.
Несколько примеров вычислений могут показать, как "работает" эта мера. Параметр p ответственен за постепенное взвешивание разностей по отдельным координатам, параметр r ответственен за прогрессивное взвешивание больших расстояний между объектами. Если оба параметра - r и p, равны двум, то это расстояние совпадает с расстоянием Евклида.
11. **Процент несогласия.** Эта мера используется в тех случаях, когда данные являются категориальными.

Категориальные данные (*Categorical Data*)- это тип данных, при сравнении которых применимы только две операции: равно (=) и не равно (<>). Упорядочить значения признака нельзя. Категориальными, как правило, являются строковые и списочные данные. Пример категориальных полей: Товар, Город, Район, Профессия.

Это расстояние вычисляется по формуле:

$$расстояние(x,y)=(Количество\ x_i \neq y_i)/\ i\ .$$

Оценка качества кластеризации

Оценка качества кластеризации может быть проведена на основе следующих процедур:

- ручная проверка;
- установление контрольных точек и проверка на полученных кластерах;
- определение стабильности кластеризации путем добавления в модель новых переменных;

- создание и сравнение кластеров с использованием различных методов.

Разные методы кластеризации могут создавать разные кластеры, и это является нормальным явлением. Однако создание схожих кластеров различными методами указывает на правильность кластеризации.

Применение кластерного анализа

Кластерный анализ применяется в различных областях. Он полезен, когда нужно классифицировать большое количество информации.

Так, в медицине используется кластеризация заболеваний, лечения заболеваний или их симптомов, а также таксономия пациентов, препаратов и т.д.

В археологии устанавливаются таксономии каменных сооружений и древних объектов и т.д.

В маркетинге это может быть задача сегментации конкурентов и потребителей.

В менеджменте примером задачи кластеризации будет разбиение персонала на различные группы, классификация потребителей и поставщиков, выявление схожих производственных ситуаций, при которых возникает брак.

В медицине – классификация симптомов.

В социологии задача кластеризации - разбиение респондентов на однородные группы.

Задачи, решаемые при помощи карт Кохонена

Самоорганизующиеся карты могут использоваться для решения таких задач, как моделирование, прогнозирование, поиск закономерностей в больших массивах данных, выявление наборов независимых признаков и сжатие информации.

Наиболее распространенное применение сетей Кохонена - решение задачи классификации без учителя, т.е. кластеризации.

При такой постановке задачи нам дан набор объектов, каждому из которых сопоставлена строка таблицы (вектор значений признаков). Требуется разбить исходное множество на классы, т.е. для каждого объекта найти класс, к которому он принадлежит.

В результате получения новой информации о классах возможна коррекция существующих правил классификации объектов.

Вот два из распространенных применений карт Кохонена: разведочный анализ данных и обнаружение новых явлений.

Разведочный анализ данных. Сеть Кохонена способна распознавать кластеры в данных, а также устанавливать близость классов.

Таким образом, пользователь может улучшить свое понимание структуры данных, чтобы затем уточнить нейросетевую модель.

Если в данных распознаны классы, то их можно обозначить, после чего сеть сможет решать задачи классификации.

Сети Кохонена можно использовать и в тех задачах классификации, где классы уже заданы, - тогда преимущество будет в том, что сеть сможет выявить сходство между различными классами.

Обнаружение новых явлений. Сеть Кохонена распознает кластеры в обучающих данных и относит все данные к тем или иным кластерам.

Если после этого сеть встретится с набором данных, непохожим ни на один из известных образцов, то она не сможет классифицировать такой набор и тем самым выявит его новизну.

Кластерный анализ в маркетинговых исследованиях

В маркетинговых исследованиях кластерный анализ применяется достаточно широко. Так, одной из наиболее важных задач является анализ поведения потребителя, а именно:
группировка потребителей в однородные классы для получения максимально полного представления о поведении клиента из каждой группы и о факторах, влияющих на его поведение.

Важной задачей, которую может решить кластерный анализ, является позиционирование, т.е. определение ниши, в которой следует позиционировать новый продукт, предлагаемый на рынке.

В результате применения кластерного анализа строится карта, по которой можно определить уровень конкуренции в различных сегментах рынка и соответствующие характеристики товара для возможности попадания в этот сегмент.

С помощью анализа такой карты возможно определение новых, незанятых ниш на рынке, в которых можно предлагать существующие товары или разрабатывать новые.

Кластерный анализ также может быть удобен, например, для анализа клиентов компании. Для этого все клиенты группируются в кластеры, и для каждого кластера вырабатывается индивидуальная политика. Такой подход позволяет существенно сократить объекты анализа, и, в то же время, индивидуально подойти к каждой группе клиентов.

Методы кластеризации.

Предположим, мы имеем таблицу с характеристиками 15 объектов:

№ п/п	Наименование объекта	a(1)	a(2)	a(3)
1	x(1)	1	1	1
2	x(2)	2	2	2
3	x(3)	3	3	3
4	x(4)	4	4	4
5	x(5)	5	5	5
6	x(6)	2	1	1
7	x(7)	4	2	2
8	x(8)	5	3	3
9	x(9)	6	4	4
10	x(10)	7	5	5
11	x(11)	1	2	1
12	x(12)	2	3	2
13	x(13)	3	4	3
14	x(14)	4	5	4
15	x(15)	5	6	5

Таблица объектов и их свойств.

Одинаковых объектов в этой таблице на первый взгляд не видно. Но при ближайшем рассмотрении видно, что часть объектов похожи друг на друга.

Такая таблица не обладает высокой наглядностью. Поэтому, имеющихся у этих объектов эмпирических закономерностей, устойчивых зависимостей между разными объектами в таблице незаметно.

Задача кластеризации заключается в том, что надо сделать имеющиеся скрытые закономерности в данных выпуклыми, выделить группы «похожих» объектов, разделить всю совокупность объектов на группы родственных объектов.

Кластеризация может выполняться по-разному, в зависимости от того, **известно заранее количество кластеров, или нет**.

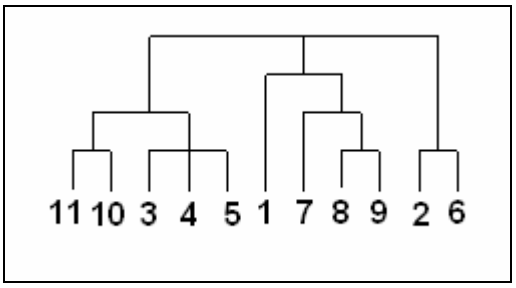
Например, при использовании метода автоматической классификации для перевода текста из графического в символьный вид, количество кластеров известно до решения задачи – оно равно количеству букв и знаков препинания, используемых в алфавите.

В других случаях количество кластеров может быть не таким очевидным, например, при распознавании речи.

В тех случаях, когда количество кластеров заранее неизвестно, классификация может быть либо **полной, либо объективной**.

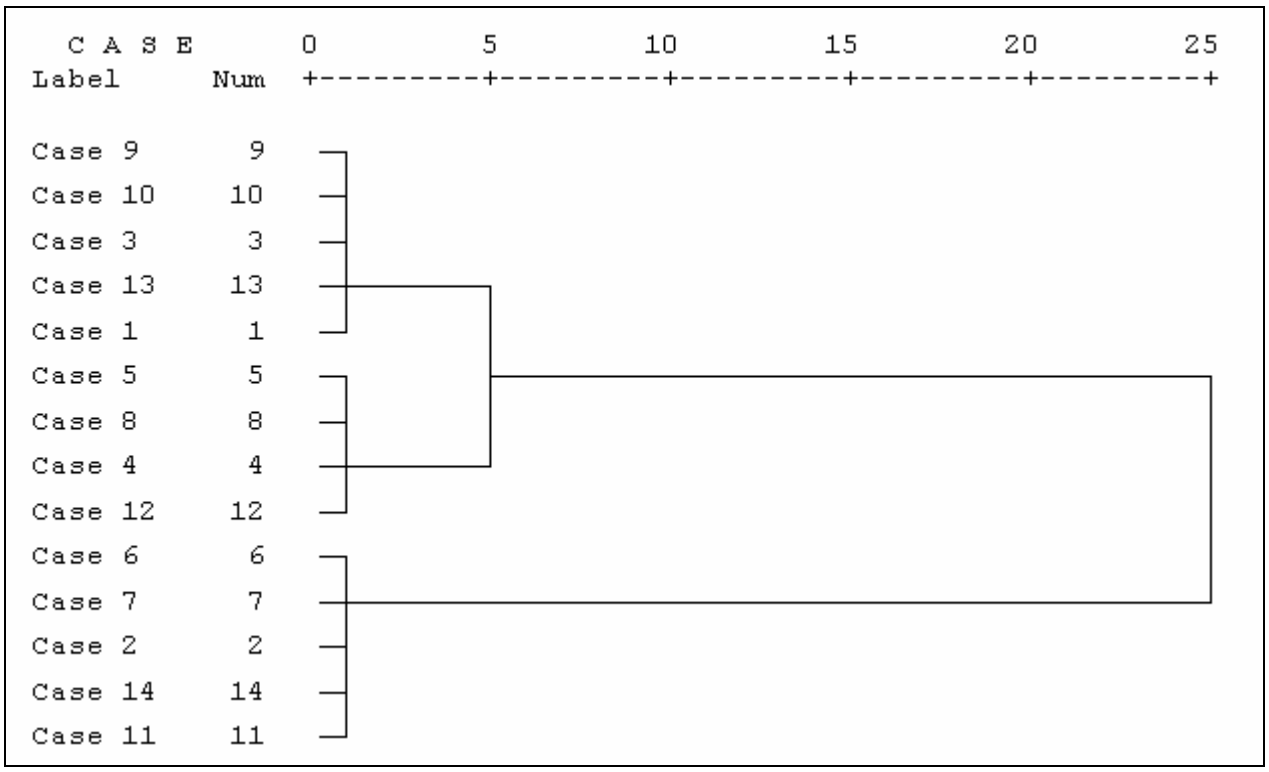
Полная классификация начинается с момента, когда каждый предъявленный объект образует свой кластер. На каждом шаге классификации наиболее близкие объекты объединяются в один кластер. Классификация заканчивается, когда все объекты попали в один кластер.

Результаты полной классификации отображаются графически в виде дендрограммы, которую трактует человек.



Пример 1. Вертикальная дендрограмма.

Это типичный пример **когнитивной графики**, так как полученные диаграммы характеризуют тяготение объектов друг к другу, силу их связи и структуру связей объектов.



Пример 2. Горизонтальная дендрограмма.

Типология объектов позволяет выделить из них сходные группы объектов.

Типология свойств (расположенных в исходной таблице по вертикали) позволяет выделить сходные свойства, которые могут дублировать друг друга, и в дальнейших исследованиях могут без особого вреда для точности не использоваться.

Это позволяет сжать описание объектов, повысить информативность используемого при анализе комплекса свойств, повысить обозримость материала и статистическую обоснованность выводов.

Традиционный способ типологизации связан с разделением множества M объектов таблицы "объект-свойство" на небольшое число групп объектов, связанных друг с другом каким-нибудь закономерным свойством. Обычно в качестве такой закономерности используется "похожесть" объектов друг на друга или на некоторый "типичный" объект. Такая группировка делается с помощью методов кластерного анализа.

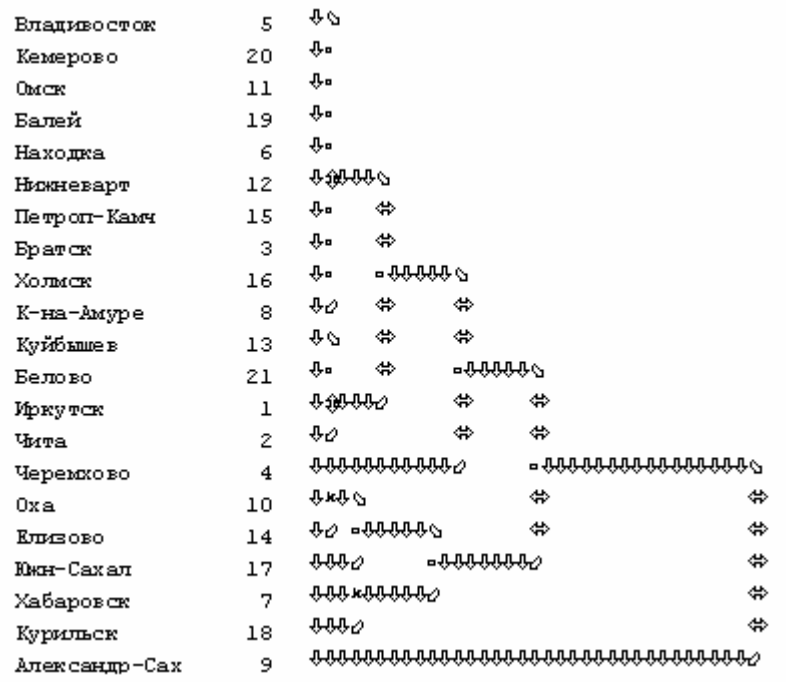
Алгоритмы кластерного анализа отличаются друг от друга применяемой мерой близости, процедурой группировки, и критериями качества.

Если свойства представить в виде координат метрического пространства, то каждый объект со своими значениями свойств будет отображаться в некоторую точку этого пространства. Два объекта с почти одинаковыми значениями свойств отобразятся в две близкие точки, а объекты с сильно отличающимися свойствами будут представлены далекими друг от друга точками.

Если имеются сгустки точек, отделенные промежутками от других сгустков, то их целесообразно выделить в отдельные структурные части множества - кластеры.

В отечественной практике исследования рынков получили распространение методы кластерного анализа, реализованные в пакетах прикладых программ BMDP (пакет программ статистического анализа биомедицинской информации), SAS (система статистического анализа), Deductor, SPSS, Статистика и др. В этих пакетах имеются

программы кластерного анализа переменных (т.е. столбцов матрицы данных), кластерного анализа реализаций (т.е. строк матрицы данных) и блочного кластерного анализа.



Пример 3. Дендрограмма классификации городов.

Объективная классификация имеет место, когда по каким-либо признакам можно определить, что на данном шаге классификацию нужно закончить. Например, гипотеза компактности образов предполагает, что родственные образы объединяются в компактные группы, для которых характерно, что расстояние между образами внутри группы меньше, чем между кластерами, объединяющими такие родственные образы. Математически условие окончания классификации в этом случае может быть сформулировано так: внутригрупповая дисперсия должна быть значительно меньше межгрупповой.

В зависимости от способа предъявления объектов, различаются **последовательная и параллельная классификации**. При последовательной классификации объекты предъявляются по одному, при параллельной – все сразу. Из-за ограничений основной памяти может быть реализована и компромиссная, параллельно – последовательная классификация (когда классификация начинается с параллельной, позволяющей сформировать начальный состав кластеров, а затем переходит к последовательной), или последовательно – параллельная, когда все объекты случайным образом (или в порядке поступления, или в ином порядке) делятся на несколько групп, которые затем последовательно предъявляются классифицирующей программе.

По существу, автоматическая классификация состоит из трех задач:

- выбор меры близости;
- выработка правила классификации;
- определение критерия окончания классификации.

В начале процесса классификации каждая исследуемая величина (переменная или реализация) рассматривается как кластер, содержащий только данную величину. На последующих шагах два наиболее близких кластера объединяются, образуя новый кластер. Процедура классификации заканчивается, когда все переменные будут объединены в один кластер, но может быть прекращена на любом шаге по указанию исследователя. После

завершения классификации печатается диаграмма (треугольная, вертикальная или горизонтальная) или таблица, объясняющая результат классификации, таблица масштабирования, и др. информация.

Процедура кластерного анализа часто ориентируется на диалоговый режим. Настройка программы позволяет выбрать для анализа только некоторые переменные или реализации.

К настоящему времени практически используются четыре группы методов распознавания образов:

- методы, основанные на измерении сходства (расстояния);
- методы, основанные на построении разделяющей образы поверхности;
- структурно – лингвистические методы распознавания;
- распознавание самоорганизующимися системами.

Если рассматриваемые образы можно однозначно охарактеризовать некоторым набором «р» признаков, каждый из которых принимает непрерывные или дискретные значения, то можно задать **р-мерное** векторное пространство, каждая координата которого представляет один признак. В этом случае образ задается точкой в многомерном (**р** – мерном) пространстве. Очевидно, похожие образы будут характеризоваться малоразличающимися между собой соответствующими координатами, и в многомерном пространстве будут располагаться близко друг к другу – в связи с чем, сходство образов можно оценивать по расстоянию между ними в многомерном пространстве. Чем больше сходство образов, тем ближе они расположены в многомерном пространстве, тем меньше расстояние между ними.

Если образы **компактны и четко разделимы**, то в многомерном пространстве они образуют хорошо различающиеся между собой множества. В каждом таком множестве можно выделить (или сформировать) типичный образ, который обладает наиболее характерными чертами данного множества. Такой типичный образ называется **эталонным**.

Задача распознавания предъявленного системе образа при таком подходе заключается в сопоставлении его с эталонами, и отнесении его к тому множеству, расстояние до которого будет минимальным.

В связи с тем, что признаки, характеризующие образ, могут быть количественными, качественными (и поддающимися ранжированию), номинальными, измерение их производится по разным типам шкал, а обработка при сопоставлении допускает использование лишь допустимых для данной шкалы математических методов.

Различие между типами признаков наиболее существенно при решении самой сложной задачи распознавания образов – выявлении закономерностей в исследуемых данных. При решении этой задачи используются разные меры близости, такие, как: L-p расстояние, Эвклидово расстояние, коэффициент корреляции, и т.д.

Для формирования кластеров используются такие правила объединения, как минимальное расстояние (единичная связь), максимальное расстояние (полная связь), среднее расстояние (средняя связь).

Метод ближнего соседа или **одиночная связь**. Здесь расстояние между двумя кластерами определяется расстоянием между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах. Этот метод позволяет выделять кластеры сколь угодно сложной формы при условии, что различные части таких кластеров соединены цепочками близких друг к другу элементов.

Метод наиболее удаленных соседей или полная связь. Здесь расстояния между кластерами определяются наибольшим расстоянием между любыми двумя объектами в различных кластерах (т.е. "наиболее удаленными соседями"). Метод хорошо использовать, когда объекты действительно происходят из различных "рощ". Если же кластеры имеют в

некотором роде удлиненную форму или их естественный тип является "цепочечным", то этот метод не следует использовать.

Метод Варда (Ward's method). В качестве расстояния между кластерами берется прирост суммы квадратов расстояний объектов до *центров кластеров*, получаемый в результате их объединения. В отличие от других методов кластерного анализа для оценки расстояний между кластерами, здесь используются методы дисперсионного анализа. На каждом шаге алгоритма объединяются такие два кластера, которые приводят к минимальному увеличению целевой функции, т.е. внутригрупповой суммы квадратов. Этот метод направлен на объединение близко расположенных кластеров и "стремится" создавать кластеры малого размера.

Помимо перечисленных существуют и другие методы объединения кластеров (Метод невзвешенного попарного среднего, Метод взвешенного попарного среднего, Невзвешенный центроидный метод, Взвешенный центроидный метод и др.).

Классификация на основе построения разделяющей поверхности.

Если различные классы образов в рассматриваемом многомерном пространстве имеют сложную форму, более эффективными являются методы распознавания образов, основанные на **построении разделяющей поверхности**. В случае двумерного пространства разделяющая поверхность представляет собой линию (рис.1.1).

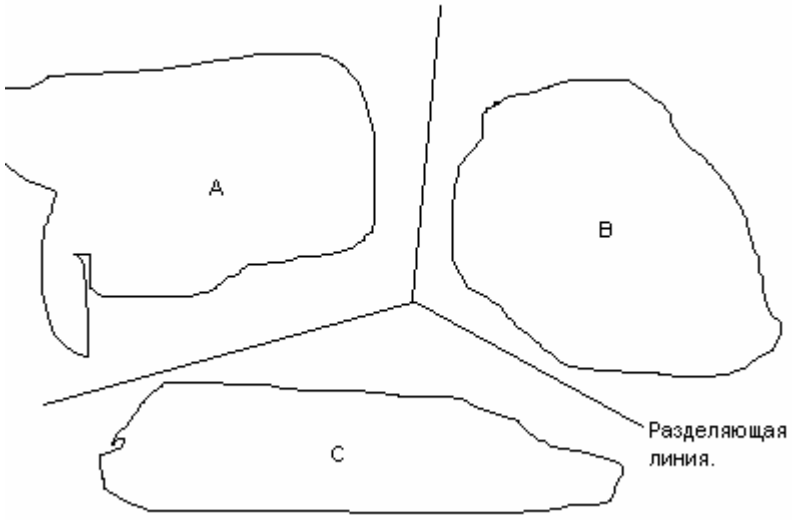


Рис. 1.1. Разделение множеств A, B, C ломаной линией.

Если данные сильно зашумлены, содержат чрезвычайно большой объем, «засоряющий» память ЭВМ, имеет смысл выделить из них для хранения лишь наиболее существенные. Это особенно важно при распознавании полутоновых изображений. Эти же проблемы возникают при выявлении закономерностей в исследуемых данных.

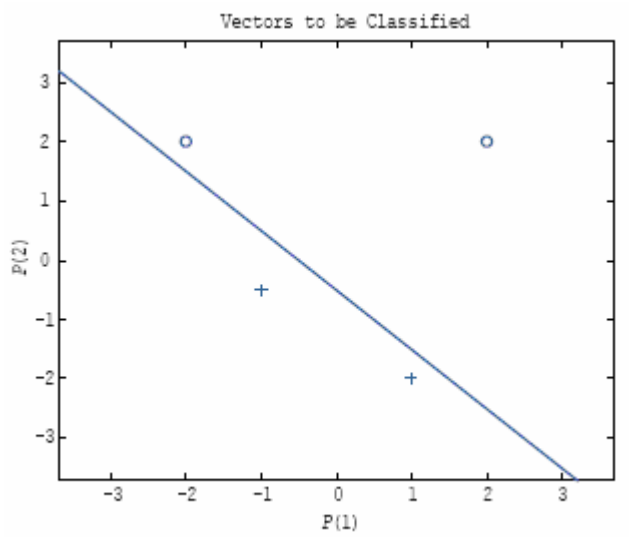


Рис. Разделяющая прямая персептрона

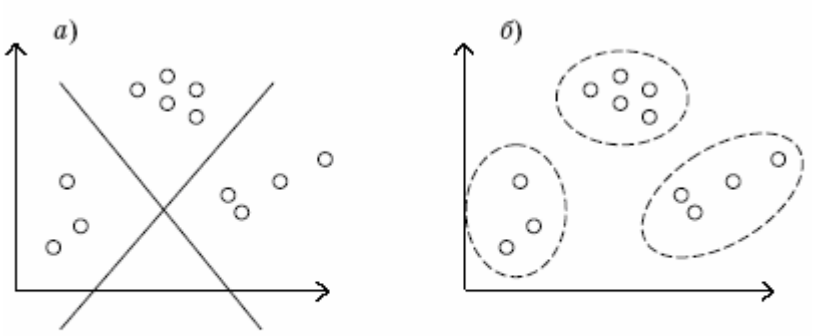


Рис. Принципы классификации с помощью многослойного персептрона (а) и RBF-сети (б)

Структурно-лингвистическая классификация.

Для решения этих проблем используются **структурно-лингвистические (синтаксические)** методы распознавания, смысл которых сводится к тому, что распознаваемый образ представляется состоящим из набора повторяющихся подобразов определенной формы, называемых «примитивами» образа. Возможный порядок следования примитивов в образе, взаимосвязи между примитивами определяются грамматикой, порождающей образ. Грамматики могут представлять собой набор правил формирования последовательностей примитивов в виде строковых данных, деревьев, сетей, графов, и т.д.

В системах структурно-лингвистического распознавания образов проводится предобработка образа (перед распознаванием). Во время предобработки образ декодируется, сглаживается, улучшается его качество. Затем образ преобразуется в строку примитивов, после чего подвергается синтаксическому анализу, в результате которого определяется, удовлетворяет или нет строка данной грамматике.

В качестве примера рассмотрим задачу по распознаванию десятичных цифр:

Распознавание десятичных цифр по табличному описанию их формы.

Табличные данные имитируют "цифры", высвечивающиеся на экране неисправного калькулятора при вводе с клавиатуры калькулятора соответствующих цифр: 0-9.

В задаче имеется 7 категориальных предикторов: Var1 - Var7 по числу линий, образующих цифру.

Уровень категориального предиктора (0 - отсутствует; 1 - присутствует) показывает, высвечивалась ли на экране соответствующая ему одна из 7 линий (3 горизонтальных и 4 вертикальных). Описание предикторных переменных: Var1 - верхняя горизонтальная, Var2 - верхняя левая вертикальная, Var3 - верхняя правая вертикальная, Var4 - средняя горизонтальная, Var5 - нижняя левая вертикальная, Var6 - нижняя правая вертикальная и Var7 - нижняя горизонтальная (см. рис.1).

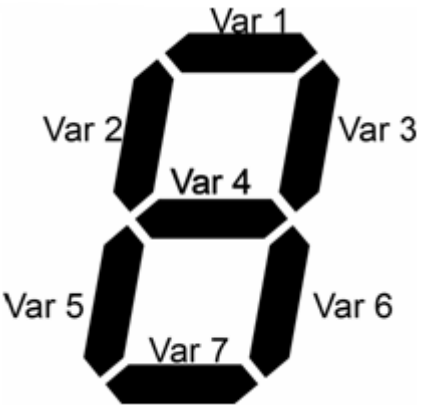


Рис.1. Названия предикторов и соответствующие им линии

Калькулятор неисправен, поэтому при нажатии какой-либо кнопки на цифровой клавиатуре на экране не всегда высвечивается правильная комбинация линий.

Структура данных

На рисунке приведены первые 10 наблюдений файла данных, которые могут отображаться не словами, а цифрами. Таблицу необходимо дополнить описанием форм слегка искажённых цифр.

Данные: Digit.sta* (8v * 500c)

Example data for pattern recognition

	1 DIGIT	2 VAR1	3 VAR2	4 VAR3	5 VAR4	6 VAR5	7 VAR6	8 VAR7
1	seven	ONE	ZERO	ONE	ZERO	ZERO	ONE	ZERO
2	one	ZERO	ZERO	ONE	ZERO	ZERO	ONE	ZERO
3	four	ZERO	ONE	ONE	ONE	ZERO	ONE	ZERO
4	two	ONE	ONE	ONE	ONE	ONE	ZERO	ZERO
5	eight	ZERO	ONE	ONE	ONE	ONE	ONE	ONE
6	one	ZERO	ZERO	ONE	ZERO	ZERO	ONE	ZERO
7	five	ONE	ONE	ZERO	ONE	ZERO	ONE	ONE
8	six	ONE	ZERO	ZERO	ONE	ONE	ONE	ONE
9	two	ONE	ZERO	ONE	ONE	ONE	ZERO	ONE
10	eight	ONE	ONE	ONE	ONE	ZERO	ONE	ONE

Рис.2. Фрагмент исходного файла данных

Результаты распознавания записаны в первой переменной (DIGIT).

В переменные VAR1-VAR7 записаны уровни независимых категориальных предикторов.

Построение модели

В данном примере необходимо построить модель распознавания цифр, выдаваемых реальным калькулятором.

Разумно сформулировать следующие требования к нейронной сети:

1) сеть должна обладать возможностью экстраполировать за область обучающих данных (т.е. давать правильный прогноз при комбинациях предикторов, которые сильно отличаются от обучающего множества),

2) требовать небольшого времени для прогноза (это диктуется требованиями практического применения).

Указанным условиям соответствует архитектура многослойного персептрона. Число элементов на скрытом слое выберем равным 5. Классификация проводится в 10 классов.

Архитектуру построенной сети удобно представить в графическом виде:

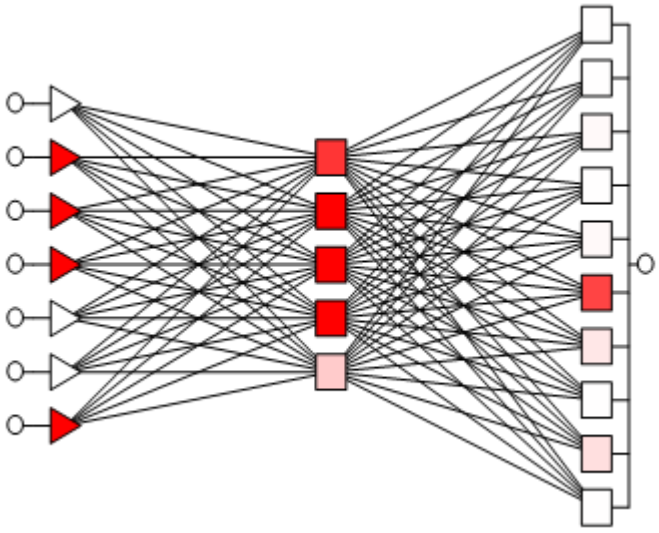


Рис. Архитектура построенной сети

Похожие образы цифр должны иметь и похожие строковые описания. Полный набор возможных строк (т.е. полный набор всех возможных образов) называется языком, а правила порождения каждого образа – соответствующей грамматикой. Поскольку различные образы должны порождаться разными грамматиками, если в результате синтаксического анализа выясняется, что строка удовлетворяет данной грамматике, это эквивалентно опознанию образа.

Приложение 1. Российский проект "Искусственные когнитивные системы (ИКС)"

образованный Министерством образования и науки совместно с Министерством связи и массовых коммуникаций Российской Федерации в 2012г. реализуется с помощью

стратегической программы прорывных исследований в области информационных технологий "Искусственные когнитивные системы"

Под "искусственными **когнитивными системами**" понимаются технические системы, способные к

- **познанию,**
- **распознаванию образов,**
- **(самостоятельному) усвоению новых знаний из различных источников,**
- **продолжительному обучению,**
- **пониманию контекстуального значения,**
- **субъективной оценке получаемой информации,**
- **синтезу нового знания,**
- **мышлению,**
- **поведению**

для успешного решения существующих проблем в условиях реального мира.

Познание начинается с созерцания, в результате которого создаются образы, которые являются элементами знания. В основе познания лежит кластеризация, т. е. разделение множества знаний на сходные группы.
Назовём образовавшиеся группы кластерами.
Изучение кластеров позволяет превратить образы в знания, сформировать их типовой образ: размеры, типовые сходные черты, отличительные черты кластеров.
Анализ типовых сходных черт и отличительных черт кластеров позволяет выявить их иерархию (род и видовые отличия) и взаимосвязи, определить их взаимозависимость.
Типовая структура кластеров (сходные и отличительные черты, иерархия, взаимосвязи и взаимозависимости) дают необходимую информацию для распознавания образов.

В основе российского проекта "Искусственные когнитивные системы" лежат разработанные компанией ТАСО

- технологии кортикоморфных нейросетей,
- технологии кибергеномики
- программные средства построения сверхбольших нейрогенетических сетей,
- технологии создания твердотельных мемристорных микросхем.

Они представлены, в частности, в опубликованных работах (2008) и на сайте taso.pro.

На втором этапе проекта (2017-2018 гг.) создается базовая программно-реализованная искусственная когнитивная система.

Для этого к программно-реализованной искусственной коре добавляются модели

1. сетчатки,
2. наружного коленчатого тела,
3. зрительная подсистема, т.е. зона V1-MT коры (стриарной коры),
4. слуховая подсистема,
5. подсистема формирования пространственно-временного континуума личности – модель энторинальной коры, субикулума, гиппокампа и зубчатой извилины,
6. подсистема субкортикального управления моторными движениями (подсистема двигательного обучения),
7. модель мозжечка,

8. модуль произвольного и непроизвольного внимания,
9. подсистема многоактного анализа,
10. система фильтрации существенного,
11. подсистема целенаправленного поведения,
12. модуль искусственной эмоциональности и субъективности.

И на первом, и на втором этапах осуществляется разработка нейроморфных электронных устройств на основе твердотельных неорганических полупроводниковых мемристорных обучаемых наноматериалов (мемристоры на основе тонких пленок диоксида титана были получены в ТюмГУ в 2012 г.).

Решение всех этих задач представляется возможным с использованием уникальных технологий, разработанных компанией ТАСО. Прежде всего –

- биоморфных моделей нейронов с многофазной консолидацией следа памяти,
- моделей кортикальных колонок,
- кортикоморфных (подобных коре головного мозга) нейронных сетей и
- кибергеномики - технологии управления ростом и развитием сверхбольших искусственных нейросетей.

Важное значение имеет то, что ТАСО создан программный комплекс "ТАСО-Нейроконструктор" для разработки нейронных сетей.

Приложение 2. Терминология теории познания.

Познание – это активное, целенаправленное отражение действительности в сознании человека; процесс проникновения человека в **сущность** действительности (вещей, явлений), в их закономерные связи и отношения, т. е. установление:
а) существенных;
б) устойчивых;
в) повторяемых связей и отношений, т. е. законов.
Теория познания **(гносеология) –** от греч. «gnosis» - познание и «logos» - понятие, учение, т. е. **учение о познании.**
Термин «гносеология» введен в философию шотландским философом Дж. Феррером в 1854 г., хотя само учение о познании разрабатывалось уже в античной философии (Гераклит, Платон, Аристотель).
В западной философии распространен термин «**эпистемология**» - от греч. «episteme» - знание. Эта часть философии исследует общие черты процесса **научного** познания и **знания,** как его результат.
Проблемы, которые исследует гносеология.
1. Природа познания (что, как и зачем познает человек).
2. Возможности и границы познания (до каких пределов познаваем мир).
3. Отношения знания и реальности (соответствуют ли наши знания действительности).
4. Условия действительности знания, критерии его истинности.
5. Формы и уровни познания и т. д.
Таким образом, **гносеология –** это раздел философии, где изучается: как мы получаем знание о разных предметах и явления действительности, каковы основания и границы нашего познания, насколько достоверно человеческое знание.
ЧЕЛОВЕК - это **мыслящая, познавательная СИСТЕМА** (идеальное действие).
Знания имеют различные уровни:
понятия языка **(ОСНОВА ВСЕХ ОСТАЛЬНЫХ ЗНАНИЙ)**

суждения
умозаключения
законы
теории
науки

http://poisk-istini.com/literatura/filosofiya-konspekty-lekciy-bushuev/teoriya-poznaniya-kak-filosofskaya-disciplina

Проблема понимания.

Настоящее понимание (а не его иллюзия) связано со смысловой обработкой текста, с обработкой знаний, а не данных. Можно знать наизусть текст “Евгения Онегина”, но не понимать его. Понимание проявляется тогда, когда текст связывается с другими знаниями человека. Например, чтобы ответить на вопрос “Какого числа состоялась дуэль Ленского с Онегиным?” (прямого указания в тексте на это нет) необходимо обратить внимание, что они поссорились на именинах Татьяны Лариной. А именины Татьяны по христианскому календарю отмечаются всегда в один и тот же день - 25 января.

Д.А.Поспелов сформулировал 5 уровней понимания:

1) прямой анализ текста, без привлечения каких - либо дополнительных знаний;

2) привлечение информации из памяти - чаще всего “бытовых знаний” о временной, пространственной и причинно-следственной структуре анализируемого текста;

3) использование всей информации по теме, имеющейся в памяти, полученной при прямом анализе текста, включая знания, отраженные на втором уровне понимания;

4) привлечение ассоциаций и аналогий - тех фрагментов памяти, которые напрямую могут быть и не связаны с анализируемым текстом, но “близки” ему;

5) из текста извлекается его прагматическое содержание за счет сопоставления выясненного на предыдущих четырех уровнях смысла со своими целевыми установками, со своей системой ценностей, со своими возможностями, ресурсами. При этом учитывается прогноз развития событий, производится планирование своих действий для достижения основной цели.

Смысл текста не является совокупностью смыслов образующих его слов и предложений.

Понимание смысла связано с выполнением умозаключений, с использованием интеллектуальных навыков, включающих в себя такие, как:

- сопоставление сложных объектов и оценку их сходства;
- выделение типового объекта из группы однородных;
- поиск типичных черт, существенных признаков;
- формирование описания типового объекта, выделение его отличительных черт;
- определение понятий (дефиниции);
- выявление причинно-следственных связей;
- интерпретация связей и свойств исследуемых объектов;
- генерация гипотез;
- выявление закономерностей;
- самообучение, адаптация;
- умение делать традуктивные, индуктивные, дедуктивные выводы;
- ...

Даже на уровне прямого анализа текста это не простая задача: каждое предложение может нести в себе лингвистически невыраженный подтекст, выявление которого является

задачей пресуппозиций. Очень ярко это показано в [ ], где приводятся следующие пресуппозиции для фразы ”Врач бегло говорила по-немецки”:

- врачем является женщина;
- немецкий для врача - не родной язык;
- отмечается степень владения языком, а не процесс говорения;
- дается оценка культурного уровня врача;
- умение “говорить по-немецки” относится к прошлому;
- имеется в виду определенный человек.

Лекция 8. Классификация на основе самоорганизующихся систем.

Одновременно с разработкой методов распознавания, реализуемых на универсальных ЭВМ фон-Неймановской архитектуры, ведется поиск методов распознавания, основанных на иных принципах. Наиболее интересные результаты в этом направлении научных исследований получены при попытках создания **распределенных самоорганизующихся систем**, например, перцептрона Розенблатта, пандемониума Селфриджа, нейронных ЭВМ (сокращенно – нейроЭВМ), моделирующих работу нервных сетей живых организмов. На их основе в настоящее время разработаны и практически используются **параллельные, волновые, матричные**, и др. системы распознавания.

Нейроны типа WTA

Нейроны типа WTA (Winner Takes All — "Победитель получает все") имеют входной модуль в виде адаптивного сумматора. Выходной сигнал i -го сумматора определяется по формуле

$$u_i = \sum_{j=0}^{N} w_{ij} x_j.$$

По результатам сравнения сигналов $u_i, i = 1, 2, \ldots, N$ отдельных нейронов победителем признается нейрон, у которого u_i оказался наибольшим. Нейрон-победитель вырабатывает на своем выходе состояние 1, а остальные (проигравшие) нейроны переходят в состояние 0.

Для обучения нейронов WTA учитель не требуется. На начальном этапе случайным образом выбираются весовые коэффициенты w_{ij} каждого нейрона, нормализуемые относительно 1 по формуле

$$w_{ij} \leftarrow w_{ij} / (w_{i1}^2 + w_{i2}^2 + \ldots + w_{iN}^2)^{1/2}.$$

После подачи входного вектора X, компоненты которого нормализованы по формуле

$$x_i \leftarrow x_i / (x_1^2 + x_2^2 + \ldots + x_N^2)^{1/2},$$

определяется победитель этапа. Победитель переходит в состояние 1, что позволяет произвести уточнение весов его входных линий w_{ij} по правилу

$w_{ij}(t+1) = w_{ij}(t) + \alpha[x - w_{ij}(t)]$.

Проигравшие нейроны формируют на своих выходах состояние 0, что блокирует процесс уточнения их весовых коэффициентов.

Выходной сигнал i-го нейрона может быть описан векторным отношением

$$u_i = w_i^T x = ||w_i|| \cdot ||x|| \cos\varphi_i.$$

Поскольку $||w_i|| = ||x|| = 1$, значение u_i определяется углом между векторами x и w_i. $u_i = \cos\varphi_i$. Поэтому победителем оказывается нейрон, вектор весов которого оказывается наиболее близким текущему обучающему вектору x. В результате победы нейрона уточняются его весовые коэффициенты, значения которых приближаются к значениям текущего обучающего вектора x.

Следствием конкуренции нейронов становится самоорганизация процесса обучения. Нейроны уточняют свои веса таким образом, что при предъявлении группы близких по значениям входных векторов победителем всегда оказывается один и тот же нейрон. Системы такого типа чаще всего применяются для классификации векторов.

Кроме приведенных в нейроматематике известны и другие типы нейронов:

- Нейроны с квадратичным сумматором,
- Сигма-Пи нейроны,
- Стохастические модели нейрона,
- Кубические модели нейронов,
- Нейрон МакКаллока-Питтса.
- Сигмоидальный нейрон.
- Нейрон типа "адалайн"
- Паде-нейрон
- Модель нейрона Хебба
- и др.

Самоорганизующиеся карты (Self-Organizing Maps, SOM)

Сети, называемые картами Кохонена, - это одна из разновидностей нейронных сетей, однако они принципиально отличаются от рассмотренных выше, поскольку используют неконтролируемое обучение.

При таком обучении обучающее множество состоит лишь из значений входных переменных, в процессе обучения нет сравнивания выходов нейронов с эталонными значениями. Можно сказать, что такая сеть учится понимать структуру данных.

Идея сети Кохонена принадлежит финскому ученому Тойво Кохонену (1982 год). Основной принцип работы сетей - введение в правило обучения нейрона информации относительно его расположения.

В основе идеи сети Кохонена лежит аналогия со свойствами человеческого мозга. Кора головного мозга человека представляет собой плоский лист и свернута складками. Таким образом, можно сказать, что она обладает определенными топологическими

свойствами (участки, ответственные за близкие части тела, примыкают друг к другу и все изображение человеческого тела отображается на эту двумерную поверхность).

Сеть Кохонена, в отличие от *многослойной нейронной сети*, очень проста; она представляет собой два слоя: входной и выходной.

Ее также называют *самоорганизующей картой*. Элементы карты располагаются в некотором пространстве, как правило, двумерном.

Сеть Кохонена изображена на рис.

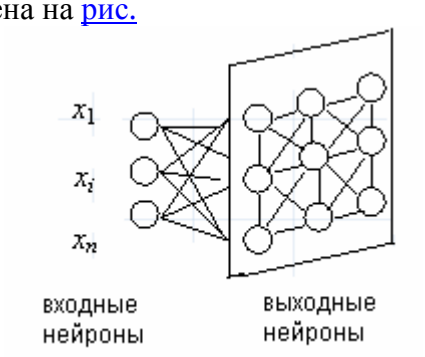


Рис. Сеть Кохонена

Сеть Кохонена обучается методом последовательных приближений. В процессе обучения таких сетей на входы подаются данные, но сеть при этом подстраивается не под эталонное значение выхода, а под закономерности во входных данных. Начинается обучение с выбранного случайным образом выходного расположения центров.

В процессе последовательной подачи на вход сети обучающих примеров определяется наиболее схожий нейрон (тот, у которого скалярное произведение весов и поданного на вход вектора минимально). Этот нейрон объявляется победителем и является центром при подстройке весов у соседних нейронов. Такое правило обучения предполагает "соревновательное" обучение с учетом расстояния нейронов от "нейрона-победителя".

Обучение при этом заключается не в минимизации ошибки, а в подстройке весов (*внутренних параметров* нейронной сети) для наибольшего совпадения с входными данными.

Основной *итерационный алгоритм* Кохонена последовательно проходит ряд эпох, на каждой из которых обрабатывается один пример из обучающей выборки. Входные сигналы последовательно предъявляются сети, при этом желаемые выходные сигналы не определяются. После предъявления достаточного числа входных векторов синаптические веса сети становятся способны определить кластеры. Веса организуются так, что топологически близкие узлы чувствительны к похожим входным сигналам.

В результате работы алгоритма *центр кластера* устанавливается в определенной позиции, удовлетворительным образом кластеризующей примеры, для которых данный нейрон является "победителем". В результате обучения сети необходимо определить меру соседства нейронов, т.е. *окрестность* нейрона-победителя.

Окрестность представляет собой несколько нейронов, которые окружают нейрон-победитель.

Сначала к *окрестности* принадлежит большое число нейронов, далее ее размер постепенно уменьшается. Сеть формирует топологическую структуру, в которой похожие примеры образуют группы примеров, близко находящиеся на топологической карте.

Полученную карту можно использовать как средство визуализации при анализе данных. В результате обучения карта Кохонена классифицирует входные примеры на

кластеры (группы схожих примеров) и визуально отображает многомерные входные данные на плоскости нейронов.

Уникальность метода *самоорганизующихся карт* состоит в преобразовании n-мерного пространства в двухмерное. Применение двухмерных сеток связано с тем, что существует проблема отображения пространственных структур большей размерности.

Имея такое представление данных, можно визуально определить наличие или отсутствие взаимосвязи во входных данных.

Нейроны карты Кохонена располагают в виде двухмерной матрицы, раскрашивают эту матрицу в зависимости от анализируемых параметров нейронов.

На рис. приведен пример карты Кохонена

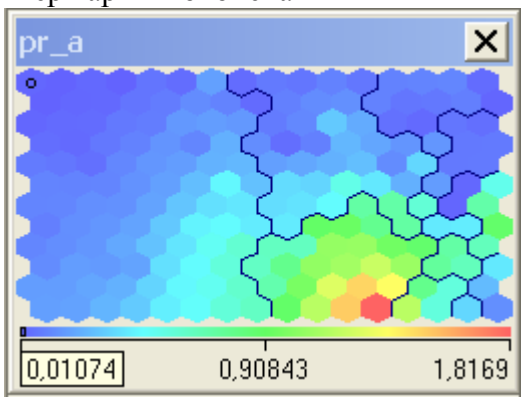


Рис. Пример карты Кохонена

Что же означает ее раскраска? На следующем рис. приведена раскраска карты, а точнее, ее i-го признака (показателя pr_a), в трехмерном представлении. Как мы видим, темно-синие участки на карте соответствуют наименьшим значениям показателя, красные - самым высоким.

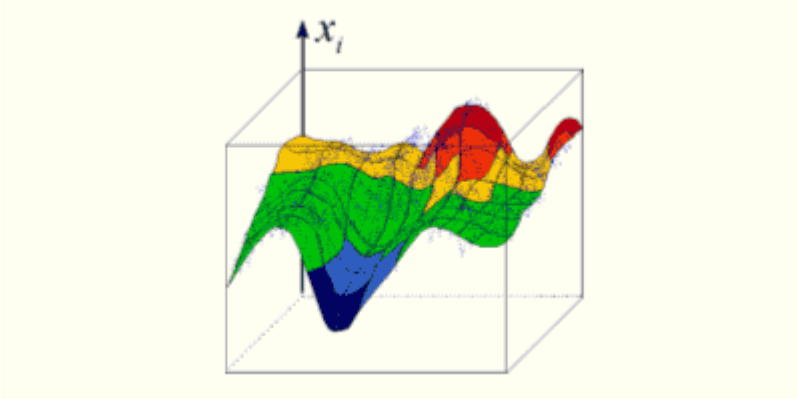


Рис. Раскраска i-го признака в трехмерном пространстве

Теперь, возвращаясь к предыдущему рисунку, мы можем сказать, какие объекты имеют наибольшие значения рассматриваемого показателя (группа объектов, обозначенная красным цветом), а какие - наименьшие значения (группа объектов, обозначенная синим цветом).

Таким образом, карты Кохонена (как и географические карты) можно отображать:

- в двухмерном виде, тогда карта раскрашивается в соответствии с уровнем выхода нейрона;
- в трехмерном виде.

В результате работы алгоритма получаем такие карты:

- *карта входов нейронов* ;
- *карта выходов нейронов* ;
- специальные карты.

Координаты каждой карты определяют положение одного нейрона. Так, координаты [15:30] определяют нейрон, который находится на пересечении 15-го столбца с 30-м рядом в матрице нейронов. Рассмотрим, что же представляют собой эти карты.

Карта входов нейронов.

Веса нейронов подстраиваются под значения входных переменных и отображают их внутреннюю структуру. Для каждого входа рисуется своя карта, раскрашенная в соответствии со значением конкретного веса нейрона.

При анализе данных используют несколько *карт входов*.

На одной из карт выделяют область определенного цвета - это означает, что соответствующие входные примеры имеют приблизительно одинаковое значение соответствующего входа. Цветовое распределение нейронов из этой области анализируется на других картах для определения схожих или отличительных характеристик. Пример рассмотренных *карт входов* будет приведен ниже.

Карта выходов нейронов.

На *карту выходов нейронов* проецируется взаимное расположение исследуемых входных данных. Нейроны с одинаковыми значениями выходов образуют кластеры - замкнутые области на карте, которые включают нейроны с одинаковыми значениями выходов.

Специальные карты. Это карта кластеров, матрица расстояний, матрица плотности попадания и другие карты, которые характеризуют кластеры, полученные в результате обучения сети Кохонена.

Важно понимать, что между всеми рассмотренными картами существует взаимосвязь - все они являются разными раскрасками одних и тех же нейронов. Каждый пример из обучающей выборки имеет одно и то же расположение на всех картах.

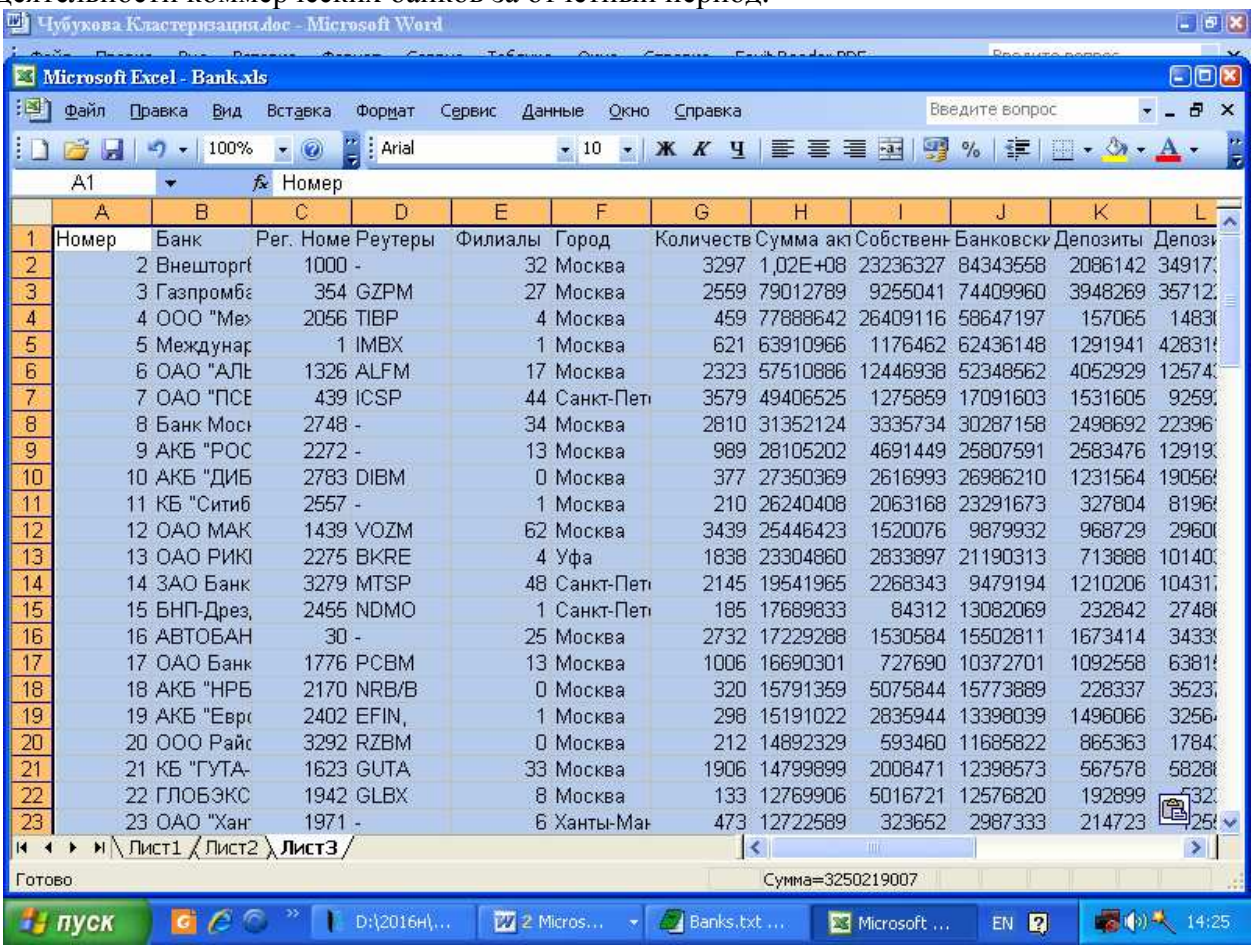
Пример решения задачи в пакете Deductor

Программное обеспечение, позволяющее работать с картами Кохонена, сейчас представлено множеством инструментов. Это могут быть как инструменты, включающие только реализацию метода *самоорганизующихся карт*, так и нейропакеты с целым набором структур нейронных сетей, среди которых - и карты Кохонена; также данный метод реализован в некоторых универсальных инструментах анализа данных.

К инструментарию, включающему реализацию метода карт Кохонена, относятся SoMine, Statistica, NeuroShell, NeuroScalp, Deductor, Matlab, MemBrain и множество других. Для решения задачи будем использовать аналитический пакет Deductor, Matlab и MemBrain.

Пусть имеется база данных коммерческих банков с показателями деятельности за текущий период. Необходимо провести их кластеризацию, т.е. выделить однородные группы банков на основе показателей из базы данных, всего показателей - 21.

Исходная таблица находится в файле "*banks*.xls". Она содержит показатели деятельности коммерческих банков за отчетный период.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Номер	Банк	Рег. Номе	Реутеры	Филиалы	Город	Количеств	Сумма акт	Собственн	Банковски	Депозиты	Депози
2	2	Внешторгf	1000	-	32	Москва	3297	1,02E+08	23236327	84343558	2086142	34917
3	3	Газпромба	354	GZPM	27	Москва	2559	79012789	9255041	74409960	3948269	35712
4	4	ООО "Мех	2056	TIBP	4	Москва	459	77888642	26409116	58647197	157065	1483
5	5	Междунар	1	IMBX	1	Москва	621	63910966	1176462	62436148	1291941	42831
6	6	ОАО "АЛЬ	1326	ALFM	17	Москва	2323	57510886	12446938	52348562	4052929	12574
7	7	ОАО "ПСБ	439	ICSP	44	Санкт-Пет	3579	49406525	1275859	17091603	1531605	9259
8	8	Банк Моск	2748	-	34	Москва	2810	31352124	3335734	30287158	2498692	22396
9	9	АКБ "РОС	2272	-	13	Москва	989	28105202	4691449	25807591	2583476	12919
10	10	АКБ "ДИБ	2783	DIBM	0	Москва	377	27350369	2616993	26986210	1231564	19056
11	11	КБ "Ситиб	2557	-	1	Москва	210	26240408	2063168	23291673	327804	8196
12	12	ОАО МАК	1439	VOZM	62	Москва	3439	25446423	1520076	9879932	968729	2960
13	13	ОАО РИКІ	2275	BKRE	4	Уфа	1838	23304860	2833897	21190313	713888	10140
14	14	ЗАО Банк	3279	MTSP	48	Санкт-Пет	2145	19541965	2268343	9479194	1210206	10431
15	15	БНП-Дрез,	2455	NDMO	1	Санкт-Пет	185	17689833	84312	13082069	232842	2748
16	16	АВТОБАН	30	-	25	Москва	2732	17229288	1530584	15502811	1673414	3433
17	17	ОАО Банк	1776	PCBM	13	Москва	1006	16690301	727690	10372701	1092558	6381
18	18	АКБ "НРБ	2170	NRB/B	0	Москва	320	15791359	5075844	15773889	228337	3523
19	19	АКБ "Евро	2402	EFIN,	1	Москва	298	15191022	2835944	13398039	1496066	3256
20	20	ООО Райс	3292	RZBM	0	Москва	212	14892329	593460	11685822	865363	1784
21	21	КБ "ГУТА-	1623	GUTA	33	Москва	1906	14799899	2008471	12398573	567578	5828
22	22	ГЛОБЭКС	1942	GLBX	8	Москва	133	12769906	5016721	12576820	192899	532
23	23	ОАО "Хан	1971	-	6	Ханты-Ман	473	12722589	323652	2987333	214723	258

Сначала импортируем данные из xls-файла в среду аналитического пакета.

На первом шаге мастера запускаем мастер обработки и выбираем из списка метод обработки "Карта Кохонена". Далее следует настроить назначения столбцов, т.е. для каждого столбца выбрать одно из назначений: входное, выходное, не используется и информационное. Укажем всем столбцам, соответствующим показателям деятельности банков, назначение "Входной". "Выходной" не назначаем.

Следующий шаг предлагает разбить исходное множество на обучающее, тестовое и валидационное. По умолчанию,программа предлагает разбить множество на обучающее - 95% и тестовое - 5%.

На шаге № 5, изображенном на рис. предлагается настроить параметры карты: количество ячеек по X и по Y их форму (шестиугольную или четырехугольную).

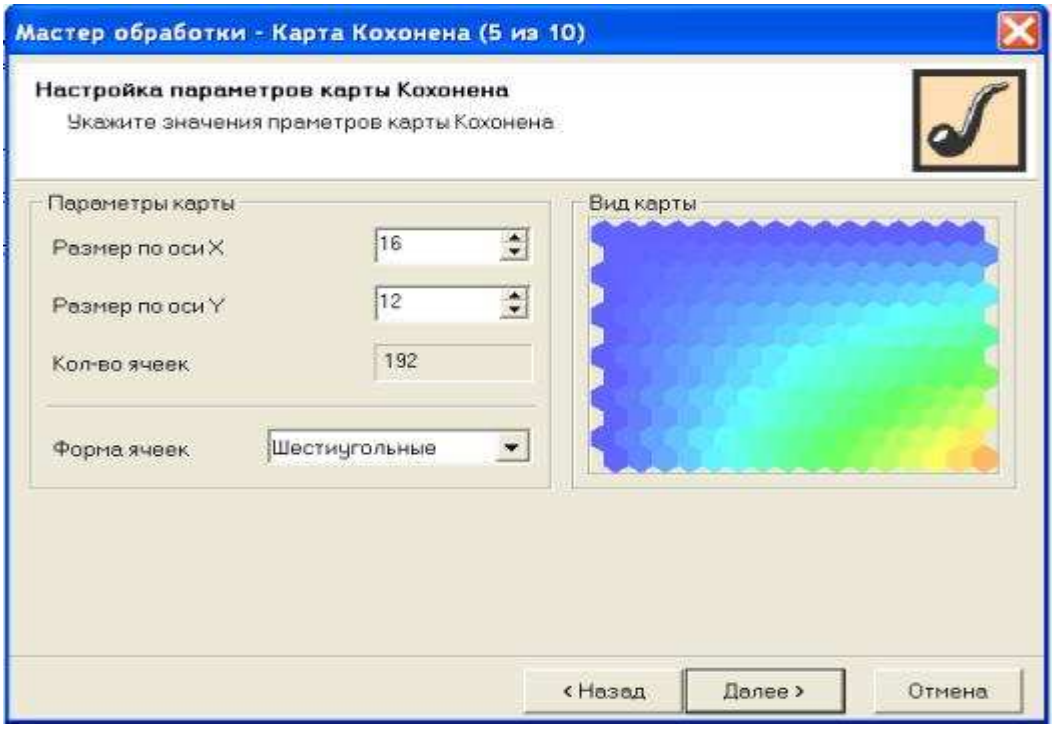


Рис. Шаг № 5 "Настройка параметров карты Кохонена"

На шестом шаге "Настройка параметров остановки обучения", проиллюстрированном на [рис.](#) , устанавливаем параметры остановки обучения и устанавливаем эпоху, по достижению которой обучение будет прекращено.

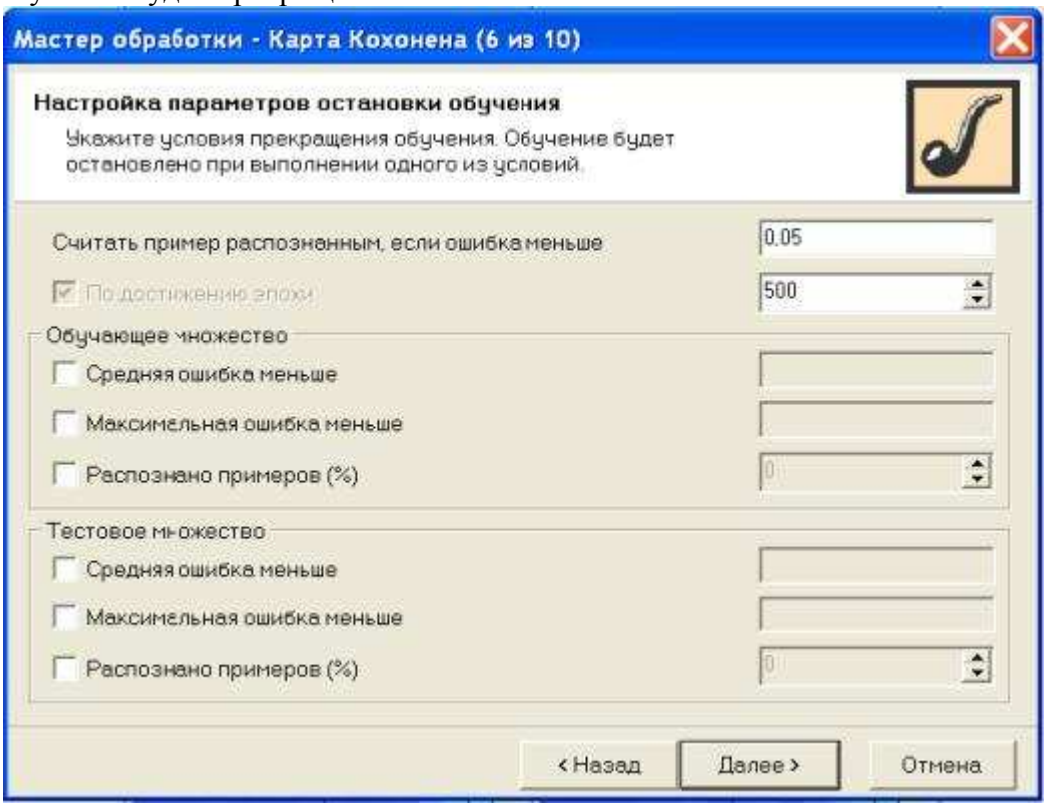


Рис. Шаг № 6 "Настройка параметров остановки обучения"

На седьмом шаге, представленном на рис. , настраиваются другие параметры обучения: способ начальной инициализации, тип функции соседства. Возможны два варианта кластеризации: автоматическое определение числа кластеров с соответствующим уровнем значимости и фиксированное количество кластеров (определяется пользователем). Поскольку нам неизвестно количество кластеров, выберем автоматическое определение их количества.

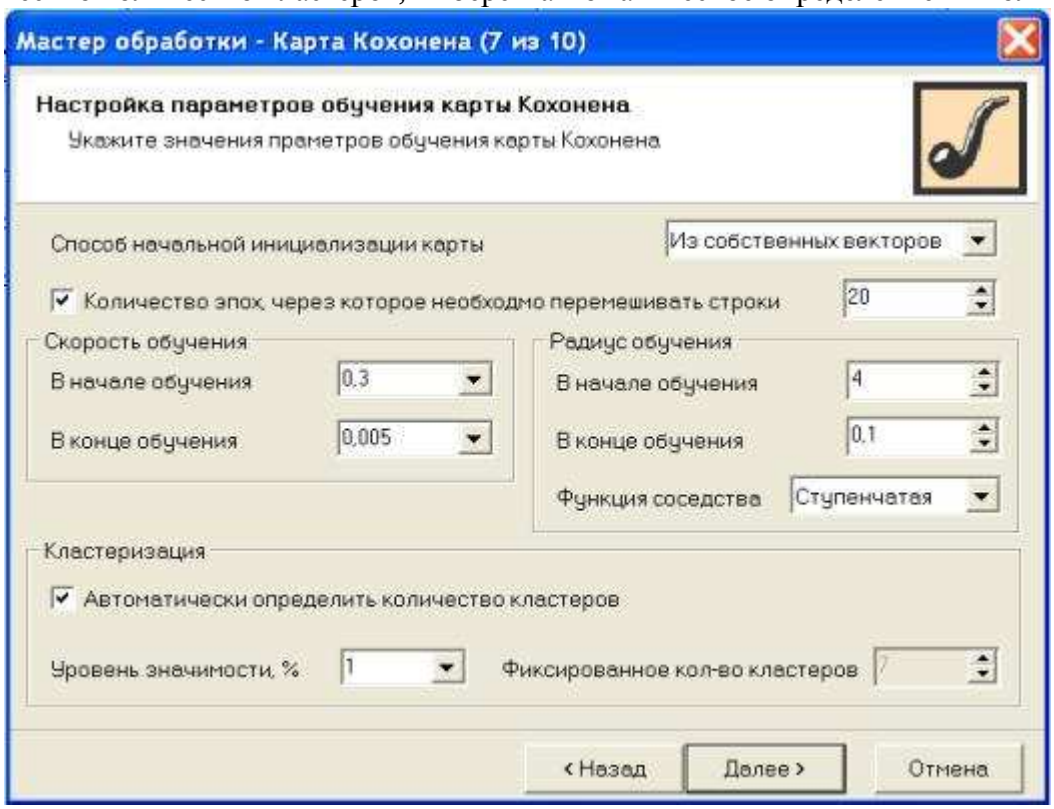


Рис. Шаг № 7 "Настройка параметров остановки обучения"

На восьмом шаге запускаем процесс обучения сети - необходимо нажать на кнопку "Пуск" и дождаться окончания процесса обучения. Во время обучения можем наблюдать изменение количества распознанных примеров и текущие значения ошибок. Этот процесс аналогичен тому, что мы рассматривали при обучении нейронных сетей в предыдущей лекции.

По окончании обучения в списке визуализаторов выберем "Карту Кохонена" и визуализатор "Что-если". На последнем шаге настраиваем отображения карты Кохонена, этот шаг проиллюстрирован на рис. .

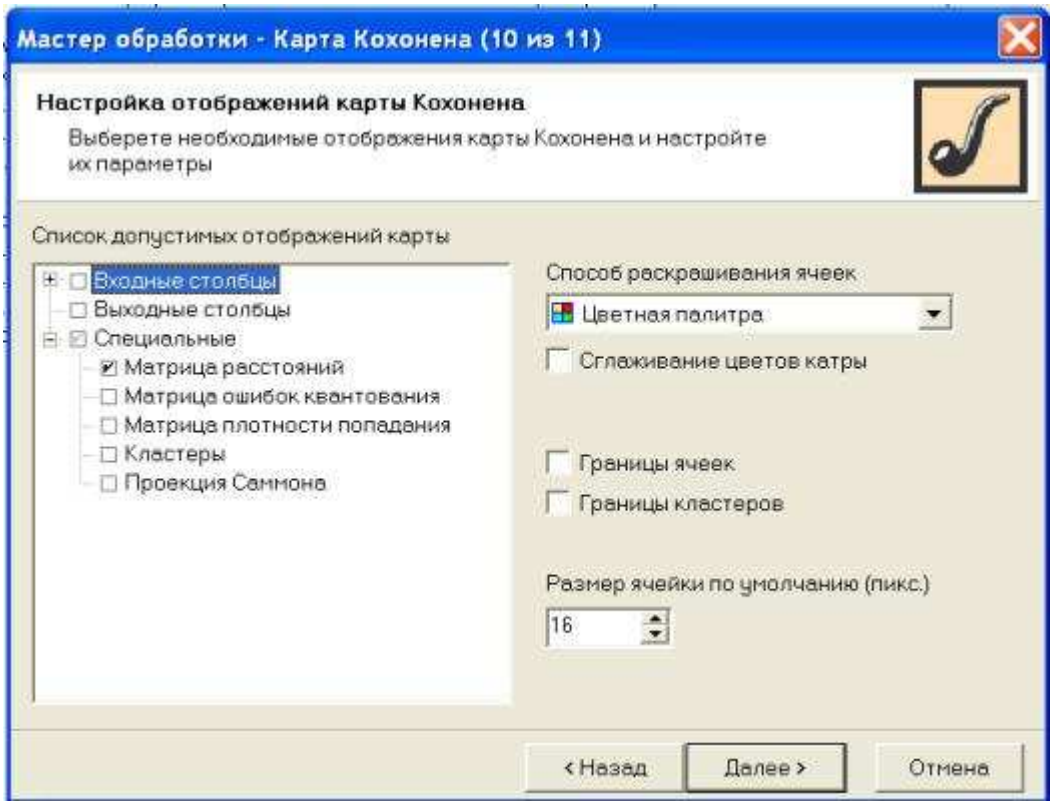


Рис. "Шаг № 10 Настройка отображений карты Кохонена"

Укажем отображения всех входных, выходных столбцов, кластеров, а также поставим флажок "Границы кластеров" для четкого отображения границ.

Карты входов

При анализе *карт входов* рекомендуют использовать сразу несколько карт. Исследуем фрагмент карты, состоящий из карт трех входов, который приведен на рисунке:

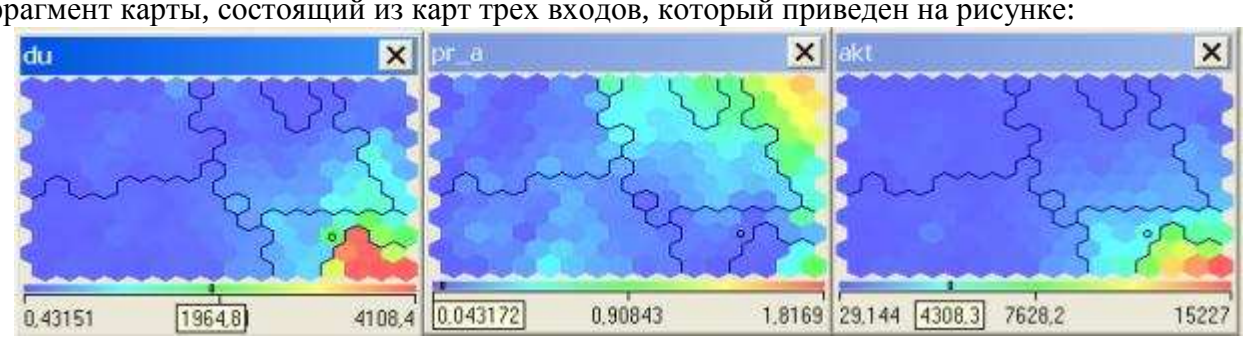


Рис. Карты трех входов

На одной из карт выделяем область с наибольшими значениями показателя. Далее имеет смысл изучить эти же нейроны на других картах.

На первой карте наибольшие значения имеют объекты, расположенные в правом нижнем углу. Рассматривая одновременно три карты, мы можем сказать, что эти же объекты имеют наибольшие значения показателя, изображенного на третьей карте. Также по раскраске первой и третьей карты можно сделать вывод, что существует взаимосвязь между этими показателями.

Также мы можем определить, например, такую характеристику: кластер, расположенный в правом верхнем углу, характеризуется низкими значениями показателей du (депозиты юридических лиц) и akt (активы банка) и высокими значениями показателей pr_a (прибыльность активов).

Эта информация позволяет так охарактеризовать кластер, находящийся в правом верхнем углу: это банки с небольшими активами, небольшими привлеченными депозитными средствами от юридических лиц, но с наиболее прибыльными активами, т.е. это группа небольших, но наиболее прибыльных банков.

Это лишь фрагмент вывода, который можно сделать, исследуя карту.

На следующем рисунке приведена иллюстрация *карт входов* и *выходов*, последняя - эта карта кластеров. Здесь мы видим несколько *карт входов* (показателей деятельности банков) и сформированные кластеры, каждый из которых выделен отдельным цветом.

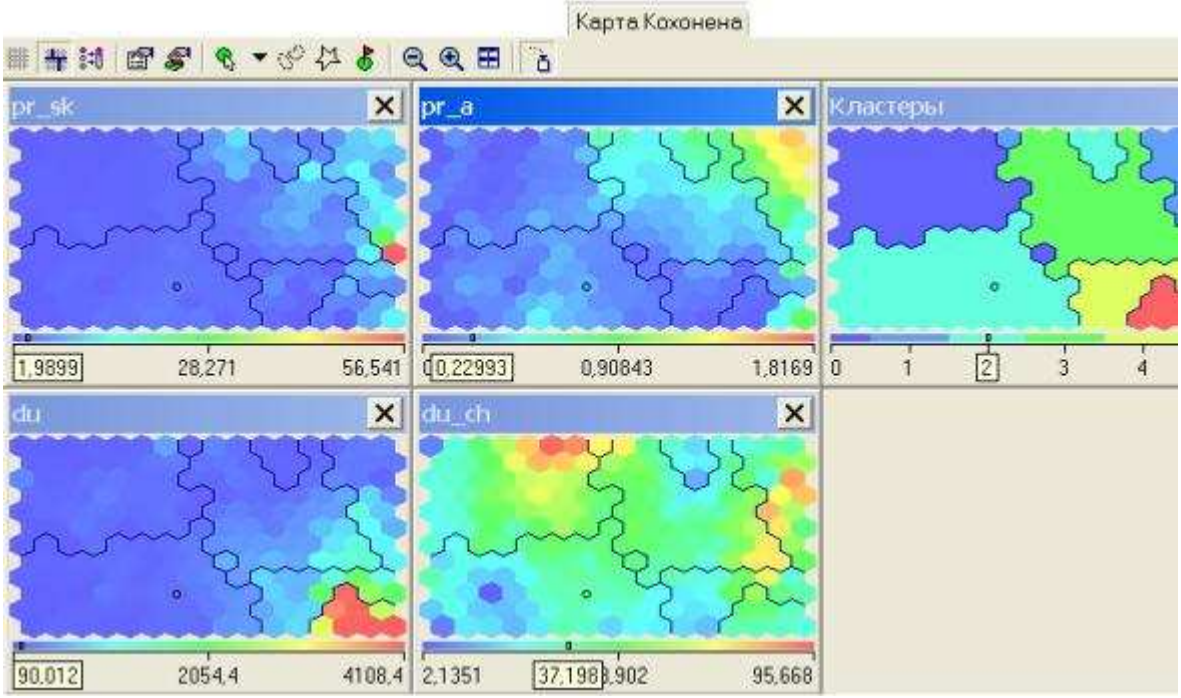


Рис. Карты входов и выходов

Для нахождения конкретного объекта на карте необходимо нажать правой кнопкой мыши на исследуемом объекте и выбрать пункт "Найти ячейку на карте". Выполнение этой процедуры показано на рис. В результате мы можем видеть как сам объект, так и значение того измерения, которое мы просматриваем. Таким образом, мы можем оценить положение анализируемого объекта, а также сравнить его с другими объектами.

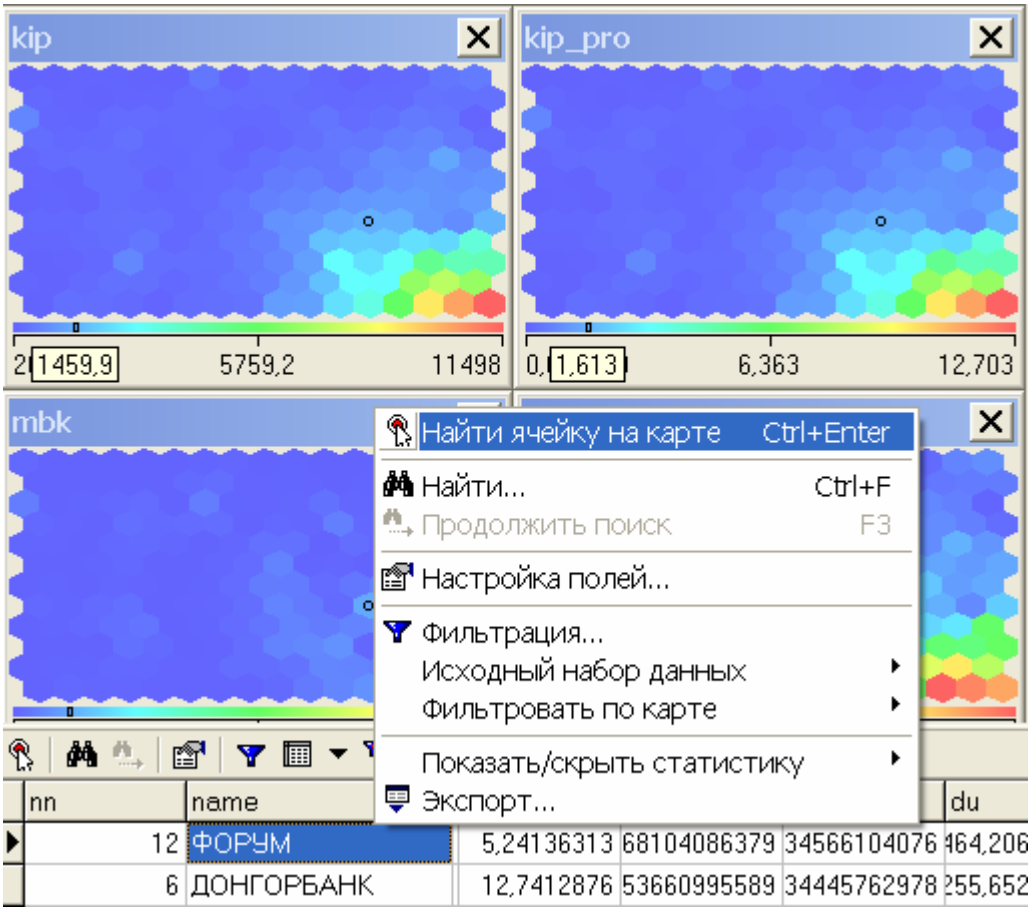


Рис. Ячейка на карте

В результате применения *самоорганизующихся карт* многомерное пространство входных факторов было представлено в двухмерном виде, в котором его достаточно удобно анализировать.

Банки были классифицированы на 7 групп, для каждой из которых возможно определение конкретных характеристик, исходя из раскраски соответствующих показателей.

Выводы

Основное отличие этих сетей от других моделей состоит в наглядности и удобстве использования. Эти сети позволяют упростить многомерную структуру, их можно считать одним из методов проецирования многомерного пространства в пространство с более низкой размерностью.

Интенсивность цвета в определенной точке карты определяется данными, которые туда попали: ячейки с минимальными значениями изображаются темно-синим цветом, ячейки с максимальными значениями - красным.

Другое принципиальное отличие карт Кохонена от других моделей нейронных сетей - иной подход к обучению, а именно - неуправляемое или неконтролируемое обучение.

Этот тип обучения позволяет данным обучающей выборки содержать значения только входных переменных.

Сеть Кохонена учится понимать саму структуру данных и решает задачи кластеризации.

Лекция 9. Кластеризация в Матлаб и MemBrain.

Пример решения задачи в пакете Matlab

Пакет Матлаб после 2002 года (версии после 6.5) резко изменился. Изменились имена функций, состав и имена встроенных в пакет нейросетей. После 2002 года и название программного средства изменилось – в названии стал указываться год создания версии пакета. Для создания новой нейросети Кохонена (SOM) стала использоваться функция «selforgmap».

Доступная же в России информация о создаваемых этим пакетом нейросетях содержится в учебных материалах, базирующихся на версии до Matlab v.6.5. Эта версия доступна в Интернет в достаточно полном объёме. Изучать возможности нейропакетов можно и на ней (см. например, http://pandia.ru/text/78/102/560.php).

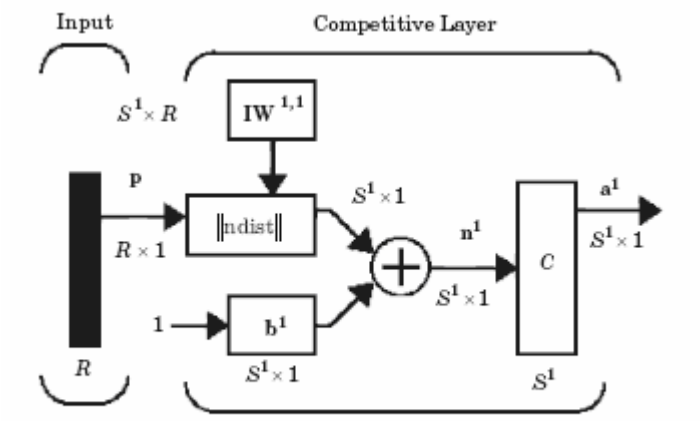
Карта Кохонена в Матлаб 6.5.

Для создания самоорганизующейся карты Кохонена SOM (Self Organizing Map) в Матлаб 6.5 предусмотрена М-функция newsom. Создание же слоя Кохонена выполняется с помощью М-функции newc.

Для сетей с конкурирующим слоем по умолчанию используется обучающая функция trainwb1, которая на каждом цикле обучения случайно выбирает входной вектор и предъявляет его сети; после этого производится коррекция весов и смещений.

Сеть этого типа использует функцию евклидова расстояния dist, функцию инициализации initwb, функцию обработки входов netsum, функцию активации compet и функцию описания топологии hextop.

Соревновательный слой Кохонена в Matlab v.6.5 может быть представлен в виде схемы:



Соревновательный слой Кохонена в MatLab

Рассмотрим задачу кластеризации точек на плоскости.

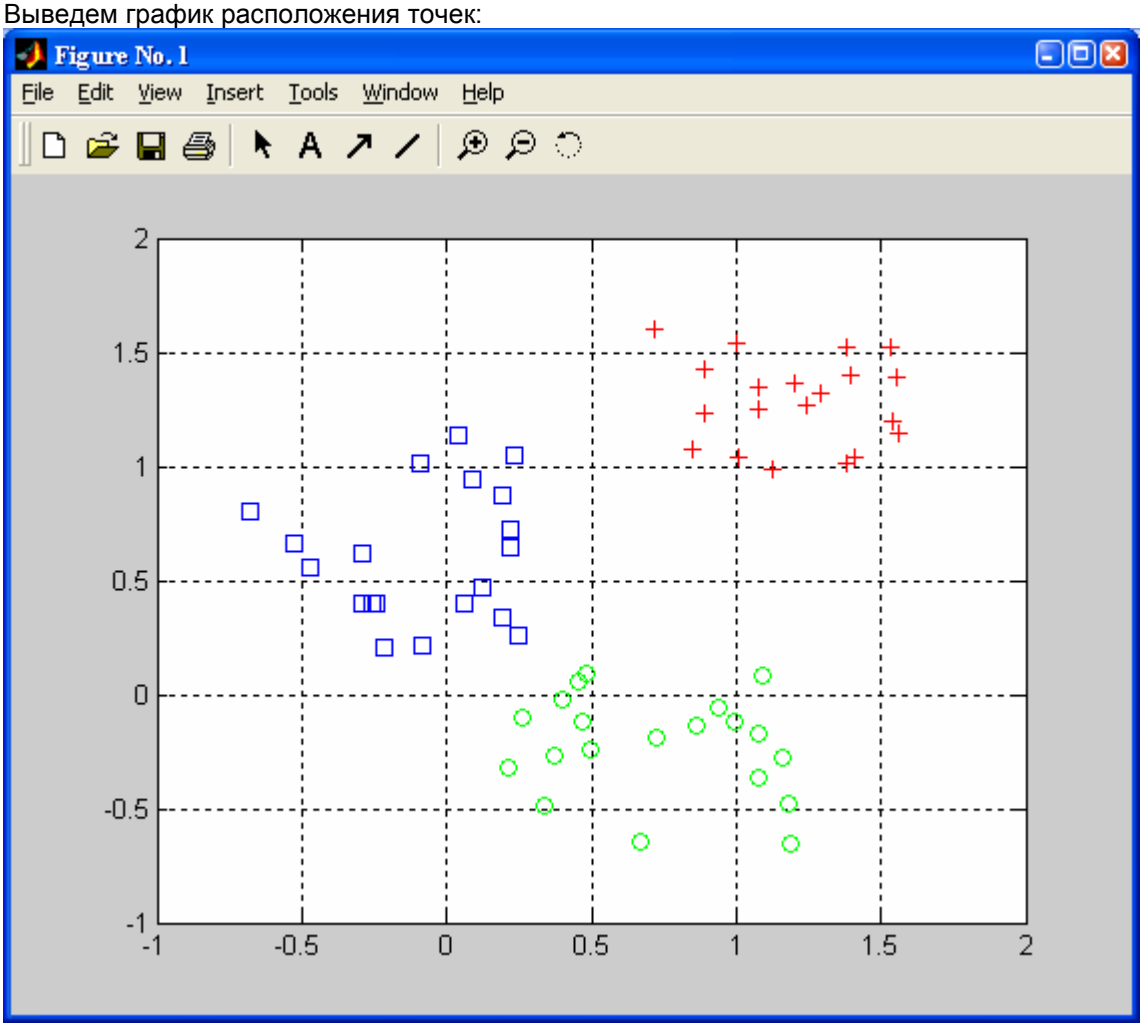
Пусть даны четыре входных вектора:

```
X=[1 1 0 0; 0 0 0 1; 1 0 0 0; 0 0 1 1]
```

Разместим случайным образом 60 точек на плоскости с помощью команд:

```
A = [rand(1,20) - 0.7; rand(1,20) + 0.2];
B = [rand(1,20) + 0.7; rand(1,20) + 0.7];
C = [rand(1,20) + 0.2; rand(1,20) - 0.7];
plot(A(1,:),A(2,:),'bs')
hold on
plot(B(1,:),B(2,:),'r+')
plot(C(1,:),C(2,:),'go')
grid on
P = [A, B, C]
```

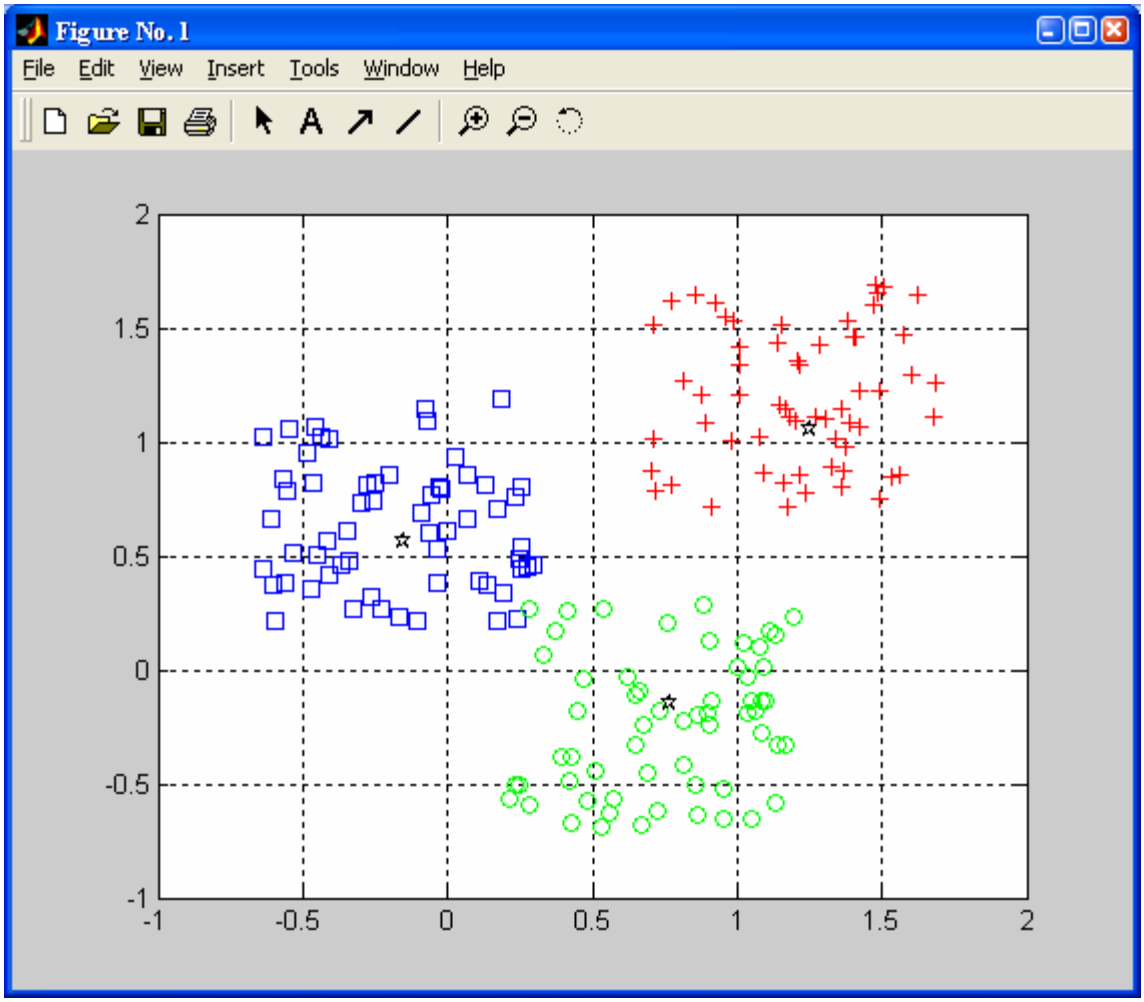
Выведем график расположения точек:



Проведём обучение слоя конкурирующих нейронов и выведем результат кластеризации:

```
ncl = 3;
MN=[min(X(1,:)) max(X(1,:)); min(X(:,2)) max(X(:,2)) ]
net = newc(MN, ncl, 0.1, 0.0005);
net.trainParam.epochs=49;
net.trainParam.show=7;
net = train(net,P);
w = net.IW{1};
plot(w(:,1),w(:,2),'kp');
```

Результат обучения приведен на следующем рисунке, где звездочками обозначены центры трех кластеров.



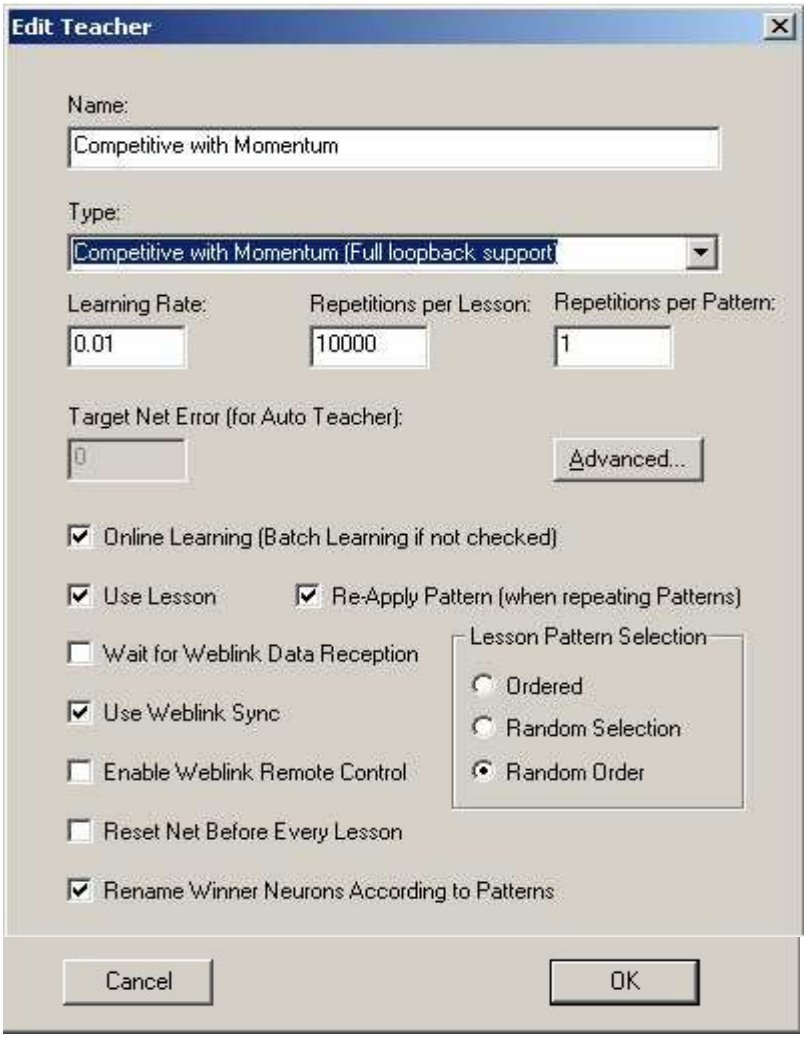
Определение трех кластеров на плоскости. Звёздочками отмечены центры кластеров.

Пример решения задачи в пакете MemBrain

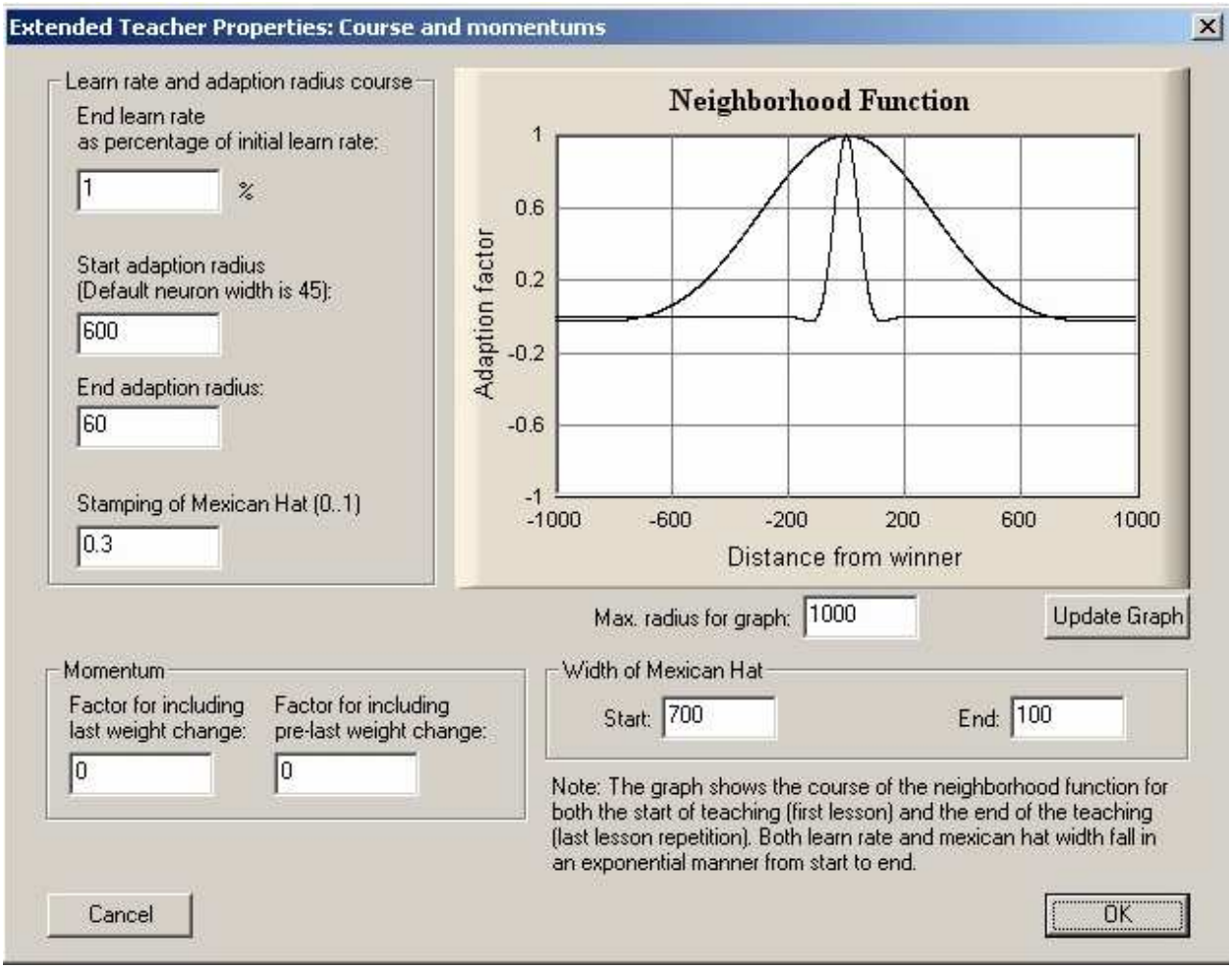
В папке MemBrainExamples содержатся файлы SOM.mbn и SOM.mbl с примером решения задачи кластеризации.

Этот пример демонстрирует работу карты SOM (Self Organizing Map) размером 10x10 выходных нейронов в нейросети Кохонена с 2 входами.

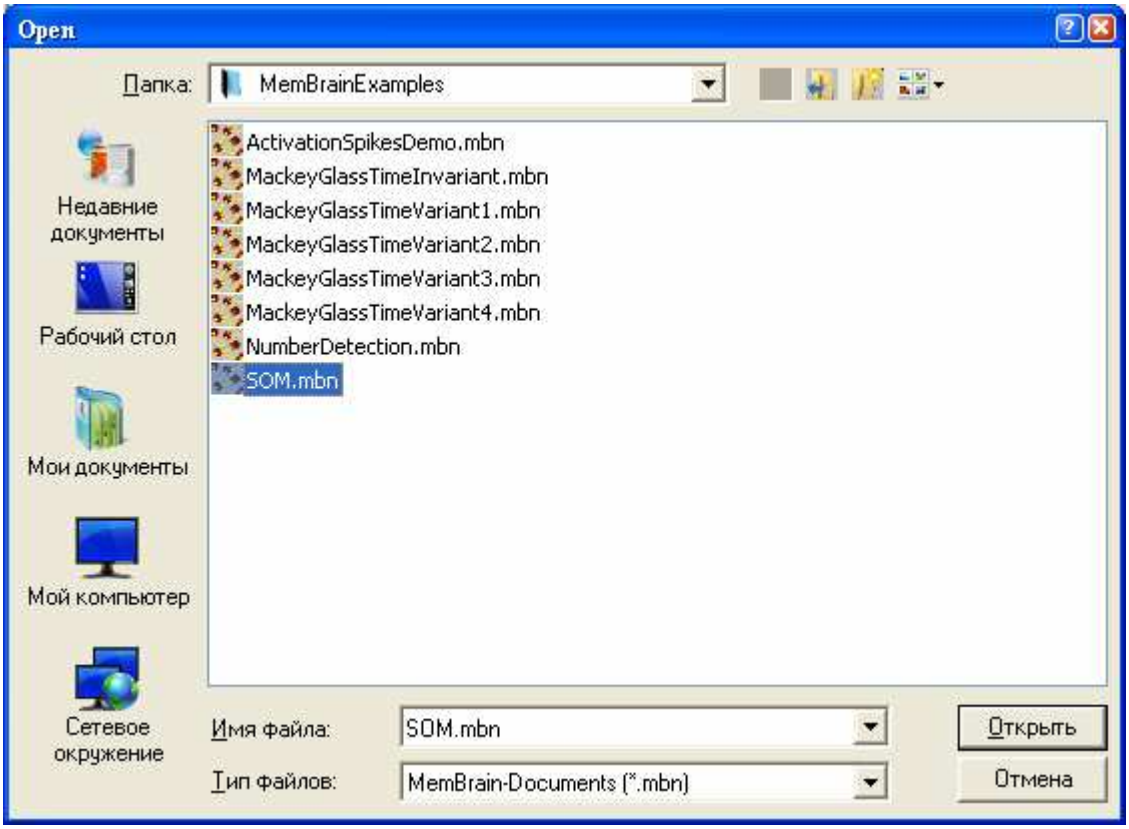
Для обучения без учителя используется метод соревновательного обучения 'Competitive with Momentum'. Приводимые далее настройки для обучения могут быть использованы, как пример. Выберите пункт меню Teach -> Teacher Manager -> Edit:

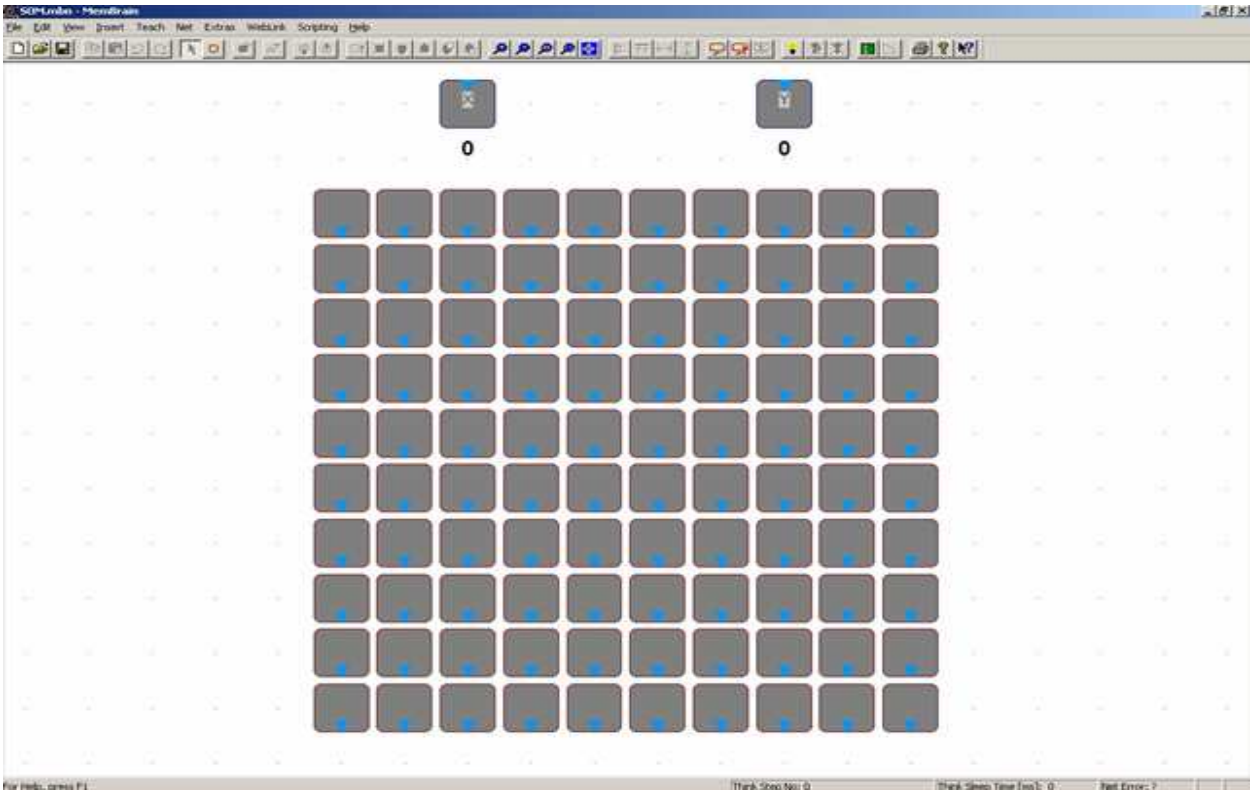


Если щёлкнуть по клавише 'Advanced' (Продвижение, или Вперёд!)? откроется окно для расширенной настройки свойств :

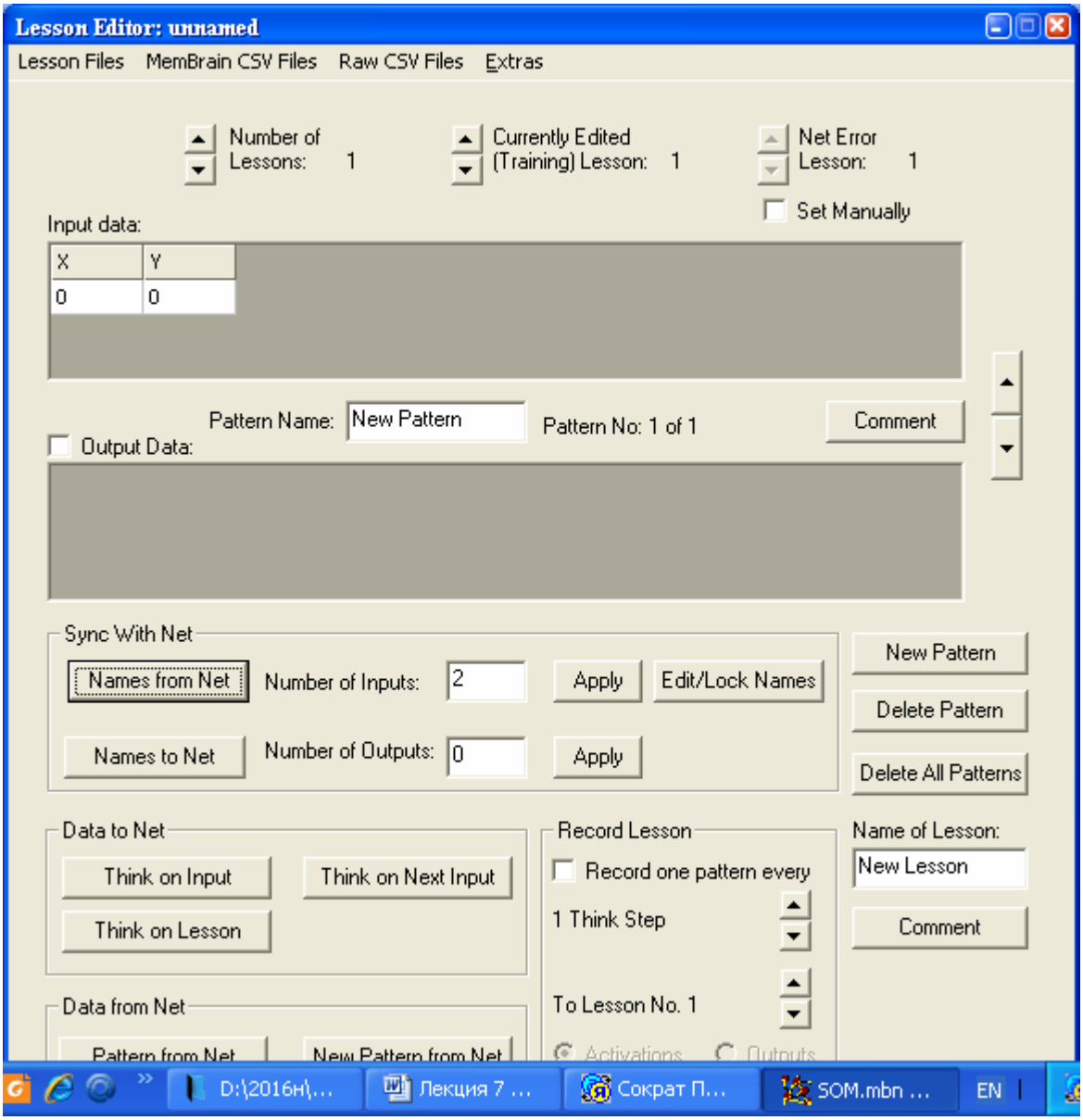


Если загрузить пример SOM из файла SOM.mbn, появится схема нейросети, содержащая 2 входных нейрона и 100 выходных:

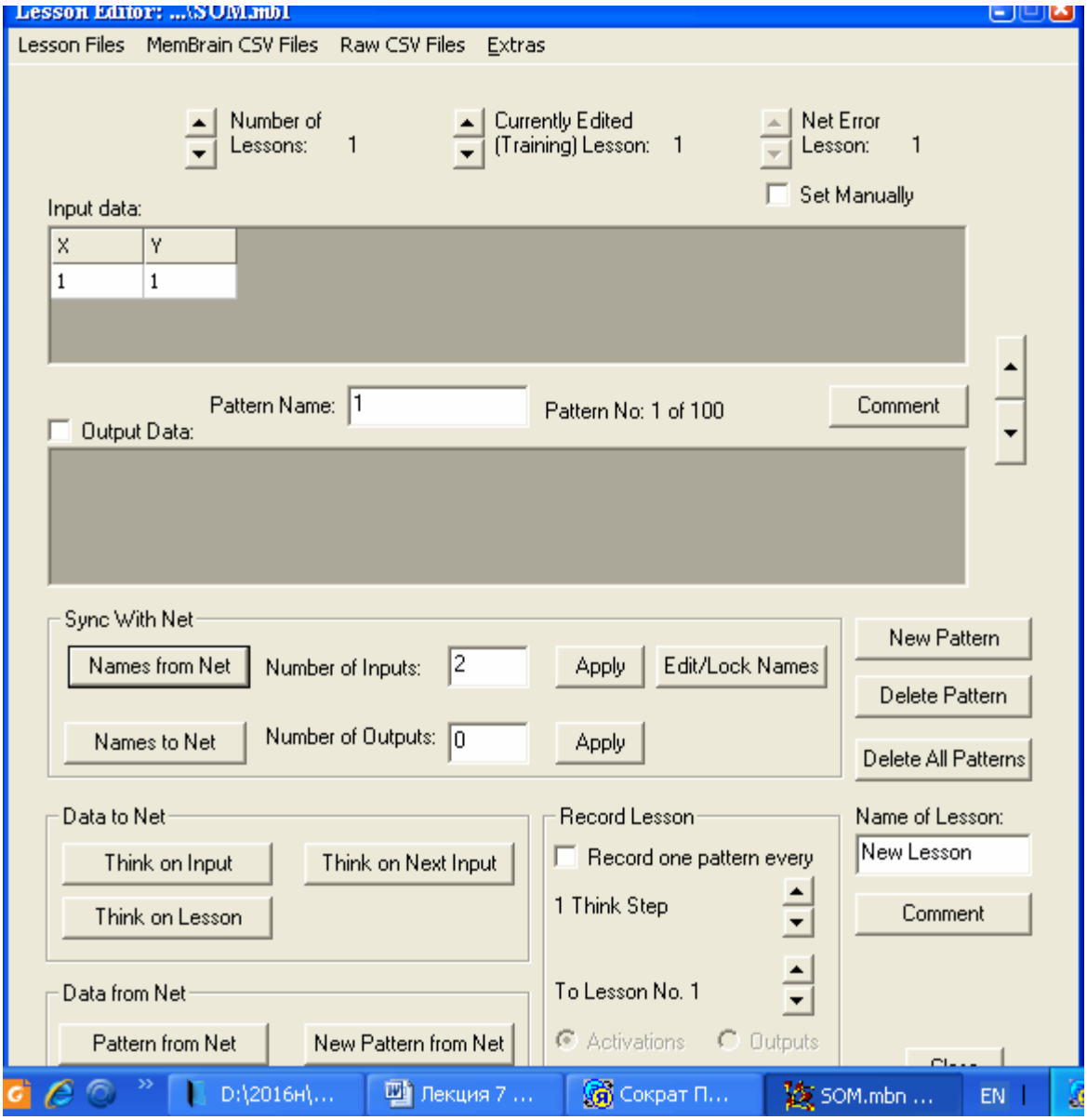




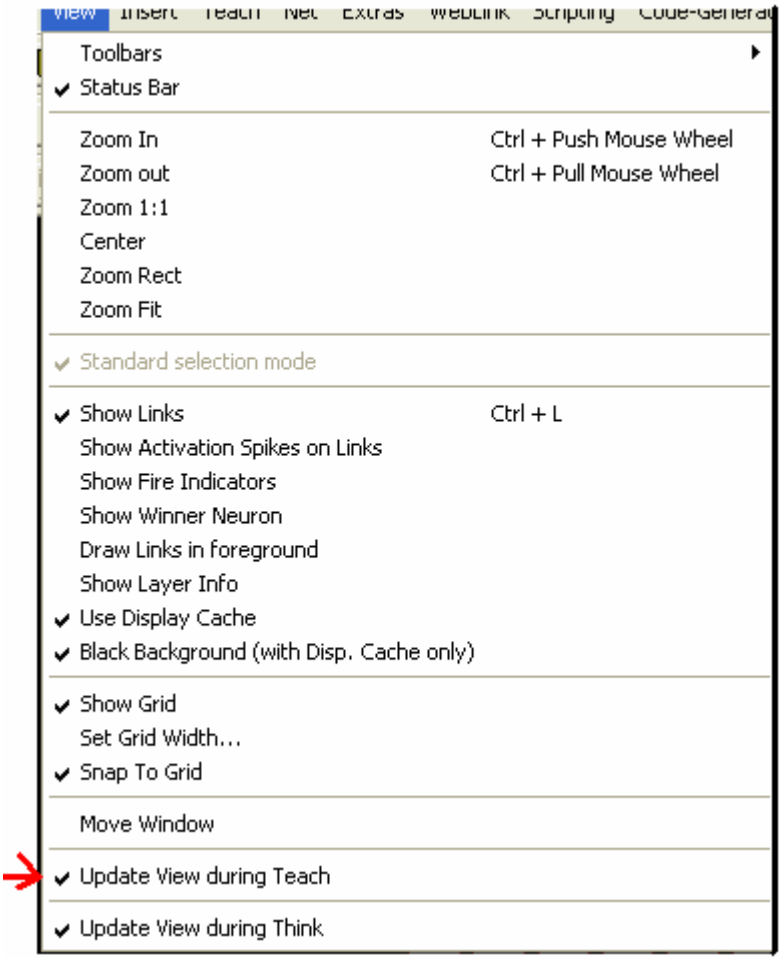
Загрузите урок SOM.mbl в редактор уроков (Lesson Editor) и начинайте обучение.



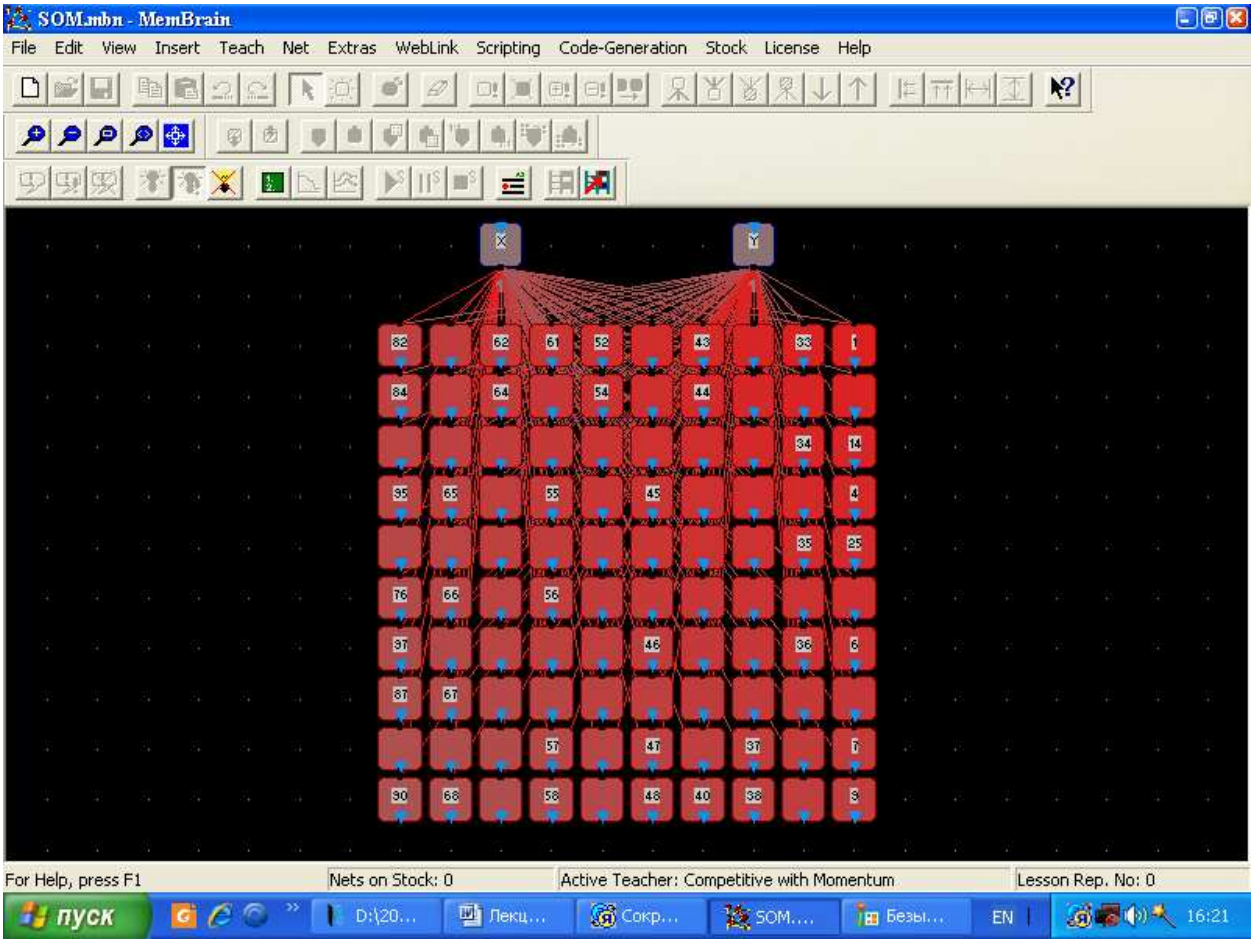
Lessons Files -> MemBrain Examles -> Som.mbl



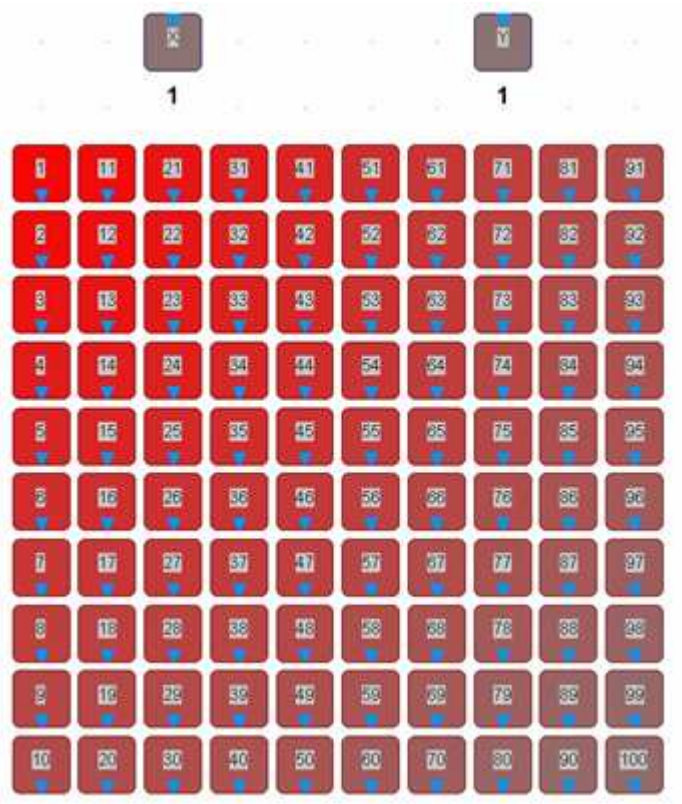
Удостоверьтесь, что активирована настройка <View><Update View during Teach>.



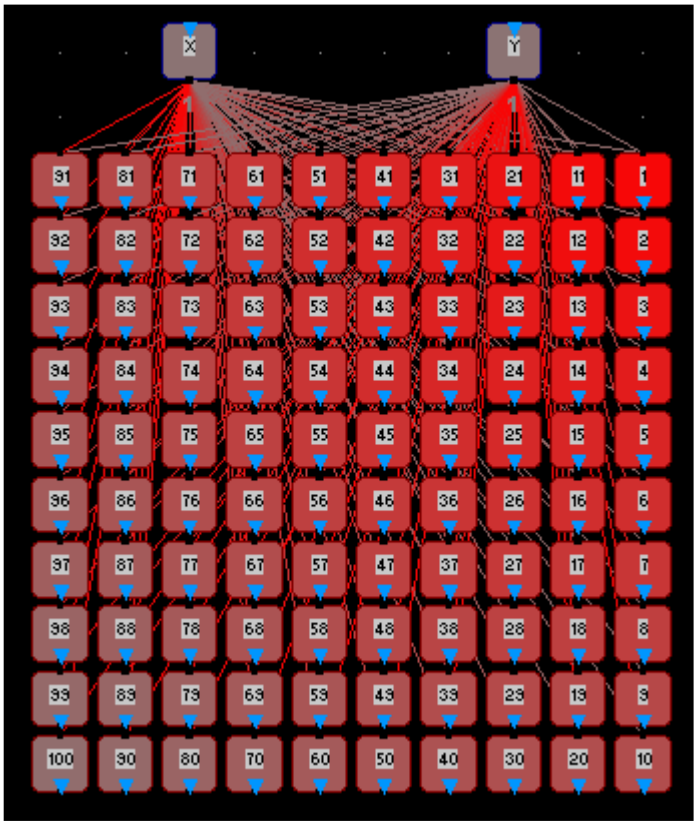
Выбрать меню Teach -> Teach Step



Станет видно, как активируются примеры (patterns) текущего урока (именуемые номерами от 1 до 100) появляются на карте SOM и находят позиции, локализующие их на карте. Окончательно, после обучения карта SOM должна выглядеть похожей на:



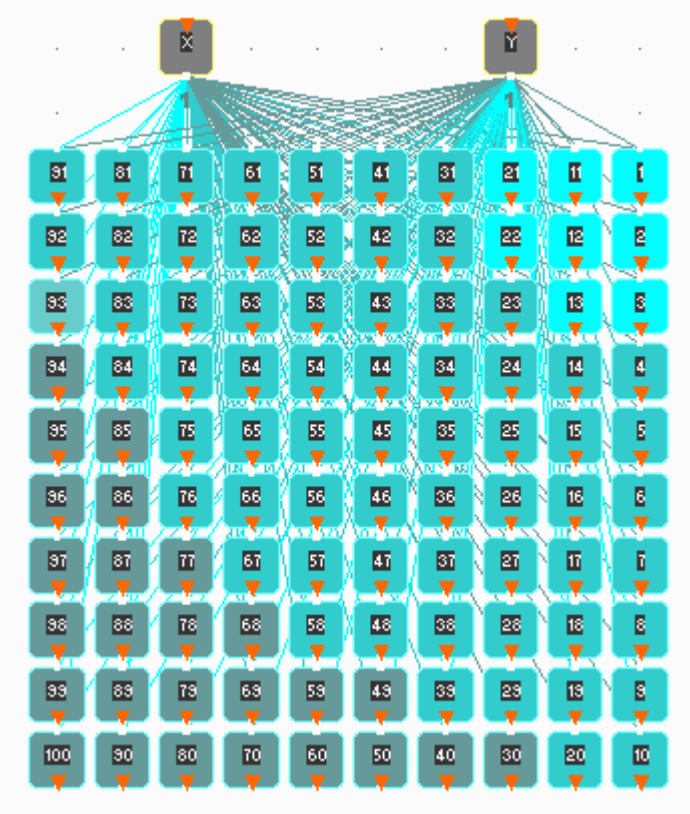
Или



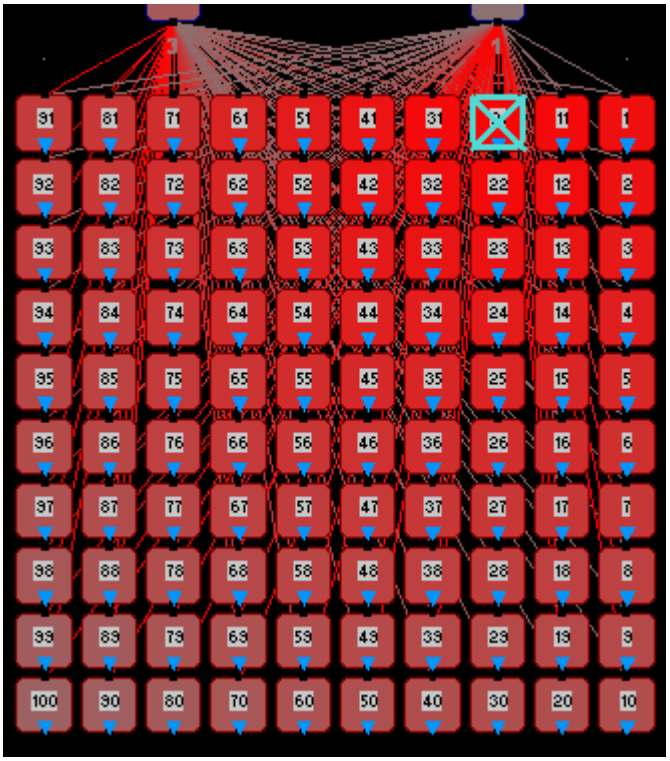
Выходные нейроны карты названы в соответствии с примерами, для которых они явились победителями, т.е. крайний левый нейрон назван '1', поскольку он был победителем в

примере 1 данного урока. Урок состоит из образцов с X- и Y- входными величинами, колеблющимися от 1 до 100.

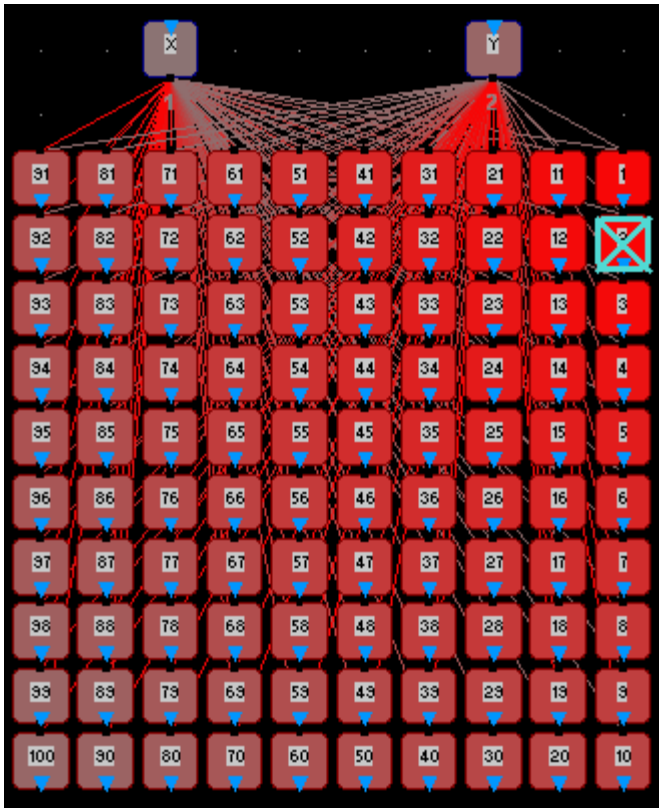
Поскольку аналогичные образцы объединяются вместе, в SOM после обучения именам примеров, совпадающим с их координатами, значения размещаются в соответствующих этим координатам строках и колонках. В процессе обучения строки и колонки могут быть изменены по сравнению с указанными на рисунке. Их размещение зависит от стартовых величин рандомизации для веса связей и так же от порядка предъявляемых примеров при обучении.



Теперь нажмите на пункт меню <View><Show Winner Neuron>,

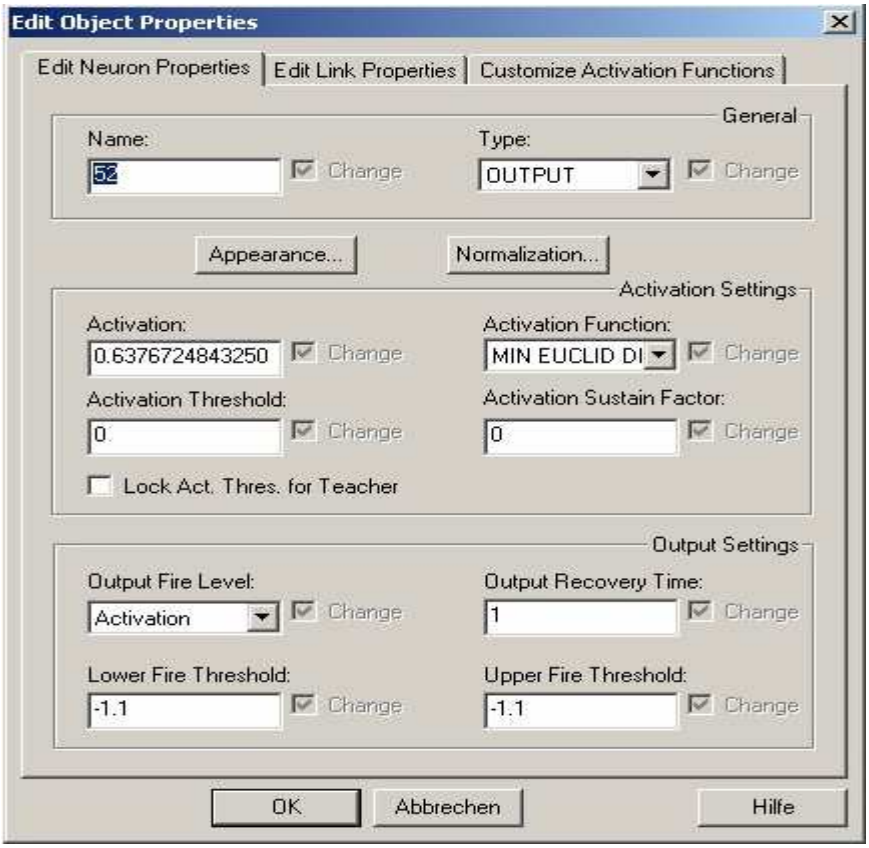


открывая редактор уроков нажмите на кнопку <Think on Next Input> несколько раз. Вы увидите, что нейрон-победитель, соответствующий входному примеру, визуализируется на SOM-карте синим пересечением:

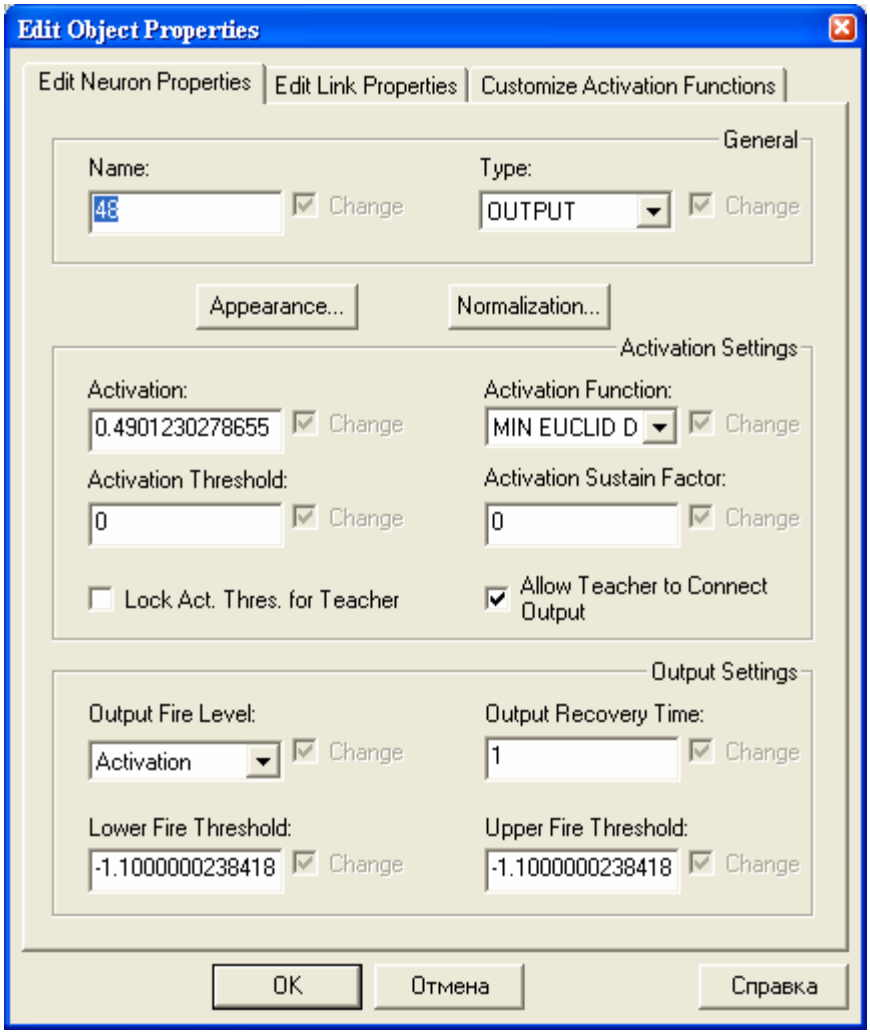


Эту функцию можно использовать, если нужно быстро идентифицировать нейрон-победитель, когда предъявляется новый, не использованный при обучении образ.

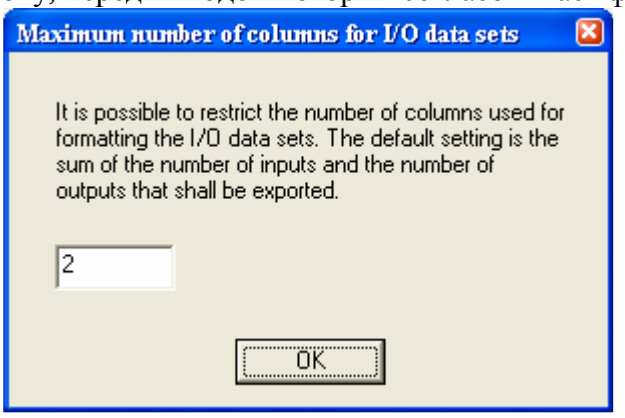
Функция активизации для выходных нейронов SOM – есть 'MIN_EUCLID_DIST', которая означает 'Minimum Euclidian Distance' – минимальное эвклидово расстояние. Это можно увидеть, если дважды щелкнут по одному из выходных нейронов:


Edit Object Properties
Edit Neuron Properties
Edit Link Properties
Customize Activation Functions
General
Name:
52
Change
Type:
OUTPUT
Change
Appearance...
Normalization...
Activation Settings
Activation:
0.6376724843250
Change
Activation Function:
MIN EUCLID DI
Change
Activation Threshold:
0
Change
Activation Sustain Factor:
0
Change
Lock Act. Thres. for Teacher
Output Settings
Output Fire Level:
Activation
Change
Output Recovery Time:
1
Change
Lower Fire Threshold:
-1.1
Change
Upper Fire Threshold:
-1.1
Change
OK
Abbrechen
Hilfe

Или



Чтобы увидеть, что собой представляет обучающая выборка, в Lesson Editor выбираем Raw CSW Files - > Export Carrent Lesson (Raw CSW) - > кластеризация.csv. Файл содержит 101 строку, перед выводом которых согласовывает формат вывода:



Текст файла кластеризация.csv:

X	Y
1	1
1	2
1	3

1	4
1	5
1	6
1	7
1	8
1	9
1	10
2	1
2	2
2	3
2	4
2	5
2	6
2	7
2	8
2	9
2	10
3	1
3	2
3	3
3	4
3	5
3	6
3	7
3	8
3	9
3	10
4	1
4	2
4	3
4	4
4	5
4	6
4	7
4	8
4	9
4	10
5	1
5	2
5	3
5	4
5	5
5	6
5	7
5	8
5	9
5	10
6	1
6	2
6	3
6	4
6	5
6	6

6	7
6	8
6	9
6	10
7	1
7	2
7	3
7	4
7	5
7	6
7	7
7	8
7	9
7	10
8	1
8	2
8	3
8	4
8	5
8	6
8	7
8	8
8	9
8	10
9	1
9	2
9	3
9	4
9	5
9	6
9	7
9	8
9	9
9	10
10	1
10	2
10	3
10	4
10	5
10	6
10	7
10	8
10	9
10	10

Лекция 10. Таксономия в проблематике искусственных когнитивных систем

В состав группы задач, решаемых при разработке искусственных когнитивных систем входят такие задачи, как: кластеризация, таксономия, автоматическая классификация, узнавание, идентификация, типология, обучение классификации. Краткая характеристика их:

- Кластеризация – это разделение накопленных образов на группы одинаковых, “похожих” объектов. Кластеризация позволяет систематизировать накопленную информацию в тот период, когда по ней практически ничего неизвестно. Полученные группы схожих образов называются кластерами или таксонами.
- “Таксономия (от греч. táxis — расположение, строй, порядок и nómos — закон) представляет собой теорию классификации и систематизации сложноорганизованных областей действительности, имеющих обычно иерархическое строение (органический мир, объекты географии, геологии, языкознания, этнографии и т. п.)” (http://ru.wikipedia.org/wiki).
- Задачи автоматической классификации и таксономии имеют следующую формулировку: имеется группа образов (объектов, кластеров, таксонов). Нужно определить, к какой из имеющихся групп принадлежит новый объект. Если ни на один из имеющихся новый объект не похож, для него создаётся новый кластер.
- Задачи узнавания (собственно распознавания образов) имеют следующую постановку: классификация уже известна. Предъявляется новый образ (объект). Нужно определить, к какому кластеру (таксону) он относится. При отрицательном результате новый объект зачисляется в группу нераспознанных. Новых кластеров не образуется.
- Идентификация образов (объектов) сводится к задаче распознавания, если каждый кластер состоит только из одного объекта.
- Задачи типологии имеют следующую постановку: классификация известна. Необходимо описать структуру каждого класса, выявить существенные признаки и характерные черты класса, описать эталонный объект каждого кластера, сформировать отличительные черты кластеров.
- Задачи обучения классификации формулируются следующим образом: выработать правило классификации, т.е. найти классифицирующие признаки или технологии, позволяющие классифицировать образы (объекты). При использовании нейронных сетей, нейропакетов, нейроимитаторов обучение классификации может заключаться в переносе коэффициентов связи обученной сети в соответствующее место необученной.

Автоматическая классификация.

Классификация является наиболее простой и одновременно наиболее часто решаемой задачей Data Mining. Ввиду распространенности задач классификации необходимо четкое понимания сути этого понятия.

Приведем несколько характеризующих данное понятие определений.

Классификация - системное распределение изучаемых предметов, явлений, процессов по родам, видам, типам, по каким-либо существенным признакам для удобства их исследования; группировка исходных понятий и расположение их в определенном порядке, отражающем степень этого сходства.

Классификация - упорядоченное по некоторому принципу множество объектов, которые имеют сходные классификационные признаки (одно или несколько свойств), выбранных для определения сходства или различия между этими объектами.

Классификация требует при разделении объектов соблюдения следующих правил:

- в каждом акте деления необходимо применять только одно основание;
- деление должно быть соразмерным, т.е. общий объем видовых понятий должен равняться объему делимого родового понятия;
- члены деления должны взаимно исключать друг друга, их объемы не должны перекрещиваться;
- деление должно быть последовательным.

Различают:

- вспомогательную (искусственную) классификацию, которая производится по внешнему признаку и служит для придания множеству предметов (процессов, явлений) нужного порядка;
- естественную классификацию, которая производится по существенным признакам, характеризующим внутреннюю общность предметов и явлений. Она является результатом и важным средством научного исследования, т.к. предполагает и закрепляет результаты изучения закономерностей классифицируемых объектов.

В зависимости от выбранных признаков, их сочетания и процедуры деления понятий классификация может быть:

- простой - деление родового понятия только по признаку и только один раз до раскрытия всех видов. Примером такой классификации является дихотомия, при которой членами деления бывают только два понятия, каждое из которых является противоречащим другому (т.е. соблюдается принцип: "А и не А");
- сложной - применяется для деления одного понятия по разным основаниям и синтеза таких простых делений в единое целое. Примером такой классификации является периодическая система химических элементов.

Под классификацией будем понимать отнесение объектов (наблюдений, событий) к одному из заранее известных классов.

Классификация - это закономерность, позволяющая делать вывод относительно определения характеристик конкретной группы. Таким образом, для проведения классификации должны присутствовать признаки, характеризующие группу, к которой принадлежит то или иное событие или объект (обычно при этом на основании анализа уже классифицированных событий формулируются некие правила).

Классификация относится к стратегии обучения с учителем (supervised learning), которое также именуют контролируемым или управляемым обучением.

Задачей классификации часто называют предсказание категориальной зависимой переменной (т.е. зависимой переменной, являющейся категорией) на основе выборки непрерывных и/или категориальных переменных.

Например, можно предсказать, кто из клиентов фирмы является потенциальным покупателем определенного товара, а кто - нет, кто воспользуется услугой фирмы, а кто - нет, и т.д. Этот тип задач относится к задачам бинарной классификации, в них зависимая переменная может принимать только два значения (например, да или нет, 0 или 1).

Другой вариант классификации возникает, если зависимая переменная может принимать значения из некоторого множества предопределенных классов. Например, когда необходимо предсказать, какую марку автомобиля захочет купить клиент. В этих случаях рассматривается множество классов для зависимой переменной.

Классификация может быть одномерной (по одному признаку) и многомерной (по двум и более признакам).

Многомерная классификация была разработана биологами при решении проблем дискриминации для классифицирования организмов. Одной из первых работ, посвященных этому направлению, считают работу Р. Фишера (1930 г.), в которой организмы разделялись на подвиды в зависимости от результатов измерений их физических параметров. Биология была и остается наиболее востребованной и удобной средой для разработки многомерных методов классификации.

Рассмотрим задачу классификации на простом примере. Допустим, имеется база данных о клиентах туристического агентства с информацией о возрасте и доходе за месяц. Есть рекламный материал двух видов: более дорогой и комфортный отдых и более дешевый, молодежный отдых. Соответственно, определены два класса клиентов: класс 1 и класс 2.

Таблица 1. База данных клиентов туристического агентства.

Код клиента	Возраст	Доход	Класс
1	18	25	1
2	22	100	1
3	30	70	1
4	32	120	1
5	24	15	2
6	25	22	1
7	32	50	2
8	19	45	2
9	22	75	1
10	40	90	2

Задача. Определить, к какому классу принадлежит новый клиент и какой из двух видов рекламных материалов ему стоит отсылать.

Для наглядности представим нашу базу данных в двухмерном измерении (возраст и доход), в виде множества объектов, принадлежащих классам 1 (оранжевая метка) и 2 (серая метка).

На рис.2 приведены объекты из двух классов.

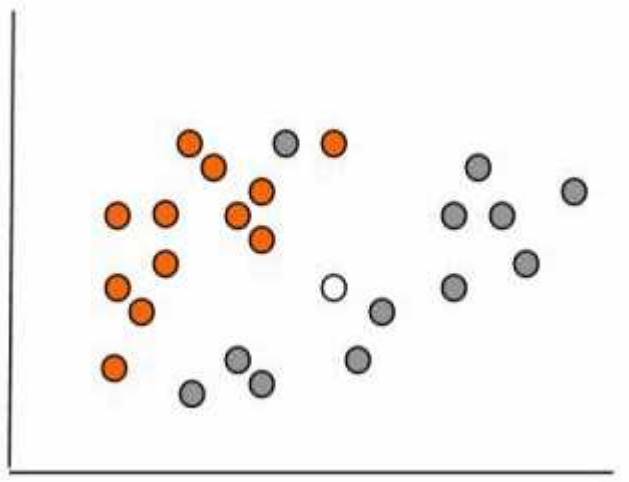


Рис. 2. Множество объектов базы данных в двухмерном измерениии

Решение нашей задачи будет состоять в том, чтобы определить, к какому классу относится новый клиент, на рисунке обозначенный белой меткой.

Процесс классификации

Цель процесса классификации состоит в том, чтобы построить модель, которая использует прогнозирующие атрибуты в качестве входных параметров и получает значение зависимого атрибута. Процесс классификации заключается в разбиении множества объектов на классы по определенному критерию.

Классификатором называется некая сущность, определяющая, какому из предопределенных классов принадлежит объект по вектору признаков.

Для проведения классификации с помощью математических методов необходимо иметь формальное описание объекта, которым можно оперировать, используя математический аппарат классификации. Таким описанием в нашем случае выступает база данных.

Каждый объект (запись базы данных) несет информацию о некотором свойстве объекта.

Набор исходных данных (или выборку данных) разбивают на два множества:

- обучающее
- и тестовое.

Обучающее множество (training set) - множество, которое включает данные, использующиеся для обучения (конструирования) модели. Такое множество содержит входные и выходные (целевые) значения примеров. Выходные значения предназначены для обучения модели.

Тестовое (test set) множество также содержит входные и выходные значения примеров. Здесь выходные значения используются для проверки работоспособности модели.

Процесс классификации состоит из двух этапов: конструирования модели и ее использования.

1. Конструирование модели:
    - описание множества предопределенных классов.
    - формирование обучающего набора данных, в котором каждый пример относится к одному предопределенному классу.
    - конструируется модель.

- полученная модель после обучения представляется классификационными правилами, деревом решений или математической формулой.

2. Использование модели: классификация новых или неизвестных значений.

Точность классификации: оценка уровня ошибок может проводиться при помощи кросс-проверки. Кросс-проверка (Cross-validation) - это процедура оценки точности классификации на данных из тестового множества, которое также называют кросс-проверочным множеством. Точность классификации тестового множества сравнивается с точностью классификации обучающего множества.

Если классификация тестового множества дает приблизительно такие же результаты по точности, как и классификация обучающего множества, считается, что данная модель прошла кросс-проверку.

Разделение на обучающее и тестовое множества осуществляется путем деления выборки в определенной пропорции, например обучающее множество - две трети данных и тестовое - одна треть данных.

Этот способ следует использовать для выборок с большим количеством примеров. Если же выборка имеет малые объемы, рекомендуется применять специальные методы, при использовании которых обучающая и тестовая выборки могут частично пересекаться.

Процесс классификации, а именно, конструирование модели и ее использование, представлен на рисунке:

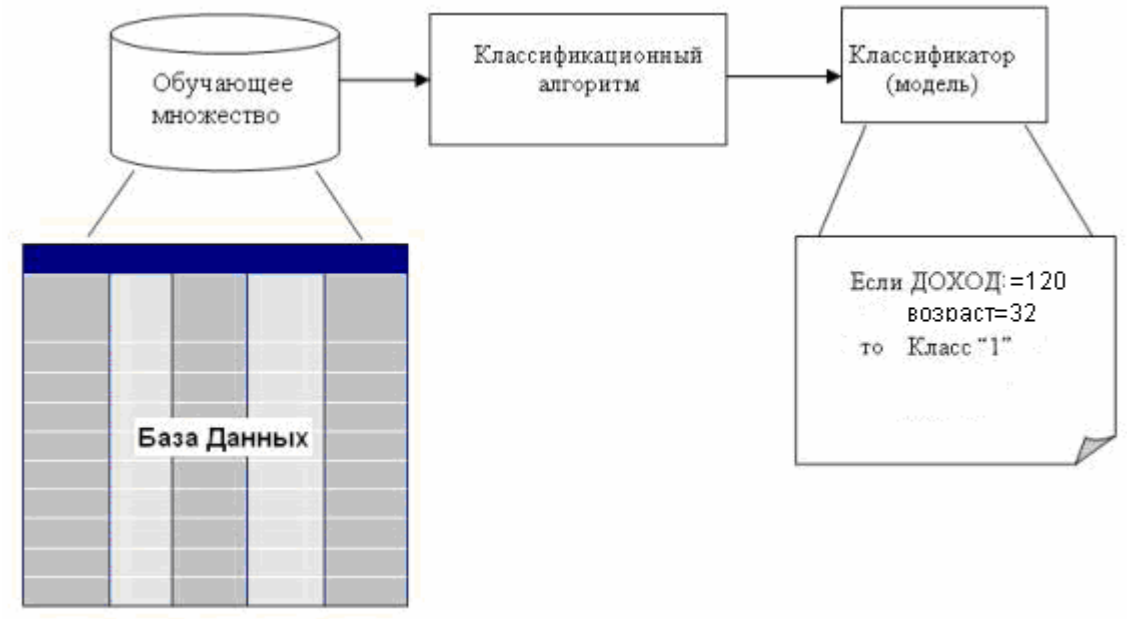


Рис. Процесс классификации.

Методы, применяемые для решения задач классификации

Для классификации используются различные методы. Основные из них:

- классификация с помощью деревьев решений;
- байесовская (наивная) классификация;
- классификация при помощи искусственных нейронных сетей;
- классификация методом опорных векторов;

- статистические методы, в частности, линейная регрессия;
- классификация при помощи метода ближайшего соседа;
- классификация при помощи генетических алгоритмов.

Схематическое решение задачи классификации некоторыми методами (при помощи линейной регрессии, деревьев решений и нейронных сетей) приведены на рисунках:

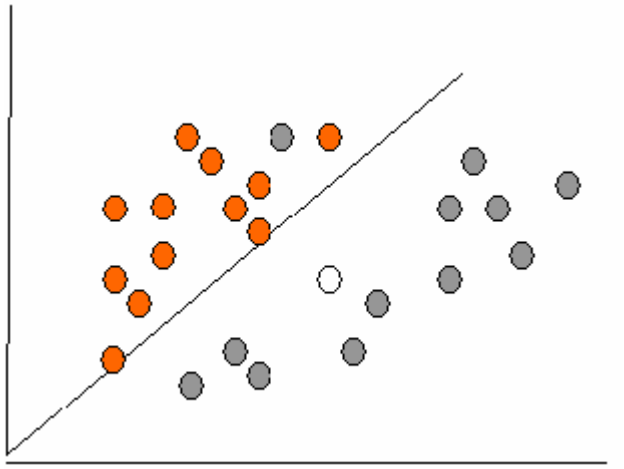


Рис. Решение задачи классификации методом линейной регрессии

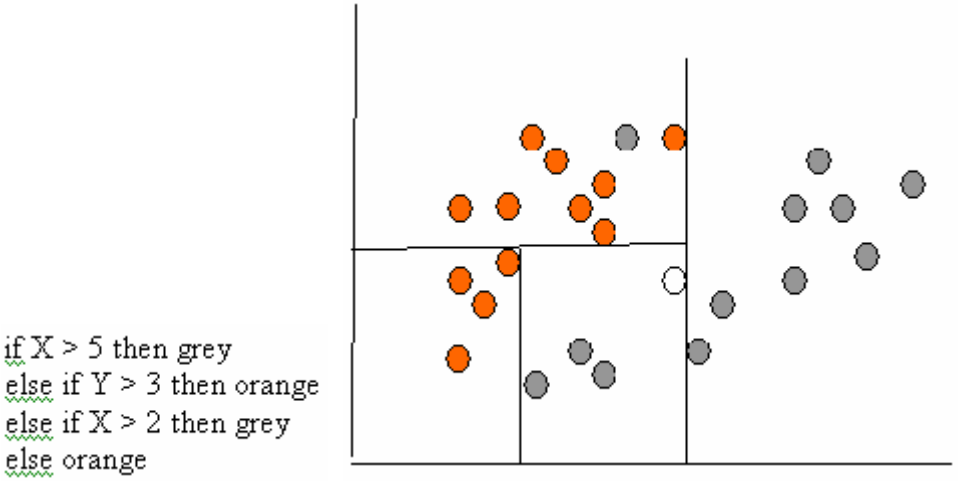


Рис. Решение задачи классификации методом деревьев решений

Рис. Решение задачи классификации методом нейронных сетей

Оценивание классификационных методов

Оценивание методов следует проводить, исходя из следующих характеристик: скорость, робастность, интерпретируемость, надежность.

Скорость характеризует время, которое требуется на создание модели и ее использование.

Робастность, т.е. устойчивость к каким-либо нарушениям исходных предпосылок, означает возможность работы с зашумленными данными и пропущенными значениями в данных.

Интерпретируемость обеспечивает возможность понимания модели аналитиком.

Надежность методов классификации предусматривает возможность работы этих методов при наличии в наборе данных шумов и выбросов.

Препроцессорная подготовка автоматической классификации (распознавания).

Распознавание образов (а часто говорят - объектов, сигналов, ситуаций, явлений или процессов) - самая распространенная задача, которую человеку приходится решать практически ежесекундно от первого до последнего дня своего существования. Для этого он использует огромные ресурсы своего мозга, которые оцениваются таким показателем как число нейронов, равное 10^{10}.

Похожие проблемы наблюдаются в биологии, в живой природе, а иногда даже в неживой. Кроме того, распознавание постоянно встречается в технике. А если это так, то, очевидно, следует считать механизм распознавания всеобъемлющим.

С более общих позиций можно утверждать, что в повседневной деятельности человек постоянно сталкивается с задачами, связанными с принятием решений, обусловленных непрерывно меняющейся окружающей обстановкой. В этом процессе принимают участие: органы чувств, с помощью которых человек воспринимает информацию извне; центральная нервная система, осуществляющая отбор, переработку информации и принятие решений; двигательные органы, реализующие принятое решение. Но в основе решений этих задач лежит распознавание образов.

Рассмотрим некоторые особенности всепроникающего механизма распознавания образов в природе и обществе.

1. Вы легко узнаете издалека своего знакомого. Обратите внимание на слабую зависимость результатов распознавания от расстояния.
2. Помимо “чистого” распознавания в этом процессе присутствуют различные действия, но при этом любому действию предшествует распознавание. А любое выполненное действие влечет за собой новый этап распознавательной деятельности.
3. Среди действий, выполняемых в процессе распознавания, встречаются и управленческие.
4. Распознавание не всегда одномоментно. Оно может быть распределено во времени, связано с определённой последовательностью действий.
5. Распознавание требует осмысливания соответствующего механизма, последовательности действий.

6. Каждый этап действий, описанных в последовательности - это действия на основе знаний, хранящихся в памяти.

Рассмотрим пример из области экономики. Руководитель экономического региона по экономическим показателям хозяйственной деятельности обнаруживает (**распознает**) ухудшение продовольственного обеспечения области, города и т.п.

Обратившись к другой группе экономических показателей, он **распознает**, что лежит в основе такого нежелательного явления (например, отсутствие горючего для автотранспорта). В итоге **принимается решение** о дополнительных договорах на бензин или дизельное топливо с поставщиками или, **найдя новых** поставщиков, **организует отправку** железнодорожного состава цистерн или автозаправщиков для доставки и т.д.

До настоящего времени полные представления о способностях живых организмов в распознавании многих явлений и объектов отсутствуют. В то же время, создавая технические системы, способные заменить его, человек высказывает гипотезы, продвигающие его к знанию распознающей деятельности в природе, что позволяет ему успешно решать стоящие задачи и научить пониманию того, что лежит в основе современных гипотез распознавательной деятельности и как на этой основе упомянутые задачи решаются.

Понятия “объект”, “образ”, “класс”.

Класс задач распознавания связан с понятием “образа”: в термине “pattern recognition” термин “pattern”, кроме значения “образ”, имеет еще значение “модель”, стиль”, “режим”, “закономерность”, “образ действия”.

В современном распознавании “образ” - это некоторое структурированное приближенное описание (эскиз, частичная определенность) изучаемого объекта, явления или процесса.

То есть, частичная определенность описания является принципиальным свойством образа.

Основное назначение описаний (образов) - это их использование в процессе установления соответствия объектов, то есть при доказательстве их идентичности, аналогичности, подобия, сходства и т.п., которое осуществляется путем сравнения (сопоставления).

Два образа считаются подобными, если удается установить их соответствие. Можно, в частности, считать, что имеет место соответствие, если достигнута их идентичность.

Сопоставление образов представляет собой основную задачу распознавания и играет существенную роль в информатике в целом. Эта задача возникает, в частности, в различных разделах искусственного интеллекта, например в понимании естественного языка компьютером, символьной обработке алгебраических выражений, экспертных системах, преобразовании и синтезе программ ЭВМ.

В различных задачах образу придается различный смысл, который характеризуется двумя составляющими: понятием, и расширяющими и конкретизирующими это понятие смысловыми ореолами. Придаваемый образу смысл определяется часто тем, какие характеристики объекта входят в описание образа, какой аппарат используется для представления этих характеристик. Именно отсюда и можно понять, почему образ является приближенным описанием объекта. Чем большее число свойств и качеств объекта отражено на принятом языке в образе рассматриваемого объекта, тем полнее это описание, тем полнее этот образ характеризует описываемый объект.

Принимают решение о распознавании на основе отождествления совокупности конкретных значений характеристик объектов или явлений не просто друг с другом, а обычно с некоторым классом, в который объединяются объекты или явления, имеющие общие свойства.

Таким образом, классы - это объединения объектов (явлений), отличающиеся общими свойствами, интересующими человека.

В самых общих чертах распознавание можно определить как соотнесение объектов или явлений на основе анализа их характеристик, представляющих образы этих объектов, с одним из нескольких, заранее определенных классов.

Практические реализации методов распознавания, носят название систем распознавания (СР), или "распознающих систем".

Широкий круг задач, возлагаемых на распознающие системы, определяется определением самого понятия "распознавание" и включает **выяснение по разнородной, часто неполной, нечеткой, искаженной и косвенной информации факта, обладают ли изучаемые объекты, явления, процессы, ситуации фиксированным конечным набором свойств, позволяющим отнести их к определенному классу.**

Сюда входят как непосредственно задачи распознавания и классификации, так и задачи предсказания (прогнозирования), т.е. задачи, в результате решения которых на основе распознавания требуется выяснить, в какой области из конечного числа областей будут находиться некоторые процессы через определенный промежуток времени.

К задачам распознавания относятся задачи:

- технической и медицинской диагностики,
- геологического прогнозирования,
- прогнозирования (предсказание) свойств химических соединений,
- распознавания свойств динамических и статических объектов в сложной фоновой обстановке и при наличии активных и пассивных помех,
- прогнозирования урожая,
- обнаружения лесных пожаров,
- управления производственными процессами.

Рассмотрим некоторые из них:

1) Системы технической диагностики.

Их внедрение - важнейший фактор повышения эффективности использования машин и технологического оборудования, резкого сокращения расходов на эксплуатацию.

Исторически сложившаяся тенденция усложнения, а значит удорожания машин постоянно увеличивает затраты на эксплуатацию. Выход - переход к системам технической диагностики (распознавания состояния машин), например, безразборный поиск неисправностей. В результате вместо планово-предупредительного ремонта - ремонт по фактической необходимости.

Например, в инструкции по эксплуатации автомобиля предусмотрены плановые технические обслуживания через 500 км, 1000 км, 2000 км и т.д. При использовании систем распознавания состояний от плановых ТО можно отказаться заменив их обслуживанием отдельных узлов и систем по необходимости.

2) Медицинская диагностика.

Автоматизированные системы диагностики в медицине (рассчитывать только на память врача во всех ситуациях очень трудно. Если функцию памяти отдать компьютеру) – это будет путь увеличения:

- широты и глубины охвата симптомов;
- оперативности; (компьютер обеспечит почти мгновенный результат)

- достоверности (диагноз компьютера не зависит от внешних факторов, как это случается с человеком)

3) Сельское хозяйство.

Области применения здесь:

- распознавание размеров урожая по данным космических наблюдений;
- уменьшение ручного труда при сортировке плодов по форме, цвету и размерам и т.п.

4) Военное дело.

Сложные системы вооружения:

- автоматический функциональный контроль технического состояния систем и ввод резервирующих;
- роботы, обслуживающие фазированные антенные решетки радаров.

На основе рассмотренного можно уже ответить на вопрос, что же необходимо системе распознавания для работы.

Физически системы распознавания -

- это и вычислительная машина как один составляющий элемент СР;
- это и такие часто более дорогостоящие технические средства, как средства обнаружения распознаваемых объектов (например, патологических изменений того или иного органа человека);
- это и средства измерений параметров обнаруженных объектов (без них не получить признаков распознавания);
- это и математическое обеспечение, в составе которого:
  - методы и алгоритмы обработки измерительной информации;
  - методы и алгоритмы определения признаков распознавания;
  - методы и алгоритмы непосредственно распознавания объектов, явлений , процессов (построения решающих правил отнесения объектов к тому или иному классу);
  - методы и алгоритмы в некотором смысле оптимального управления процессом распознавания;
- наконец, для больших систем это и коллектив подготовленных специалистов обеспечивающих жизненный цикл существования системы.

Последняя составляющая на первый взгляд не имеет отношения к системе. Однако без коллектива подготовленных специалистов трудно обойтись в больших системах, решения которых чрезвычайно ответственны. В таких системах оценка эффективности - это показатель, которым пользуются с момента создания СР и до конца ее существования. При этом пользуются этим показателем специалисты, а не система.

Таким образом, СР - сложная динамическая система, состоящая в общем случае из коллектива подготовленных специалистов и совокупности технических средств получения и переработки информации, обеспечивающих на основе специально сконструированных алгоритмов решение задачи классификации соответствующих объектов, явлений или процессов.

Технология решения задач распознавания.

Несмотря на многообразие и особенности приложений, в которых решаются задачи распознавания, в них есть много общего, не зависящего от отраслевой специфики.

Вот почему для выработки методических подходов технологии распознавания имеет смысл выделять общие повторяющиеся приемы.

В приложении 2 приведен пример систем распознавания, используемых для решения двух задач: «Распознавание стороной А самолётов стороны В», и «распознавание заболеваний сердца».

Эти два примера показали, что подходы к построению систем распознавания практически ничем не отличаются, несмотря на специфику самих создаваемых систем.

В результате мы получили общие представления о последовательности решения и составляющих задачи создания и использования системы распознавания:

- совокупность объектов или явлений подразделяется на ряд классов (говорят: назначается алфавит классов);
- разрабатывается совокупность признаков (говорят: словарь);
- на языке словаря признаков описывается каждый класс;
- выбираются и (или) создаются средства определения признаков;
- на вычислительных средствах реализуется алгоритм сопоставления апостериорных и априорных данных и принимается решение о результатах распознавания.

В то же время, несмотря на полученное определение последовательности действий, проведенное рассмотрение не позволяет ответить на следующие вопросы:

- как лучше производить разбиение объектов (самолеты, заболевания и пр.) по классам;
- как накапливать и обрабатывать априорную информацию;
- из каких соображений выбирать признаки;
- как описывать классы на языке признаков;
- на основе каких методов сравнивать априорную и апостериорную информацию;

Решение задач распознавания должно начинаться с самого начала проведения исследования в виде модели - прообраза будущей системы распознавания. Без такой модели мы не сможем выбрать ни набор классов, ни перечень признаков, ни средства измерений их, ни решающие правила, обеспечивающие в комплексе, во взаимосвязи требуемое качество решений о принадлежности. Это обусловлено тем, что полная информация в начале исследования всегда отсутствует и без экспериментальной отработки всего процесса принятия решений не всегда ясно, какая информация может вообще потребоваться.

Итак, главные выводы:

1. Задачи, решаемые в процессе исследования систем распознавания, инвариантны относительно предметной области, имеют много общего, основываются на едином методологическом подходе.

2. Технология распознавания должна уточняться проведением последовательных приближений её структуры на ее математической модели по мере накопления необходимой информации.

3. Технология распознавания начинается не с применения какого-либо технического средства, а с анализа постановки задачи, с изучения смысла и целей проводимого

исследования, формулировки модели изучаемого объекта, процесса или явления. Эта работа представляет собой препроцессорную подготовку проводимого исследования.

Один из вариантов такой препроцессорной подготовки предусматривает решение таких задач:

1. Определение полного перечня признаков, характеризующих объекты;
2. Составление априорного алфавита классов;
3. Разработка априорного словаря признаков распознавания;
4. Описание классов априорного алфавита на языке априорного словаря признаков;
5. Выбор алгоритма классификации;
6. Определение рабочего алфавита классов.

Определение полного перечня признаков, характеризующих объекты

При определении полного перечня признаков (параметров), характеризующих объекты или явления, для которых данная система разрабатывается, главное - найти все признаки, характеризующие существо распознаваемых объектов (явлений). Любые ограничения, любая неполнота, приводят к ошибкам или полной невозможности правильной классификации объектов (явлений).

Можем себе представить такую неполноту в уже рассмотренной нами задаче распознавания самолетов как использование одного признака - потолок высоты полета самолетов. В результате - бомбардировщики не удастся отличать от истребителей (при создании бомбардировщиков стремятся к обеспечению максимально возможной высоты полета, а при создании истребителей добиваются, чтобы они могли уничтожать бомбардировщики).

Реально даже целая группа признаков может оказаться неэффективной.

Поэтому для решения 1-ой задачи создания СР необходимо найти все возможные признаки, описывающие объекты распознавания, с тем, чтобы при оценке эффективности решений системы не возвращаться к этой задаче, обнаружив ограниченность выбранных признаков на последующих этапах разработки.

Но чтобы назначать признаки распознавания, необходимо, во-первых, понять, что не существует способов их автоматической генерации. На сегодня это под силу только человеку. Поэтому говорят, что выбор признаков - эвристическая операция.

Во-вторых, выбор признаков можно осуществлять, имея представление об их общих свойствах. С этих позиций достаточно принять, что признаки могут подразделяться на:

- детерминированные;
- вероятностные;
- логические;
- структурные.

Таким образом, при определении полного перечня признаков (параметров), характеризующих объекты или явления, для которых данная система разрабатывается можно сделать такие выводы:

1) Выбор, назначение признаков распознавания - эвристическая операция, зависящая от творчества, изобретательности разработчика.

2) Состав признаков, выбираемых на этом этапе, должен быть как можно более разносторонним и полным, независимым от того, можно или нельзя эти признаки получить.

3) Выбор признаков должен осуществляться в группах детерминированных, вероятностных, логических и структурных.

Составление априорного алфавита классов

Первоначальная классификация объектов (явлений), подлежащих распознаванию заключается в необходимости выбрать (назначить) классы объектов (явлений) распознавания. Решение этой задачи осуществляется чаще всего эвристически, как и выбор признаков распознавания, а логика ее решения следующая:

1-е - определяется, какие решения могут приниматься по результатам распознавания либо человеком, либо автоматической системой управления объектом (цель распознавания).

2-е - на основе определенной выше цели формулируются требования к системе распознавания, позволяющие выбрать принцип классификации.

3-е - составляется априорный алфавит классов объектов (явлений).

Заметим, что, кроме ситуации, предложенной в рассмотренной задаче, возможны и другие, когда количество классов, по которым надежно распознаются некоторые объекты (явления), заранее неизвестно и должно определяться самой системой распознавания. Эта задача называется задачей кластеризации, в которой можно отказаться уже от эвристического подхода. Однако решение здесь достигается при выборе некоторых общих правил кластеризации, которые задает разработчик системы.

Разработка априорного словаря признаков распознавания.

Решая задачу №1, мы должны были найти все возможные признаки распознавания заданных объектов или явлений. Точно также при решении задачи №2 определился состав классов.

Теперь, располагая соответствующим перечнем и априорным алфавитом классов, необходимо провести анализ возможностей измерения признаков или расчета их по данным измерений, выбрать те из них, которые обеспечиваются измерениями, а также в случае необходимости разработать предложения и создать новые средства измерений для обеспечения требуемой эффективности распознавания.

Таким образом, главное содержание рассматриваемой задачи - создание словаря, обеспечиваемого реально возможными измерениями.

Однако, хороший или плохой набор признаков распознавания получился в результате указанных действий разработчика, можно понять, выполнив испытания модели системы распознавания и оценив эффективность распознавания. Только методом последовательных приближений удается добиться выбора словаря признаков, обеспечивающего желаемое качество решений.

Выходом из создавшегося положения является возможность создания на данном этапе математической модели системы. Математические модели СР и используются для реализации указанных последовательных приближений.

Описание классов априорного алфавита на языке априорного словаря признаков.

Априорное описание классов - наиболее трудоемкая из задач в процессе создания системы распознавания, требующая глубокого изучения свойств объектов распознавания, а также и наиболее творческая задача.

В рамках этой задачи необходимо каждому классу поставить в соответствие числовые параметры детерминированных и вероятностных признаков, значения логических признаков и предложения, составленные из структурных признаков-примитивов.

Значения этих параметров описаний можно получить из совокупности следующих работ и действий:

- специально поставленные экспериментальные работы или - экспериментальные наблюдения;
- результаты обработки экспериментальных данных;
- математические расчеты;
- результаты математического моделирования;
- извлечения из литературных источников.

Выбор алгоритма классификации, обеспечивающего отнесение распознаваемого объекта или явления к соответствующему классу.

Непосредственное решение задачи распознавания на основе использования словаря признаков и алфавита классов объектов или явлений фактически заключается в разбиении пространства значений признаков распознавания на области D1,D2,...,Dn, соответствующие классам W1,W2,...,Wn (вспоминаем определение “образа”).

Указанное разбиение должно быть выполнено таким образом, чтобы обеспечивались минимальные значения ошибок отнесения классифицируемых объектов или явлений к “чужим” классам.

Результатом такой операции является отнесение объекта, имеющего набор признаков X1,X2,....,Xn (точка в n-мерном пространстве), к классу Wi, если указанная точка лежит в соответствующей классу области признаков - Di.

Разбиение пространства признаков можно представлять как построение разделяющих функций fi(x1,x2,....,xn) между множествами (областями) признаков Di, принадлежащим разным классам.

В алгоритмах распознавания, использующих детерминированные признаки в качестве меры близости, используется среднеквадратическое расстояние между данным объектом w и совокупностью объектов (w1,w2,....,wn), представляющих (описывающих) каждый класс.

При этом в качестве методов измерений расстояния между объектами d(w,wg) могут использоваться любые методы (творческий процесс здесь не ограничивается).

В алгоритме распознавания, использующем детерминированные признаки можно учитывать и их веса Vj (устанавливать степень доверия или важности).

В алгоритмах распознавания, использующих вероятностные признаки, в качестве меры близости используется риск, связанный с решением о принадлежности объекта к классу Wi, где i - номер класса. (i=1,2,..,m.).

Для алгоритмов, основанных на логических признаках, понятие “мера близости” не имеет смысла. Вспомним упрощенный пример, рассмотренный нами для логических признаков заболеваний (простой простуды и ангины).

Для алгоритмов, основанных на структурных (лингвистических) признаках, понятие “меры близости” более специфично.

С учетом того, что каждый класс описывается совокупностью предложений, характеризующих структурные особенности объектов соответствующих классов,

распознавание неизвестного объекта осуществляется идентификацией предложения, описывающего этот объект, с одним из предложений в составе описания какого-либо класса.

При этом идентификация может подразумевать наибольшее сходство предложения, описывающего распознаваемый объект с предложениями из наборов описания каждого класса.

Создание СР осуществляется последовательными приближениями по мере получения дополнительной информации. В этом ряду последовательных приближений главную роль играют признаки распознавания. От эффективности их набора зависит, эффективность системы в целом. В процессе совершенствования системы указанный набор пополняется, неэффективные признаки исключаются.

Поэтому одной из задач распознавания должна быть и задача перехода от априорного словаря признаков к рабочему. То же касается и априорного алфавита классов.

Определение рабочего алфавита классов и рабочего словаря признаков системы распознавания.

Настоящая задача на уровне разработки, прошедшей этапы решения задач 1 - 5, по крайней мере уже может быть поставлена, так как в результате выполнения предшествующих задач создана система распознавания первого приближения (априорный алфавит классов и априорный словарь признаков, выбран алгоритм распознавания).

Суть стоящей задачи - разработка такого (рабочего) алфавита классов и такого (рабочего) словаря признаков, которые обеспечили бы максимальное значение показателя эффективности распознавания. То есть, из априорного словаря мы должны выбрать признаки, позволяющие при всех имеющихся ограничениях на их получение (измерение) доставить максимум вероятности правильной классификации объектов (явлений) и (или) минимальные вероятности ошибочных классификаций создаваемой системой. Такой выбор не может не предполагать оценку указанных показателей до того, как создана система.

Уточнение принципов классификации

Классификация - это распределение предметов, явлений по классам, отделам, разрядам в зависимости от их общих свойств.

В основе классификации лежат определенные принципы. Например, помимо приведенной в лекции 5, существует классификация систем распознавания на основе следующих принципов:

1.Однородность информации для описания распознаваемых объектов или явлений.

2.Способ получения апостериорной информации.

3.Количество первоначальной априорной информации.

4.Характер информации о признаках распознавания.

Кластерный анализ.

Наиболее распространенные алгоритмы кластерного анализа строятся на основе **метода Джонсона иерархического группирования**.

Суть иерархической кластеризации состоит в последовательном объединении меньших кластеров в большие или разделении больших кластеров на меньшие.

Этот метод можно использовать как в эвклидовом, так и в неэвклидовом пространствах, что позволяет пользоваться им при измерении свойств по различным

типам шкал (номинальная, порядковая, абсолютная). Для этого лишь необходимо выбрать соответствующую меру близости.

Сущность метода Джонсона заключается в следующем:

1. Определяется матрица H, элементами которой служат расстояния между парами объектов (парами экспериментальных точек в многомерном пространстве свойств):

$$h(ik) = h(a(i),a(k))$$

2. Проводится группирование, в котором каждый объект относится к классу, содержащему только его самого. Расстояние между любыми двумя скоплениями C(0i) и C(0k) в этом случае равно

$$h(C(0i),C(0k)) = h(ik)$$

3. Предположим, что проведено группирование и определена соответствующая матрица H с элементами h(ik). Пусть b и g - индексы двух ближайших скоплений, т.е. h(bg) - наименьший элемент матрицы H. Тогда группирование C(t) получается из C(t-1) объединением элементов b и g в одну группу [b,g].

4. Перераспределяем матрицу H, вычеркивая все элементы, относящиеся к b и g, и добавляя расстояния до нового скопления [b,g], определяемые для [b,g] и некоторого третьего элемента f по правилу максимума

$$h[(b,g),f] = \max[h(b,f),h(g,f)]$$

или правилу минимума

$$h[(b,g),f] = \min[h(b,f),h(g,f)].$$

5. Повторяем шаги 3 и 4 до тех пор, пока все начальные объекты (экспериментальные точки) не объединятся в одну группу.

Условимся для каждого скопления точек под его диаметром понимать наибольшее расстояние между любыми двумя его точками.

Если применяется правило максимума, то на каждом этапе (шаг 3) будут формироваться группы наименьшего диаметра.

Правило минимума приведет к такой иерархии групп, при которой минимизируется расстояние между соседними точками цепи в пределах одной группы.

Эти правила в литературе называются алгоритмами "дальнего соседа" и "ближайшего соседа". Проиллюстррировать их применение можно на примере рис.1.

Для ситуации, изображенной на рис.1, на первом шаге образуется кластер из точек "a" и "в", на втором шаге - из точек "e и "d". Результат третьего шага будет зависеть от того, какой алгоритм - дальнего или ближайшего соседа будет реализован. При алгоритме дальнего соседа расстояние от точки "c" до кластера "ав" будет равно расстоянию между точками "c" и "a" (т.е. "ac"); а расстояние от точки "c" до кластера "de" будет равно "ce", причем, "ce">"ac" и точка "c" будет отнесена к кластеру "ав". Таким образом, будет сформирован кластер "авс", в котором точки расположены наиболее плотно.

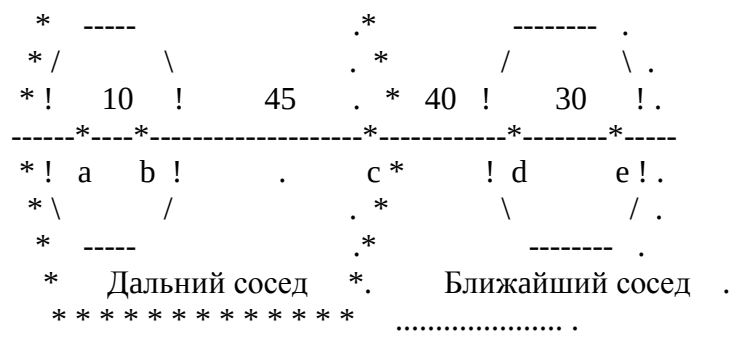


Рис.1.

При алгоритме ближнего соседа расстояние от точки "с" до кластера "ав" будет равно "вс" тогда, как до кластера "de" - "cd", причем, "вс">"cd", и точка "с" будет отнесена к кластеру "de". Образованный при этом кластер "cde" будет более размытым, чем кластер "авс", но для него характерно более близкое расположение точек друг к другу в пределах одной группы.

Пример кластерного анализа.

Допустим, мы имеем набор данных A, состоящий из 14-ти примеров, у которых имеется по два признака X и Y. Данные по ним приведены в таблице 13.1.

Таблица 13.1. Набор данных A

№ примера	признак X	признак Y
1	27	19
2	11	46
3	25	15
4	36	27
5	35	25
6	10	43
7	11	44
8	36	24
9	26	14
10	26	14
11	9	45
12	33	23
13	27	16
14	10	47

Данные в табличной форме не носят информативный характер. Представим переменные X и Y в виде диаграммы рассеивания, изображенной на рис. 13.1.

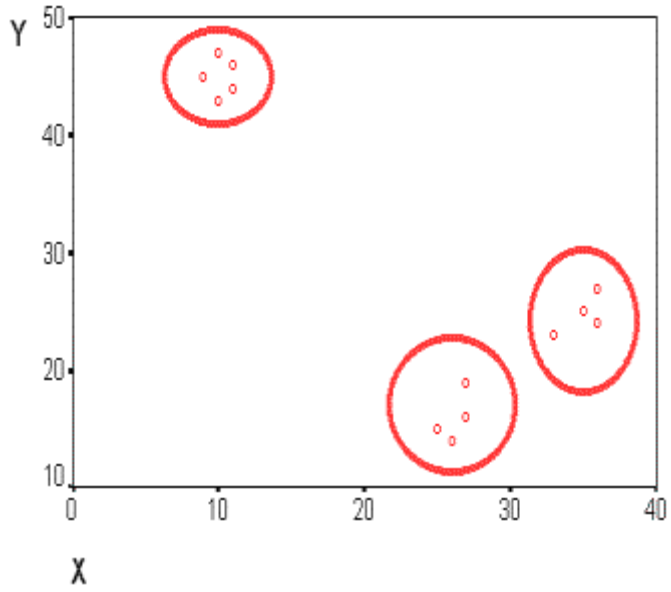


Рис. 13.1. Диаграмма рассеивания переменных X и Y

На рисунке мы видим несколько групп "похожих" примеров. Примеры (объекты), которые по значениям X и Y "похожи" друг на друга, принадлежат к одной группе (кластеру); объекты из разных кластеров не похожи друг на друга.

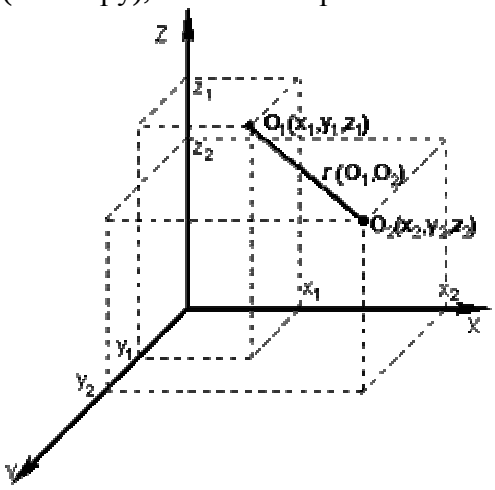


Рис. 13.2. Расстояние между двумя точками в пространстве трех измерений

Таблица 13.2. Порядок агломерации

	Cluster Combined		Coefficients
	Cluster 1	Cluster 2	
1	9	10	,000
2	2	14	1,461E-02
3	3	9	1,461E-02
4	5	8	1,461E-02
5	6	7	1,461E-02
6	3	13	3,490E-02
7	2	11	3,651E-02

8	4	5	4,144E-02
9	2	6	5,118E-02
10	4	12	,105
11	1	3	,120
12	1	4	1,217
13	1	2	7,516

Так, в колонке Cluster Combined можно увидеть порядок объединения в кластеры: на первом шаге были объединены наблюдения 9 и 10, они образовывают кластер под номером 9, кластер 10 в обзорной таблице больше не появляется. На следующем шаге происходит объединение кластеров 2 и 14, далее 3 и 9, и т.д.

В колонке Coefficients приведено количество кластеров, которое следовало бы считать оптимальным; под значением этого показателя подразумевается расстояние между двумя кластерами, определенное на основании выбранной *меры расстояния*. В нашем случае это квадрат *евклидова расстояния*, определенный с использованием стандартизированных значений. Процедура *стандартизации* используется для исключения вероятности того, что классификацию будут определять переменные, имеющие наибольший разброс значений.

Метод K средних

Общая логика

Этот метод кластеризации существенно отличается от таких агломеративных методов, как Объединение (древовидная кластеризация) и Двувходовое объединение. Предположим, вы уже имеете гипотезы относительно числа кластеров (по наблюдениям или по переменным). Вы можете указать системе образовать ровно три кластера так, чтобы они были настолько различны, насколько это возможно. Это именно тот тип задач, которые решает алгоритм метода K средних. В общем случае метод K средних строит ровно K различных кластеров, расположенных на возможно больших расстояниях друг от друга.

Пример

В примере с физическим состоянием (см. Двувходовое объединение), медицинский исследователь может иметь "подозрение" из своего клинического опыта, что его пациенты в основном попадают в три различные категории. Далее он может захотеть узнать, может ли его интуиция быть подтверждена численно, то есть, в самом ли деле кластерный анализ K средних даст три кластера пациентов, как ожидалось? Если это так, то средние различных мер физических параметров для каждого кластера будут давать количественный способ представления гипотез исследователя (например, пациенты в кластере 1 имеют высокий параметр 1, меньший параметр 2 и т.д.).

Вычисления

С вычислительной точки зрения вы можете рассматривать этот метод, как дисперсионный анализ (см. Дисперсионный анализ) "наоборот". Программа начинает с K случайно выбранных кластеров, а затем изменяет принадлежность объектов к ним, чтобы: (1) - минимизировать изменчивость внутри кластеров, и (2) - максимизировать изменчивость

между кластерами. Данный способ аналогичен методу "дисперсионный анализ (ANOVA) наоборот" в том смысле, что критерий значимости в дисперсионном анализе сравнивает межгрупповую изменчивость с внутригрупповой при проверке гипотезы о том, что средние в группах отличаются друг от друга. В кластеризации методом K средних программа перемещает объекты (т.е. наблюдения) из одних групп (кластеров) в другие для того, чтобы получить наиболее значимый результат при проведении дисперсионного анализа (ANOVA).

Интерпретация результатов

Обычно, когда результаты кластерного анализа методом K средних получены, можно рассчитать средние для каждого кластера по каждому измерению, чтобы оценить, насколько кластеры различаются друг от друга. В идеале вы должны получить сильно различающиеся средние для большинства, если не для всех измерений, используемых в анализе. Значения F-статистики, полученные для каждого измерения, являются другим индикатором того, насколько хорошо соответствующее измерение дискриминирует кластеры.

Литература.

1. Чубукова И.А. «Data Mining», Интуит.ru.
2. Белозерский Л.А. «Основы построения систем распознавания образов», курс лекций, Донецкий государственный институт искусственного интеллекта, 1997.
3. Ю.П.Маслобоев "Свойства и параметры нейронной сети как объекта MATLAB" http://matlab.exponenta.ru/neuralnetwork/book1/index.php

Лекция 11. Нечёткая логика.

Введение в теорию нечётких множеств.

Одним из примечательных свойств человеческого интеллекта является способность принимать правильные решения в обстановке неполной и нечеткой информации. Поэтому можно утверждать, что построение моделей приближенных рассуждений человека и их использование в перспективных компьютерных системах представляет одно из важнейших направлений современных информационных технологий.

Теория нечетких множеств (fuzzy sets theory) ведет свое начало с 1965г., когда профессор Лотфи Заде (Lotfi Zadeh) из университета Беркли опубликовал основополагающую работу “Fuzzy Sets” в журнале “Information and Control”. Прилагательное "fuzzy", которое можно перевести на русский как нечеткий, размытый, ворсистый, пушистый, введено в название новой теории с целью дистанцирования от традиционной четкой математики и аристотелевой логики, оперирующих с четкими понятиями: “принадлежит - не принадлежит”, “истина - ложь”. Концепция нечеткого множества зародилась у Заде “как неудовлетворенность математическими методами классической теории систем, которая вынуждала добиваться искусственной точности, неуместной во многих системах реального мира, особенно в так называемых гуманистических системах, включающих людей”.

Математическая логика оперирует понятиями логических переменных. Каждая логическая переменная может находиться в одном из двух состояний: True и False. Заде предложил ввести понятие нечёткой логической переменной, в которой нечёткось

оценивается числом от 0 до 1. Тогда понятие “мужчина среднего роста” можно определить в виде множества A:

A = 0/155+0.1/160 + 0.3/165 + 0.8/170 +1/175 +1/180 + 0.5/185 +0/180.

Такое множество называется нечётким множеством. Коэффициенты, определяющие значения цифр данного нечёткого множества, называются ‘функцией принадлежности’ (membership function).

Кроме логических переменных в нечётких множествах могут встречаться *лингвистические переменные* (linguistic variable), т.е. переменные, значениями которых могут быть слова или словосочетания некоторого естественного или искусственного языка.

Множество всех возможных значений лингвистической переменной называется *Терм–множеством (term set).*

Термом (term) называется любой элемент терм–множества. В теории нечетких множеств терм формализуется нечетким множеством с помощью функции принадлежности.

Пример. Рассмотрим переменную "*скорость автомобиля*", которая оценивается по шкале "*низкая*", "*средняя*", "*высокая*" и "*очень высокая*".

В этом примере лингвистической переменной является "*скорость автомобиля*", термами - лингвистические оценки "*низкая*", "*средняя*", "*высокая*" и "*очень высокая*", которые и составляют терм–множество.

Дефаззификацией (defuzzification) называется процедура преобразования нечеткого множества в четкое число.

В теории нечетких множеств процедура дефаззификации аналогична нахождения характеристик положения (математического ожидания, моды, медианы) случайных величин в теории вероятности. Простейшим способом выполнения процедуры дефаззификации является выбор четкого числа, соответствующего максимуму функции принадлежности. Однако пригодность этого способа ограничивается лишь одноэкстремальными функциями принадлежности. Для многоэкстремальных функций принадлежности в Fuzzy Logic Toolbox запрограммированы такие методы дефаззификации:

Centroid - центр тяжести;

Bisector - медиана;

LOM (Largest Of Maximums) - наибольший из максимумов;

SOM (Smallest Of Maximums) - наименьший из максимумов;

Mom (Mean Of Maximums) - центр максимумов.

Пример. Провести дефаззификацию нечеткого множества "мужчина среднего роста" из примера 1 по методу центра тяжести.

Решение:

$$a = \frac{0\cdot 155 + 0.1\cdot 160 + 0.3\cdot 165 + 0.8\cdot 170 + 1\cdot 175 + 1\cdot 180 + 0.5\cdot 185 + 0\cdot 190}{0 + 0.1 + 0.3 + 0.8 + 1 + 1 + 0.5 + 0} = 175.4$$

Нечеткой базой знаний (fuzzy knowledge base) о влиянии факторов $X = \{x_1, x_2, \ldots, x_n\}$ на значение параметра y называется совокупность логических высказываний типа:

ЕСЛИ $\left(x_1 = a_1^{j1}\right)$ И $\left(x_2 = a_2^{j1}\right)$ И…И $\left(x_n = a_n^{j1}\right)$

ИЛИ $\left(x_1 = a_1^{j2}\right)$ И $\left(x_2 = a_2^{j2}\right)$ И…И $\left(x_n = a_n^{j2}\right)$…

ИЛИ $\left(x_1 = a_1^{jk_j}\right)$И$\left(x_2 = a_2^{jk_j}\right)$И…И$\left(x_n = a_n^{jk_j}\right)$,

ТО $y = d_j$, для всех $j = \overline{1,m}$,

где a_i^{jp} - нечеткий терм, которым оценивается переменная x_i в строчке с номером jp ($p = \overline{1,k_j}$);

k_j - количество строчек-конъюнкций, в которых выход y оценивается нечетким термом d_j, $j = \overline{1,m}$;

m - количество термов, используемых для лингвистической оценки выходного параметра y.

Нечетким логическим выводом (fuzzy logic inference) называется апроксимация зависимости $y = f(x_1, x_2, \ldots, x_n)$ с помощью нечеткой базы знаний и операций над нечеткими множествами.

Операции над нечеткими множествами

Определения нечетких теоретико-множественных операций объединения, пересечения и дополнения могут быть обобщены из обычной теории множеств. В отличие от обычных множеств, в теории нечетких множеств степень принадлежности не ограничена лишь бинарными значениями 0 и 1 - она может принимать значения из интервала [0, 1]. Поэтому, нечеткие теоретико-множественные операции могут быть определены по-разному. Ясно, что выполнение нечетких операций объединения, пересечения и дополнения над не нечеткими множествами должно дать такие же результаты, как и при использование обычных канторовских теоретико-множественных операций.

Нечеткая логика это обобщение традиционной аристотелевой логики на случай, когда истинность рассматривается как лингвистическая переменная, принимающая значения типа: "очень истинно", "более-менее истинно", "не очень ложно" и т.п. Указанные лингвистические значения представляются нечеткими множествами.

Лингвистической называется переменная, принимающая значения из множества слов или словосочетаний некоторого естественного или искусственного языка. Множество допустимых значений лингвистической переменной называется терм-множеством. Задание значения переменной словами, без использования чисел, для человека более естественно. Ежедневно мы принимаем решения на основе лингвистической информации типа: "очень высокая температура"; "длительная поездка"; "быстрый ответ"; "красивый букет"; "гармоничный вкус" и т.п. Психологи установили, что в человеческом мозге почти вся числовая информация вербально перекодируется и хранится в виде лингвистических термов. Понятие лингвистической переменной играет важную роль в нечетком логическом выводе и в принятии решений на основе приближенных рассуждений. Формально, лингвистическая переменная определяется следующим образом.

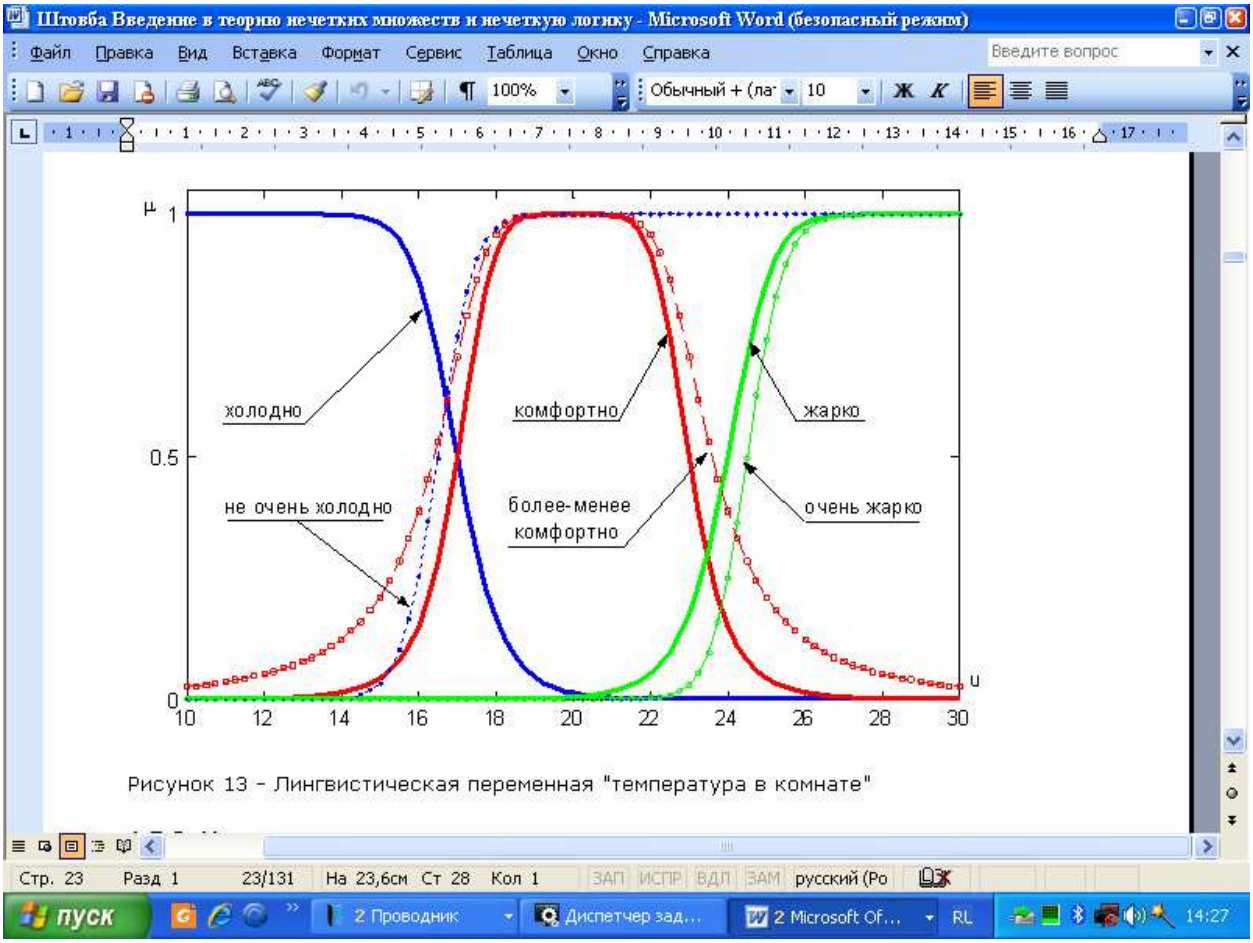
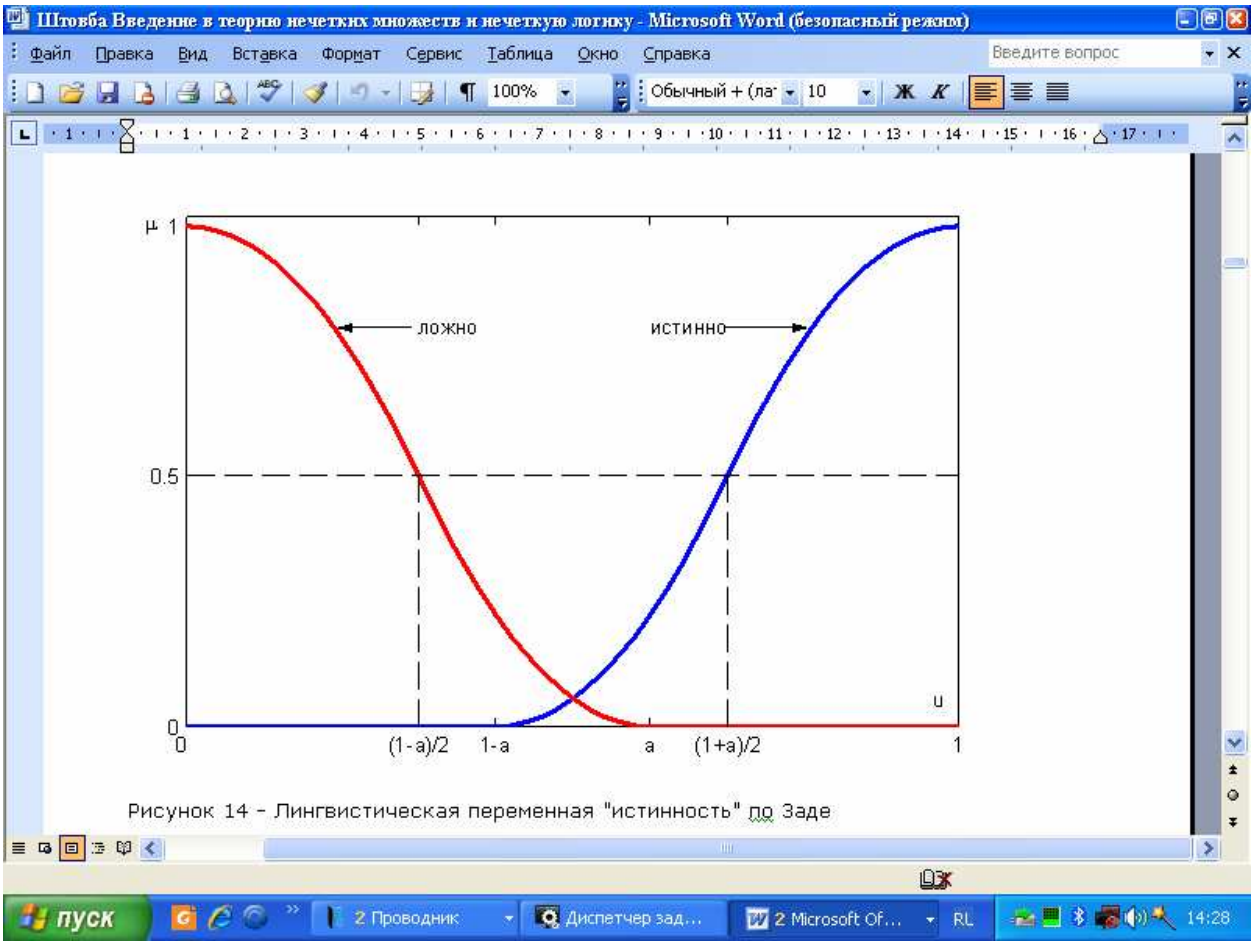
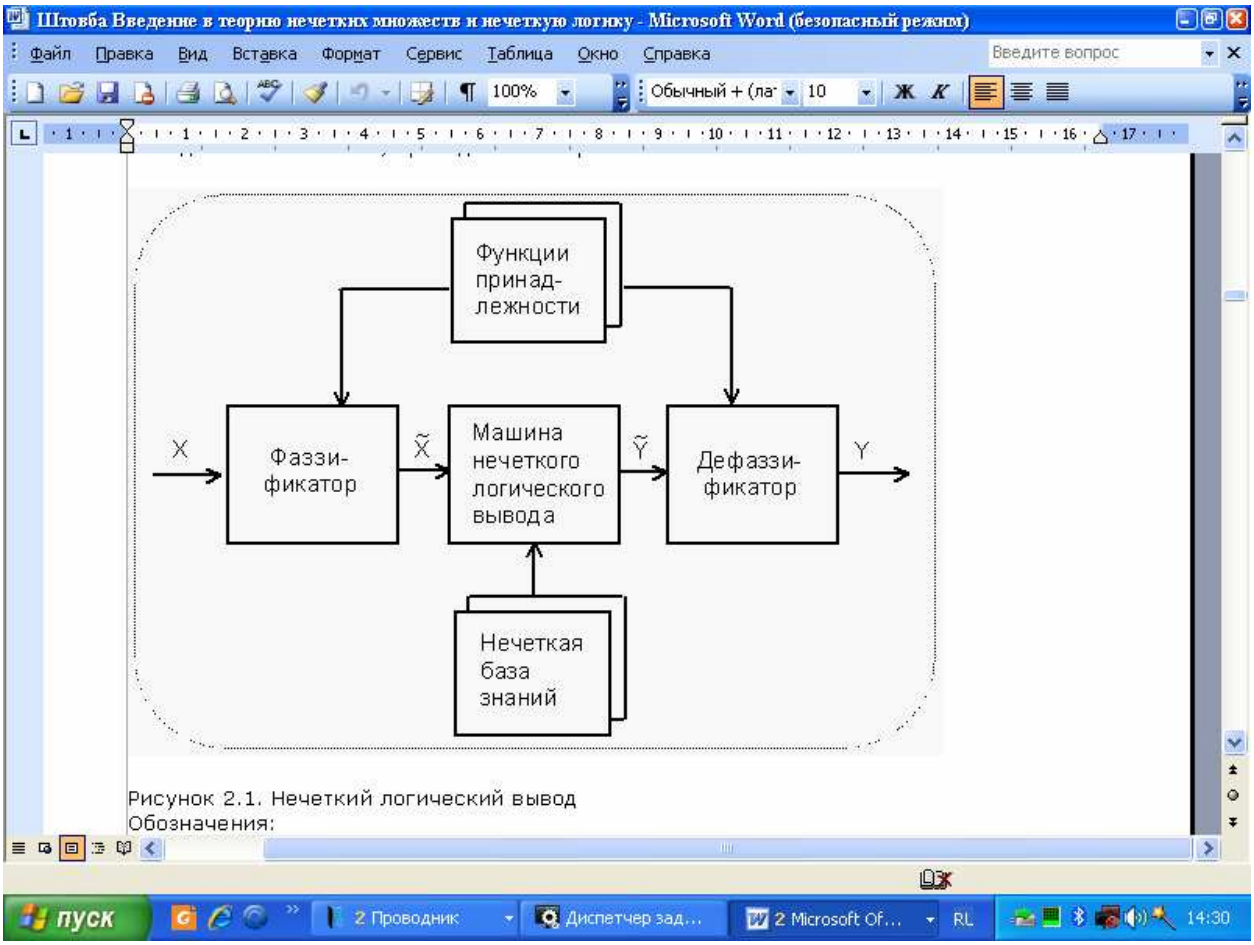


Рисунок 13 - Лингвистическая переменная "температура в комнате"


Штовба Введение в теорию нечетких множеств и нечеткую логику - Microsoft Word (безопасный режим)
Файл Правка Вид Вставка Формат Сервис Таблица Окно Справка
Введите вопрос
μ 1
ложно
истинно
0.5
0
u
0
(1-a)/2
1-a
a
(1+a)/2
1
Рисунок 14 - Лингвистическая переменная "истинность" по Заде
пуск
2 Проводник
Диспетчер зад...
2 Microsoft Of...
RL
14:28



Модуль fuzzy позволяет строит нечеткие системы двух типов - Мамдани и Сугэно. В системах типа Мамдани база знаний состоит из правил вида *"Если x_1=низкий и x_2=средний, то y=высокий"*. В системах типа Сугэно база знаний состоит из правил вида *"Если x_1=низкий и x_2=средний, то $y=a_0+a_1x_1+a_2x_2$"*. Таким образом, основное отличие между системами Мамдани и Сугэно заключается в разных способах задания значений выходной переменной в правилах, образующих базу знаний. В системах типа Мамдани значения выходной переменной задаются нечеткими термами, в системах типа Сугэно - как линейная комбинация входных переменных.

Проектирование систем типа Мамдани

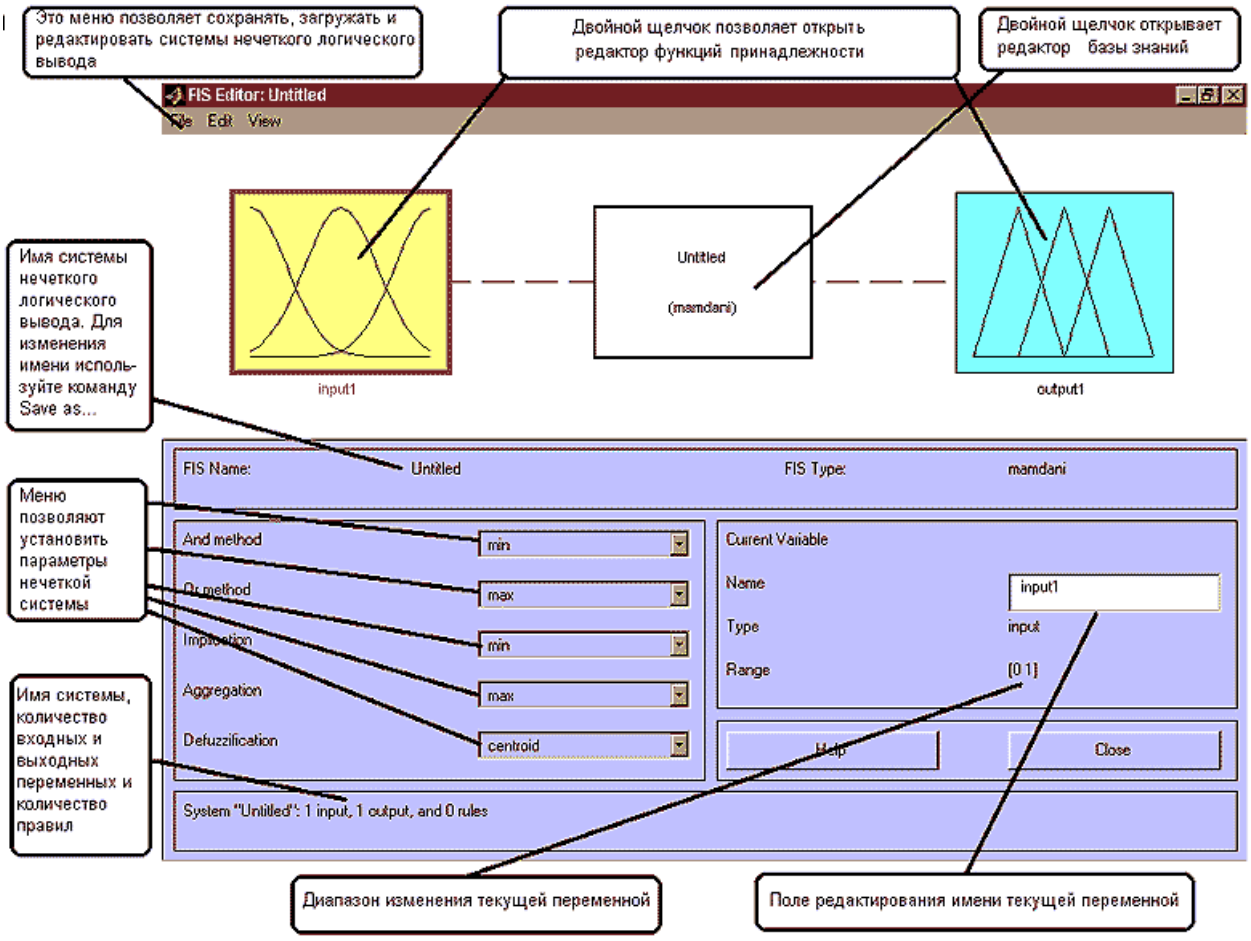
Рассмотрим основные этапы проектирования систем типа Мамдани на примере создания системы нечеткого логического вывода, моделирующей зависимость $y = x_1^2 \cdot \sin(x_2 - 1)$, $x_1 \in [-7,3], x_2 \in [-4.4,1.7]$. Проектирование системы нечеткого логического вывода будем проводить на основе графического изображения указанной зависимости.

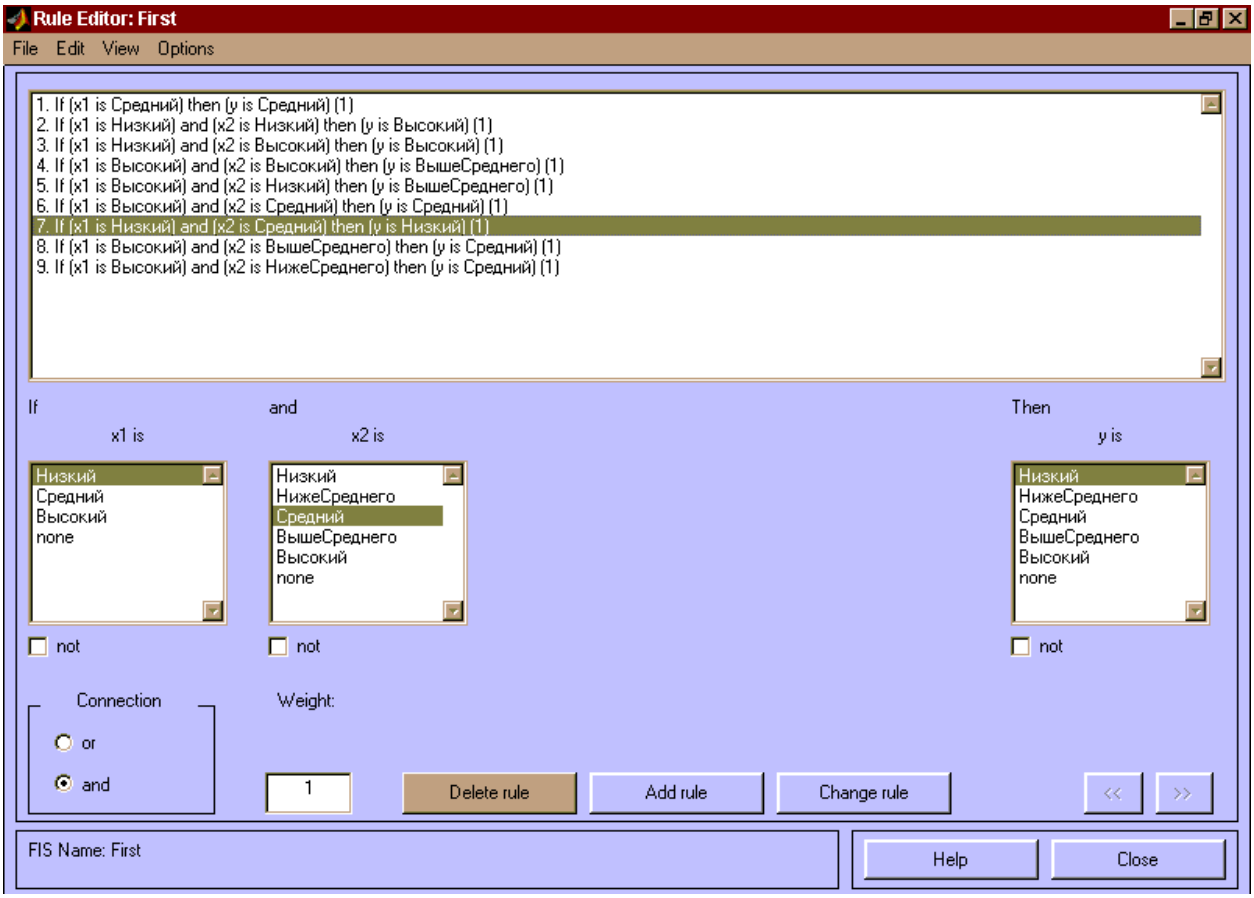
Для построения трехмерного изображения функции $y = x_1^2 \cdot \sin(x_2 - 1)$, в области $x_1 \in [-7,3], x_2 \in [-4.4,1.7]$, составим следующую программу:

```
%Построение графика функции y=x1^2*sin(x2-1)
%в области x1є[-7,3] и x2є[-4.4,1.7].
n=15;
x1=-7:10/(n-1):3;
```

```
x2=-4.4:6.1/(n-1):1.7;
y=zeros(n,n);
for j=1:n
y(j,:)=x1.^2*sin(x2(j)-1);
end
surf(x1,x2,y)
xlabel('x1')
ylabel('x2')
zlabel('y')
title('Target');
```

В результате выполнения программы получим графическое изображение, приведенное на рис. 3.1. Проектирование системы нечеткого логического вывода, соответствующей приведенному графику, состоит в выполнении следующей последовательности шагов.





База знаний в RuleEditor

Проектирование систем типа Сугэно

Рассмотрим основные этапы проектирования систем типа Сугэно на примере создания системы нечеткого логического вывода, моделирующей зависимость $y = x_1^2 \cdot \sin(x_2 - 1)$, $x_1 \in [-7,3], x_2 \in [-4.4,1.7]$ (рис. 3.1). Моделирование этой зависимости будем осуществлять с помощью следующей базы знаний:

1. Если x1=Средний, то y=0;
2. Если x1=Высокий и x2=Высокий, то y=2x1+2x2+1;
3. Если x1=Высокий и x2=Низкий, то y=4x1-x2;
4. Если x1=Низкий и x2=Средний, то y=8x1+2x2+8;
5. Если x1=Низкий и x2=Низкий, то y=50;
6. Если x1=Низкий и x2=Высокий, то y=50.

Началом практического применения теории нечетких множеств можно считать 1975г., когда Мамдани и Ассилиан (Mamdani and Assilian) построили первый нечеткий контроллер для управления простым паровым двигателем.

В 1982 Холмблад и Остергад (Holmblad and Osregaad) разработали первый промышленный нечеткий контроллер, который был внедрен в управление процессом обжига

цемента на заводе в Дании. Успех первого промышленного контролера, основанного на нечетких лингвистических правилах “Если - то” привел к всплеску интереса к теории нечетких множеств среди математиков и инженеров.

Несколько позже Бартоломеем Коско (Bart Kosko) была доказана теорема о нечеткой аппроксимации (Fuzzy Approximation Theorem), согласно которой любая математическая система может быть аппроксимирована системой, основанной на нечеткой логике. Другими словами, с помощью естественно-языковых высказываний-правил “Если - то”, с последующей их формализацией средствами теории нечетких множеств, можно сколько угодно точно отразить произвольную взаимосвязь “входы-выход” без использования сложного аппарата дифференциального и интегрального исчислений, традиционно применяемого в управлении и идентификации.

Системы, основанные на нечетких множествах разработаны и успешно внедрены в таких областях, как: управление технологическими процессами, управление транспортом, медицинская диагностика, техническая диагностика, финансовый менеджмент, биржевое прогнозирование, распознавание образов. Спектр приложений очень широкий - от видеокамер и бытовых стиральных машин до средств наведения ракет ПВО и управления боевыми вертолетами. Практический опыт разработки систем нечеткого логического вывода свидетельствует, что сроки и стоимость их проектирования значительно меньше, чем при использовании традиционного математического аппарата, при этом обеспечивается требуемый уровень робастности и прозрачности моделей.

Интеграция нейросетевых и нечетких систем

По утверждению специалистов [5], основное преимущество нечетких систем в отличие от нейросетей заключается в том, что знания в этих системах представляются в форме легко понимаемых человеком гибких логических конструкций, таких, как правила "IF... - THEN...". Кроме того, необходимо отметить, что, в соответствии с теоремой, доказанной Б. Коско (1993), любая математическая функция может быть аппроксимирована системой, основанной на нечеткой логике, следовательно, такие системы являются универсальными.

Основные трудности при использовании нечетких систем на практике связаны с априорным определением правил и построением функций принадлежности для каждого значения лингвистических переменных, описывающих структуру объекта, которые обычно проектировщик выполняет вручную. Поскольку вид и параметры функций принадлежности выбираются субъективно, они могут быть не вполне адекватны реальной действительности.

Главное преимущество нейросетевого подхода - возможность выявления закономерностей в данных, их обобщение, т.е. извлечение знаний из данных, а основной недостаток - невозможность непосредственно (в явном виде, а не в виде вектора весовых коэффициентов межнейронных связей) представить функциональную зависимость между входом и выходом исследуемого объекта. Недостатком нейросетевого подхода является также трудность формирования представительной выборки, большое число циклов обучения и забывание "старых" примеров, трудность определения размера и структуры сети.

Эти два подхода исследования сложных объектов на основе нейросетей и нечетких систем взаимно дополняют друг друга, поэтому целесообразно их объединение на основе принципа "мягких" вычислений (Soft Calculation). Основы построения таких систем с использованием разнородных методов были сформулированы Л. Заде в 1994 г. Они сводятся к следующему: терпимость к нечеткости и частичной истинности используемых данных для достижения интерпретируемости, гибкости и низкой стоимости решений.

Реализация объединения рассмотренных двух подходов возможна в следующих гибридных системах :

- нейросетевые нечеткие системы, в которых нейросетевая технология используется как инструмент в нечетких логических системах;
- нечеткие нейросети, в которых с помощью аппарата нечеткой математики осуществляется фазификация отдельных элементов нейросетевых моделей;
- нечетконейросетевые гибридные системы, в которых осуществляется объединение нечетких и нейросетевых моделей в единую систему.

Далее рассматриваются возможности реализации некоторых элементов гибридных систем согласно работе [5].

Нейросетевая реализация элементов нечетких систем.

Нейросетевой нечеткой системой называется такая система, в которой отдельные элементы нечеткости (функции принадлежности, логические операторы, отношения) и алгоритмы вывода реализуются с помощью нейросетей (НС).

Рассмотрим вопросы, связанные с обучением НС для представления функций принадлежности произвольной формы. Построение простых форм функции принадлежности, таких как треугольная или холмообразная, может осуществляться на одном нейроне путем подстройки функции активации к желаемой функции принадлежности. Например, для представления холмообразная функции принадлежности может быть использован нейрон, функция активации которого имеет вид:

$f(s) = \exp\{-(s-m)^2 / \square^2\}$ где s - сумма частных произведений входов нейрона;

m - значение нечеткой переменной, при которой достигается наибольшее значение функции принадлежности;

$\square$ - среднеквадратичное отклонение функции принадлежности от максимального значения.

В качестве примера рассмотрим возможность построения функций принадлежности для трех значений некоторой лингвистической переменной: "мало (S - Small)", "средне (M-Medium)", "много (L - Large)".

Например, для значения лингвистической переменной "мало" функцию принадлежности можно определить следующим образом:

$y_1 = \mu_S(x) = 1/1 + \exp(- w_g(x + w_c))$

где w_c - весовой коэффициент для смещения и w_g- вес суммарного сигнала на входе нелинейного преобразователя. Путем подстройки весовых коэффициентов формируются соответствующие функции принадлежности.

Реализация нечетких И (AND), ИЛИ (OR) и других операторов в нейросетевом логическом базисе дает основу для построения нейросетевых нечетких моделей. Возможность использования различных функций активации нейрона позволяет разрабатывать различные (желаемые) логические операторы, т.е. можно задать функцию активации, реализующую min-оператор для нечеткой операции AND, max-оператор для нечеткой операции OR.

Нейросетевая реализация нечетких отношений.

Как было показано выше, возможны различные способы реализации нечетких отношений в системах нечеткого логического вывода. Основные трудности построения нечетких

отношений связаны с настройкой большого числа функций принадлежности. Для устранения этих трудностей можно использовать подход, основанный на обучении нейросетей с весовыми коэффициентами, соответствующими этим функциям принадлежности .

Пусть X и Y- нечеткие множества в U и V соответственно, R - нечеткое отношение в U×V , где U , V- универсальные множества.

Рассмотрим уравнение нечеткого отношения

Y=X•R,

где • - операция композиции (например, max-min операция). Пусть U ={x_1, x_2, ..., x_n} и V={y_1,y_2, ...,y_n}. Тогда можно записать max-min композицию

$\mu_Y(y_j) = \max (\min(\mu_X(x_i), \mu_R(x_i y_i))$, (4.1)

где X- нечеткое множество на U; Y- нечеткое множество на V, R - нечеткое отношение R:U×V - >[0,1], описываемое всеми отношениями принадлежности (связями) между входом и выходом.

Выражение (4.1) можно записать в другом виде, представив отношение R в виде матрицы. Тогда величина $\mu_Y(y_j)$ будет определяться путем композиции вектора X и j-ой колонки матрицы отношения R.

Такое отношение может быть реализовано нейросетью, структура которой показана на рис. 4.1. В этой сети выходной узел выполняет max-min композицию вместо обычной операции суммирования, реализуемой нейросетью.

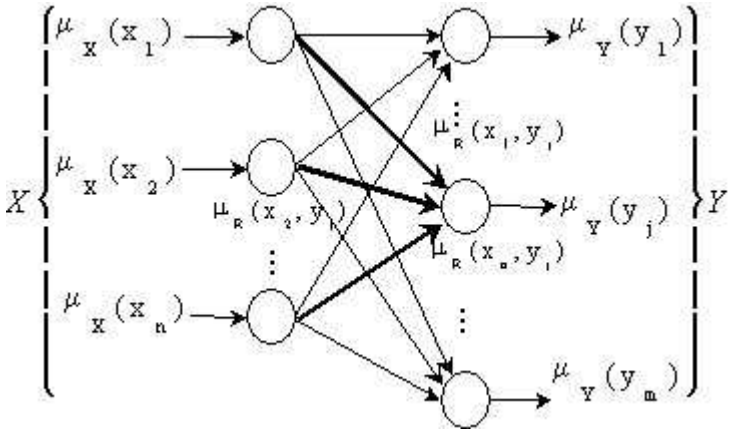


Рис.4.1. Нейросетевая модель нечетких отношений.

Нейросетевая модель нечеткого вывода.

Анализ процедуры классического нечеткого вывода позволяет выделить две главные проблемы, возникающие при организации вывода на основе нечеткой логики, которые более эффективно можно реализовать на основе нейросетевого подхода. Первая связана с определением функций принадлежности, использующихся в условной части правил, а вторая - с выбором правила, определяющего решение, из совокупности правил-кандидатов. Рассмотрим модель, в которой нейросеть используется для организации вывода, аналогичного композиционному нечеткому выводу.

Один из вариантов реализации нейросетевого нечеткого вывода (для базы из трех правил R_1, R_2, R_3), позволяющего формировать функцию принадлежности условной части IF правила и управлять выводом на основе сформированной функции принадлежности, представлен на рис. 4.2.

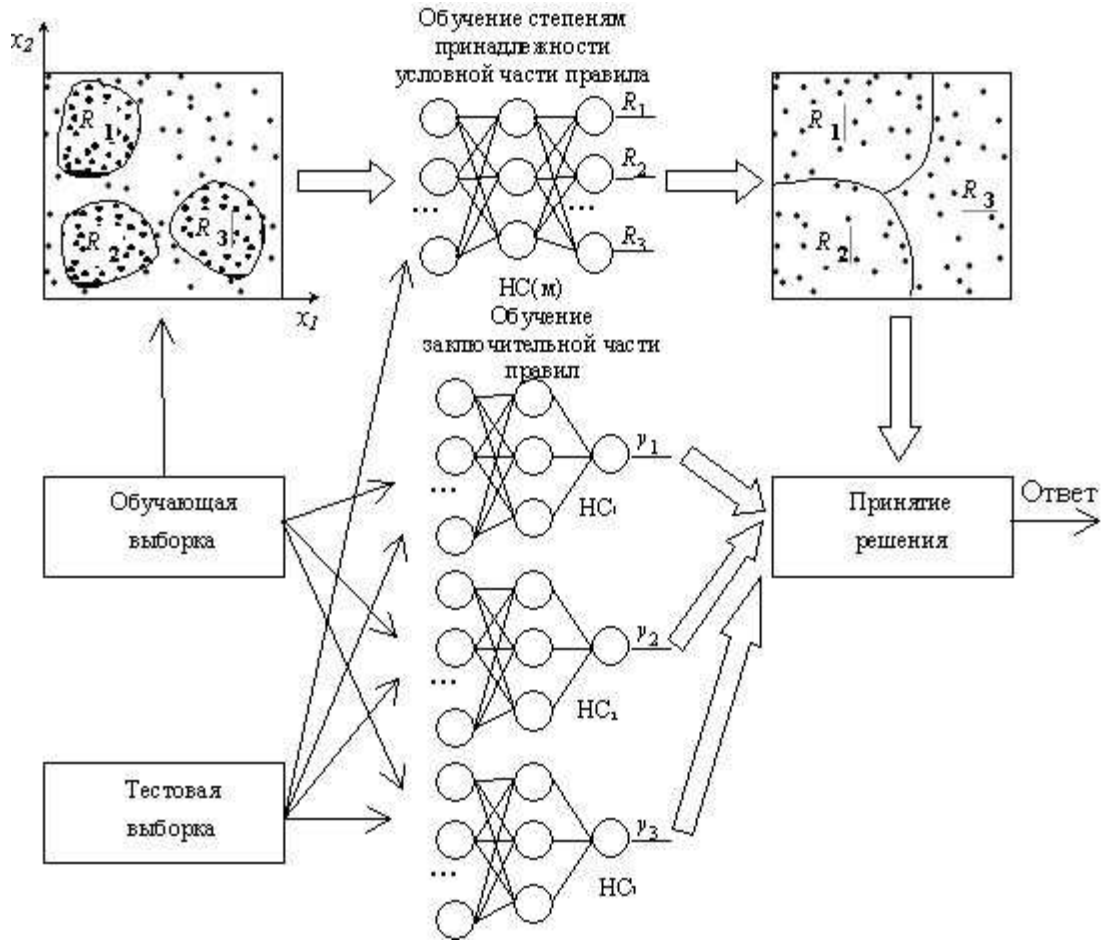


Рис. 4.2. Схема нейросетевой реализации нечеткого вывода.

Правила нечеткого вывода в рассматриваемой модели имеют следующий формат: R_s: IF X= (x_1, x_2, ..., x_n) is A_s THEN $y_s = HC_S$ (y_1,y_2, ...,y_n), s = 1, 2,.., r, где r - число правил вывода; A_s - представляет нечеткое множество условной части каждого правила; $HC_S(\bullet)$ - определяет структуру нейросетевой модели с входами x_1, x_2, ..., x_n и выходом y_s , причем для каждого правила используется своя нейросеть. Для определения функций принадлежности условной части правил используется нейронная сеть $HC_S(\mu)$. В качестве вида нейросетей целесообразно выбрать многослойный персептрон.

Нечеткие элементы нейросетевых систем

Нечеткой нейросетью называется такая система, в которой отдельные элементы (нейроны, функции активации и т.д.) и алгоритмы обучения являются нечеткими. Рассмотрим три базовых типа нечетких нейронов, на основе которых можно построить нечеткие НС:

- нечеткий нейрон с четкими входными сигналами, но с нечеткими весами;
- нечеткий нейрон с нечеткими входными сигналами и нечеткими весами;
- нечеткий нейрон, описываемый посредством нечетких логических уравнений.

Нечеткий нейрон первого типа (рис. 4.3.) имеет n четких входов x_1, x_2,..., $x_п$. Весовые коэффициенты такого нейрона представляют собой нечеткие множества A_i, i = 1, ..., n, т.е. операции взвешивания входных сигналов в классическом нейроне заменяются функциями

принадлежности и происходит их объединение (агрегирование) в узле n_1.Выходной сигнал y $=\mu_{n1}(x_1, x_2, ..., x_n)$, принадлежит интервалу [0,1].

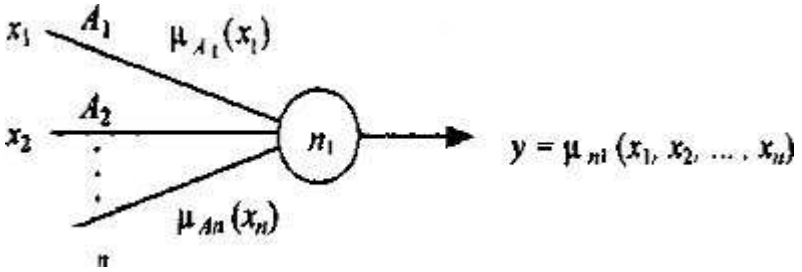


Рис 4.3. Модель нечеткого нейрона с четкими входными сигналами и нечеткими весами.

Операцию агрегирования взвешенных сигналов обозначим как⊗ , для реализации которой можно использовать нечеткие операторы t-нормы и t-ко-нормы. В зависимости от типа используемой операции⊗ можно получить различные схемы нечетких нейронов. Математическое описание такого нейрона можно представить в виде:

$\mu_{n1}(x_1, x_2, ..., x_n)= \mu_{A1}(x_1) \otimes \mu_{A2}(x_2) \otimes \ldots \otimes \mu_{An}(x_n)$

где ⊗ - знак операции агрегирования;

x_i - i-ый (четкий) вход нейрона;

$\mu_{Ai}(x_i)$- функция принадлежности i -го (нечеткого) веса;

μ_{n1}- выходная функция нейрона.

Нечеткий нейрон второго типа n_2 подобен нейрону первого типа, но все входы и выходы нейрона являются нечеткими множествами. Каждый сигнал на каждом нечетком входе X_i преобразуется путем операции взвешивания в нечеткое множество $X_{i'} = A_i * X_i$, i= 1 , . . . , n, с использованием некоторого оператора * , где A_i — i -й нечеткий вес.

Операция взвешивания здесь не является функцией принадлежности, как в нейроне первого типа; вместо нее проводится модификация каждого нечеткого входа. Математическое описание нейрона второго типа имеет вид:

$X_{i'} = A_i * X_i$, $Y= X_{1'} \otimes X_{2'} \otimes \ldots \otimes X_{n'}$

где Y- нечеткое множество (выход нейрона n_2);

X_i ,$X_{i'}$ — i-ый вход до и после операции взвешивания соответственно;

A_i — вес на i-ой связи;⊗ - операция агрегирования;

* - оператор взвешивания двух нечетких множеств (например, операция произведения двух нечетких множеств).

Отношение входа - выхода нечеткого нейрона третьего типа n_3 с n нечеткими входами и одним нечетким выходом может задаваться с помощью одного нечеткого правила IF-THEN:

IF X_1 AND X_2 AND. . . AND X_n THEN Y,

где X_1,X_2, ..., X_n- текущие входы, Y- текущий выход.

В общем случае нейрон рассматриваемого типа описывается нечетким отношением R

$R=f(X_l,X_2,...,X_n,Y)$,

где f (•) - функция импликации. Задавая соответствующие входы (четкие или нечеткие) x_1,x_2, ...,x_n, в соответствии с композиционным правилом вывода получим

$Y_{i'}= X_1 \bullet (X_2 \bullet (... \bullet (X_n \bullet R_i)\ldots)$,

где • оператор композиционного правила вывода (свертки).

На основе рассмотренных типов нечетких нейронов можно конструировать нечеткие нейросети различной архитектуры, которые в отличие от обычных НС обладают расширенными возможностями представления и обработки данных.

Графические средства пакета Fuzzy Logic Toolbox системы Матлаб.

Пример разработки системы нечеткого вывода в интерактивном режиме

В качестве примера разработки системы нечеткого вывода в интерактивном режиме с помощью графических средств пакета Fuzzy Logic Toolbox рассмотрим следующую нечеткую модель, которая входит в число демонстрационных примеров системы MATLAB.

Пример. Рассмотрим ситуацию в ресторане, при которой, согласно принятым в США традициям, после окончания обслуживания посетителя принято оставлять официанту чаевые. Основываясь на устоявшихся в этой стране обычаях и интуитивных представлениях посетителей ресторанов, величина суммы чаевых не является постоянной и зависит, например, от качества обслуживания и качества приготовления заказанных блюд.

Задача состоит в том, чтобы разработать некоторую экспертную систему, которая была бы реализована в виде системы нечеткого вывода и позволяла бы определять величину чаевых на основе субъективных оценок посетителей качества обслуживания и качества приготовления заказанных блюд.

Эмпирические знания о рассматриваемой проблемной области могут быть представлены в форме следующих эвристических правил продукций:

1. Если обслуживание плохое или ужин подгоревший, то чаевые — малые.
2. Если обслуживание хорошее, то чаевые — средние.
3. Если обслуживание отличное или ужин превосходный, то чаевые – щедрые.

В качестве входных параметров системы нечеткого вывода будем рассматривать две нечеткие лингвистические переменные: «*service*» («*качество обслуживания*») и «*food*» («*качество приготовления заказанных блюд*»), а в качестве выходных параметров — нечеткую лингвистическую переменную «*tip*» («*величина чаевых*»).

В качестве терм-множества первой лингвистической переменной «*service*» будем использовать множество T_1 = {«*poor*», «*good*», «*excellent*»} («*плохое*», «*хорошее*», «*отличное*»), а в качестве терм-множества второй лингвистической переменной «*food*» будем использовать множество T_2 = {«*rancid*», «*delicious*»} («*подгоревший*», «*превосходный*»). В качестве терм-множества выходной лингвистической переменной «*tip*» будем использовать множество T_3 = {«*cheap*», «*average*», «*generous*»} («*малые*», «*средние*», «*щедрые*»). При этом каждый из термов входных переменных будем оценивать по 10-балльной шкале, а термы выходной переменной – в процентах от стоимости заказанных блюд.

С учетом сделанных уточнений рассмотренная субъективная информация о величине чаевых может быть представлена в форме трех правил нечетких продукций следующего вида (система нечеткого вывода типа Мамдани):

ПРАВИЛО_1: ЕСЛИ «*качество обслуживания плохое*» ИЛИ «*ужин подгоревший*» ТО «*величина чаевых малая*»,

ПРАВИЛО_2: ЕСЛИ «*качество обслуживания хорошее*» ТО «*величина чаевых средняя*»,

ПРАВИЛО_3: ЕСЛИ «*качество обслуживания отличное*» ИЛИ «*ужин превосходный*» ТО «*величина чаевых щедрая*».

Следует отметить, что соответствующая данному примеру система нечеткого вывода хранится во внешнем файле с именем **tipper.fis** в папке …\MATLAB\toolbox\fuzzy\fuzdemos.

Процесс разработки системы нечеткого вывода в интерактивном режиме для рассмотренного выше примера «*Чаевые в ресторане*» состоит в выполнении следующей последовательности действий.

1. Вызвать редактор систем нечеткого вывода FIS, для чего в окне команд набрать имя соответствующей функции **fuzzy**. После выполнения этой команды на экране появится графический интерфейс редактора FIS с именем системы нечеткого вывода **Untitled** и типом системы нечеткого вывода (Мамдани), предложенными по умолчанию (см. рис. 1.1).

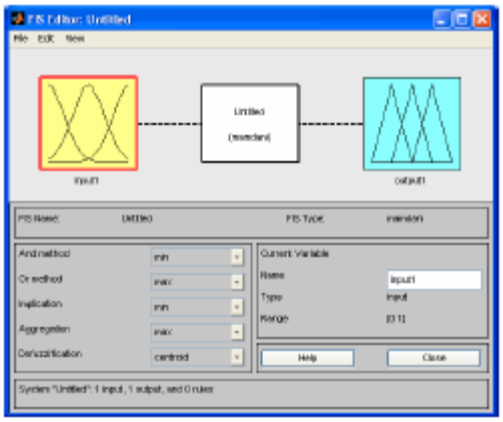


Рис. 1.1. Графический интерфейс редактора FIS, вызываемый функцией **fuzzy**

2. Поскольку в примере рассматривается система нечеткого вывода с двумя входами, необходимо добавить в разрабатываемую систему FIS еще одну входную переменную. Для этого следует выполнить команду меню **Edit -> Add Variable... -> Input. B** результате выполнения данной команды на диаграмме системы нечеткого вывода появится новый желтый прямоугольник с именем второй входной переменной **input2** (рис. 2.1).

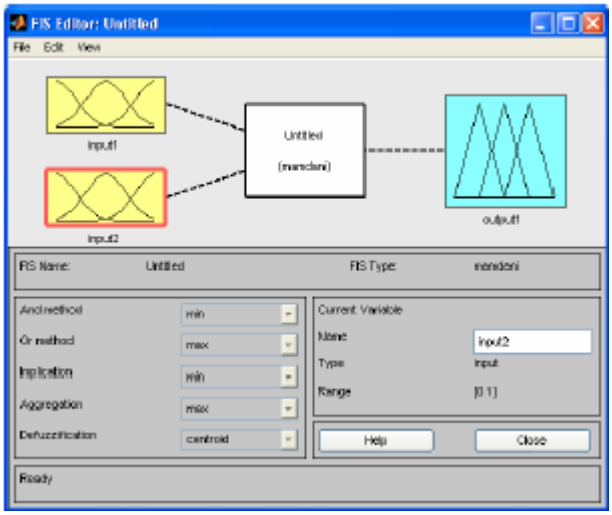


Рис. 2.1. Вид редактора FIS после добавления второй входной переменной

3. Изменим имена входных и выходных переменных, предложенных системой MATLAB по умолчанию: необходимо выделить прямоугольник с именем соответствующей переменной, выполнив щелчок на его изображении на диаграмме (стороны выделенного прямоугольника имеют красный цвет). После чего следует набрать новое имя переменной в поле ввода **Name** в правой части редактора FIS. Результат изменения имен переменных системы нечеткого вывода изображен на рис. 2.2.

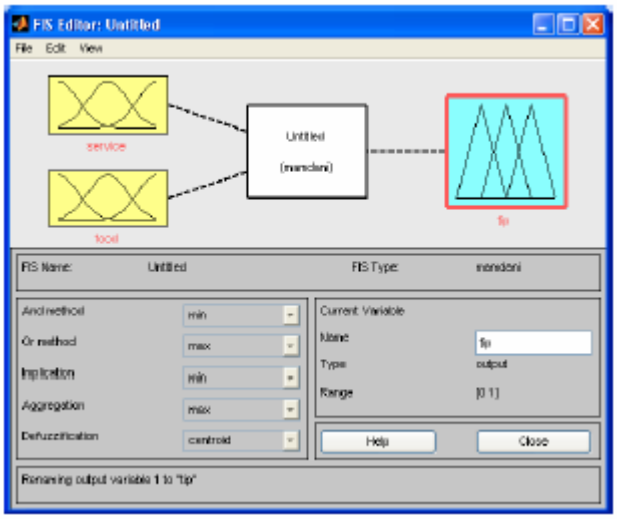


Рис. 2.2. Вид редактора FIS после изменения имен переменных, предложенных системой MATLAB по умолчанию

При создании системы нечеткого вывода в интерактивном режиме допускается применение русскоязычных названий. В этом случае следует иметь в виду, что в некоторых версиях MATLAB имеются проблемы с кириллицей в названиях переменных и термов (символы «с» и «я»). Чтобы избежать проблем с корректным отображением символов кириллицы, следует давать такие имена переменным, которые состоят из одного слова без дополнительных служебных символов. Также следует иметь в виду, что при отображении на экране имен переменных используется LaTeX-транслятор, который воспринимает символ подчеркивания как преобразование следующего символа в нижний индекс. Например, имя переменной input_11 отобразится как **input11**. Более подробно об использовании LaTeX-транслятора для записи математических выражений можно узнать в соответствующих разделах помощи по LaTeX и MATLAB.

4. Изменим имя системы нечеткого вывода (Untitled), предложенное по умолчанию. Для этого сохраним создаваемую структуру FIS во внешнем файле с именем **mytip.fis**, выполнив команду меню

File -> Export -> To Disk... .

При этом будет вызвано стандартное диалоговое окно сохранения файла, в котором пользователю предлагается ввести имя соответствующего файла (расширение файла приписывается автоматически).

Оставим без изменения предложенные системой MATLAB по умолчанию:

- метод нечеткого логического И **(And method)** — значение «min»,
- метод нечеткого логического ИЛИ **(Or method)** — значение «max»,
- метод импликации **(Implication)** – значение «min»,
- метод агрегирования **(Aggregation)** — значение «max» и метод дефаззификации **(Defuzzification)** – значение «centroid».

5. Теперь необходимо определить термы и их функции принадлежности для входных и выходных переменных нашей системы нечеткого вывода. Для этой цели следует воспользоваться редактором функций принадлежности, который может быть вызван одним из следующих способов:

— двойным щелчком на значке прямоугольника с именем соответствующей переменной;

— командой меню **Edit -> Membership Functions...** (предварительно должен быть выделен прямоугольник с именем соответствующей переменной);

— нажатием клавиш <Ctrl>+<2> (предварительно также должен быть выделен прямоугольник с именем соответствующей переменной).

После вызова редактора функций принадлежности каждой из переменных по умолчанию предлагается три терма с треугольными функции принадлежности (рис. 2.3).

Сначала изменим диапазон определения значений входных переменных, для чего в полях ввода **Range** и **Display Range** изменим верхнее значение с 1 на 10 (баллов). Аналогично выполняются изменения соответствующих диапазонов для выходной переменной «чаевые», при этом верхнее значение 1 следует заменить на 30 (%).

Изменения подтверждаются нажатием на клавишу <Enter>.

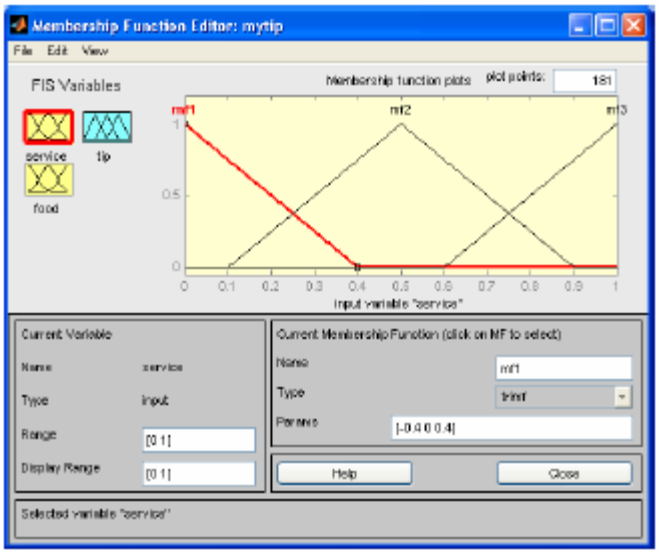


Рис. 2.3. Вид редактора функций принадлежности после его вызова с функциями принадлежности для термов переменной «*service*», предложенных системой MATLAB по умолчанию

Далее изменим названия термов первой входной переменной «*service*», предложенные системой MATLAB по умолчанию (mf1, mf2, mf3) на «*poor*», «*good*», «*excellent*» соответственно. После чего изменим тип функций принадлежности первой переменной, предложенный по умолчанию, на функции типа Гаусса (gaussmf), выбрав соответствующий пункт в поле **Type**. Параметры вновь заданных функций принадлежности оставим без изменения. Вид редактора функций принадлежности после внесенных изменений для первой из входных переменных изображен на рис. 2.4.

Аналогичным образом изменим названия термов второй входной переменной «*food*» и удалим один из термов с соответствующей функцией принадлежности. Для удаления терма следует выделить удаляемую функцию принадлежности и нажать клавишу <Delete> на клавиатуре. Переход к редактированию переменной осуществляется щелчком на изображении прямоугольника с именем необходимой переменной. Для переменной «*food*» изменим тип функций принадлежности ее термов на трапециевидные функции (trapmf) и их параметры следующим образом: для терма «*rancid*» зададим параметры [0 0 1 3], а для терма «*delicious*» — [7 9 10 10].

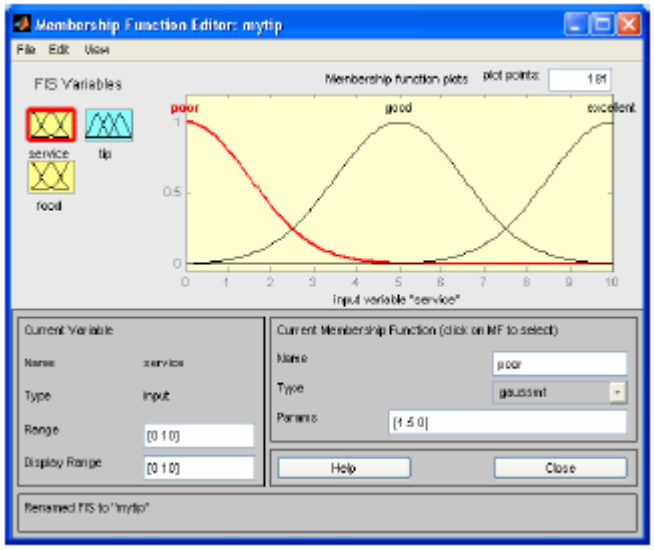


Рис. 2.4. Вид редактора функций принадлежности после изменения названий термов и типа их функций принадлежности для первой входной переменной «*service*»

Вид редактора функций принадлежности после внесенных изменений для второй входной переменной изображен на рис. 2.5.

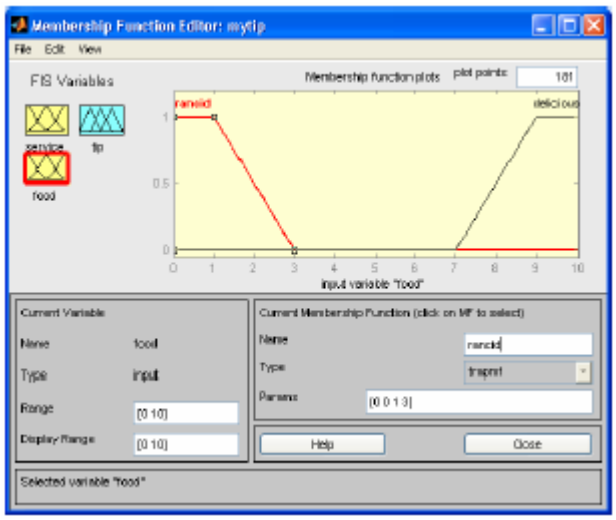


Рис. 2.5. Вид редактора функций принадлежности после изменения названия термов и типа их функций принадлежности для второй входной переменной «*food*»

Наконец, изменим названия термов и параметры функций принадлежности для выходной переменной «*tip*», оставив без изменения треугольный тип функций принадлежности, предложенный системой MATLAB. Для терма «*cheap*» зададим параметры [0 5 10], а для терма «*average*» — [10 15 20], для терма «*generous*» — [20 25 30]. Вид редактора функций принадлежности после сделанных изменений для выходной переменной «*tip*» изображен на рис. 2.6.

6. Теперь определим правила нечеткого вывода для разрабатываемой экспертной системы. Для этой цели следует воспользоваться редактором правил, который может быть вызван одним из следующих способов:

— двойным щелчком на значке квадрата в центре с именем создаваемой системы нечеткого вывода (**myfis**);

— командой меню **Edit -> Rules...;**

— нажатием клавиш <Ctrl>+<3>.

Поскольку первоначально база правил нечеткого вывода пуста, то после вызова редактора правил центральное многострочное поле ввода не содержит никаких правил. Для их определения следует использовать поля меню и переключатели в нижней части графического интерфейса редактора правил. Для задания первого правила следует оставить выделенные по умолчанию поле с именем терма «*poor*» для первой входной переменной, поле с именем терма «*rancid*» для второй входной переменной и поле с именем терма «*poor*» для выходной переменной. Далее следует переключатель **Connection** поставить в положение **or** (логическое ИЛИ) и нажать на кнопку **Add rule.** После этого первое правило с символами кириллицы отобразится в верхнем окне.

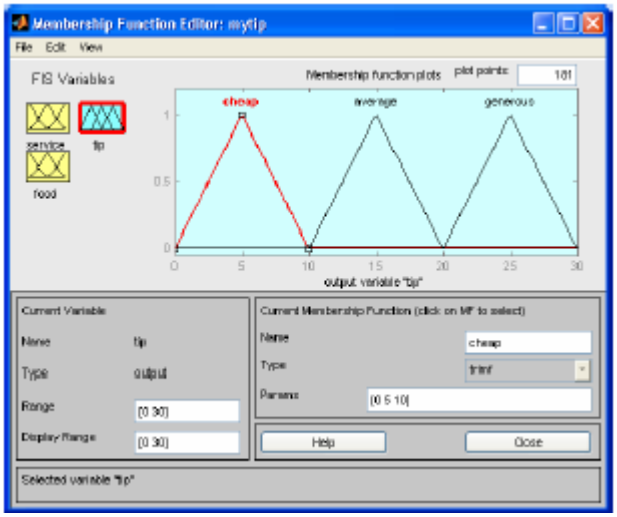


Рис. 2.6. Вид редактора функций принадлежности после изменения названия термов и типа их функций принадлежности для выходной переменной «*tip*»

Аналогичным образом задается второе правило, для которого следует выделить имена термов «*good*», «*none*» и «*average*», и третье правило с именами термов «*excellent*», «*delicious*» и «*generous*» для соответствующих переменных. Вид редактора правил после их определения для разрабатываемой экспертной системы изображен на рис. 2.7.

Заметим, что в поле ввода **Weight** отображается вес каждого правила, который можно изменять в пределах интервала [0, 1] (оставим без изменения его значение по умолчанию, равное 1 для всех правил). Этот же вес правил записывается в круглых скобках в окне правил после каждого из правил нечеткого вывода.

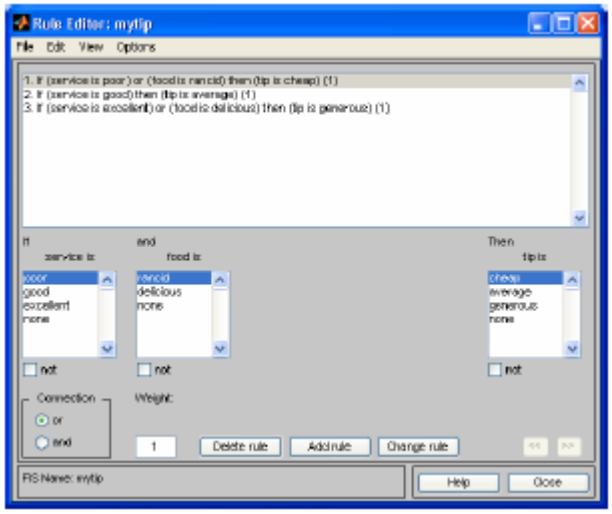


Рис. 2.7. Вид редактора правил нечеткого вывода после их определения

7. После задания правил нечеткого вывода оказывается возможным получить результат нечеткого вывода (значение выходной переменной) для конкретных значений входных переменных. С этой целью необходимо открыть программу просмотра правил одним из следующих способов:

— командой меню **View -> Rules** редактора FIS;

— командой меню **View -> Rules** редактора функций принадлежности;

— командой меню **View -> Rules** редактора правил;

— нажатием клавиш <Ctri>+<5>.

После вызова программы просмотра правил для нашей системы нечеткого вывода по умолчанию для входных переменных предложены средние значения из интервала их допустимых значений (значения [5 5] в поле ввода **Input**). Это означает, что посетитель ресторана оценивает качество обслуживания в 5 баллов и качество ужина также в 5 баллов. Таким значениям входных переменных соответствует значение чаевых в 15 %, которое отображается выше прямоугольников правил в правой части окна программы просмотра.

Изменим значения входных переменных для другого случая, которому соответствует качество обслуживания в 0 баллов («хуже некуда») и качество ужина в 10 баллов («лучше не бывает»). Для этого курсор мыши переместим в поле ввода **Input** и введем соответствующие значения входных переменных: [0 10]. Система MATLAB оставит значение чаевых без изменения (15 %), однако на диаграмме правил можно заметить результаты выполненных изменений (рис. 2.8).

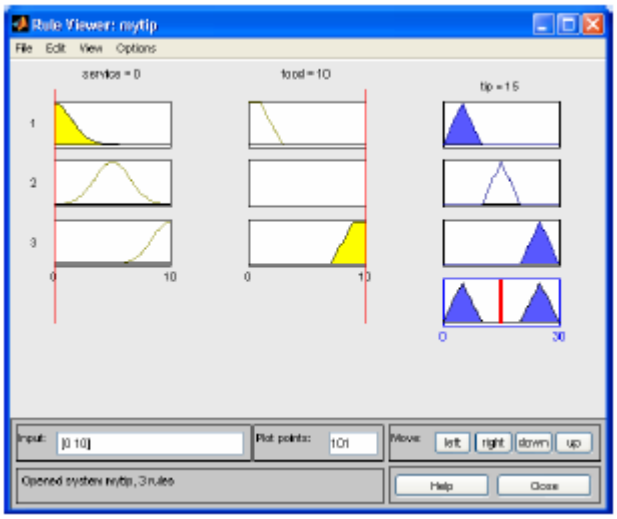


Рис. 2.8. Вид программы просмотра правил нечеткого вывода после изменения значений входных переменных на [0 10]

Поскольку процесс нечеткого моделирования предполагает анализ результатов нечеткого вывода при различных значениях входных переменных с целью установления адекватности разработанной нечеткой модели (в данном случае — экспертной системы), рассмотрим и другие случаи. Предположим, что качество обслуживания оценивается в 10 баллов («лучше не бывает»), а качество ужина — в 2 балла («бывает и хуже, но реже»). Введем соответствующие значения переменных аналогичным способом. В этом случае разработанная нами система нечеткого вывода рекомендует нам оставить чаевые в размере 16.4 %.

Если же предположить, что качество обслуживания по-прежнему отличное (10 баллов), а качество ужина несколько улучшилось и оценивается в 3 балла, то величина чаевых существенно изменится и станет равной 24.7 %. Более того, дальнейшее увеличение качества ужина не оказывает изменения величины чаевых. В частности, для значений входных переменных [10 10] величина чаевых составит попрежнему 24.7 %. Если некоторым из посетителей такая экспертная система покажется неадекватной (в частности, для случая значений входных переменных [10 10] можно бы оставить максимальные чаевые в 30 %), то разработанная система нечеткого вывода потребует модификации. Данная модификация может потребовать изменения существующих правил или добавления новых, а также изменения параметров функций принадлежности входных и выходной переменных. Более тонкая настройка модели может быть связана с увеличением количества термов для каждой из входных и выходных переменных, что, в свою очередь, приведет к увеличению количества правил в системе нечеткого вывода и общему усложнению модели.

8. Для окончательного анализа разработанной нечеткой модели может оказаться полезной программа просмотра поверхности нечеткого вывода, которая может быть вызвана одним из следующих способов:

— командой меню **View -> Surface** редактора FIS;

— командой меню **View -> Surface** редактора функций принадлежности;

— командой меню **View -> Surface** редактора правил;

— командой меню **View -> Surface** программы просмотра правил;

— нажатием клавиш <Ctrl>+<6>.

Графический интерфейс программы просмотра поверхности нечеткого вывода для разработанной нечеткой модели изображен на рис. 2.9.

Эта программа служит для общего анализа адекватности нечеткой модели, позволяя оценить влияние изменения значений входных нечетких переменных на значение одной из выходных нечетких переменных. В случае необходимости можно получить график зависимости выходной переменной от одной из входных переменных.

Для этого необходимо выбрать нужную переменную в раскрывающемся списке **X (input)**, а в раскрывающемся списке **Y (input)** выбрать значение «none». Полученный график зависимости изображен на рис. 2.10.

Построенный график зависимости соответствует среднему значению второй входной переменной *(«food»)* в 5 баллов. Это значение может быть изменено пользователем, для чего следует ввести нужное значение в поле ввода **Ref.Input**. Заметим, что значение **NaN** для первой входной переменной соответствует ее изменению во всем интервале определения [0, 10].

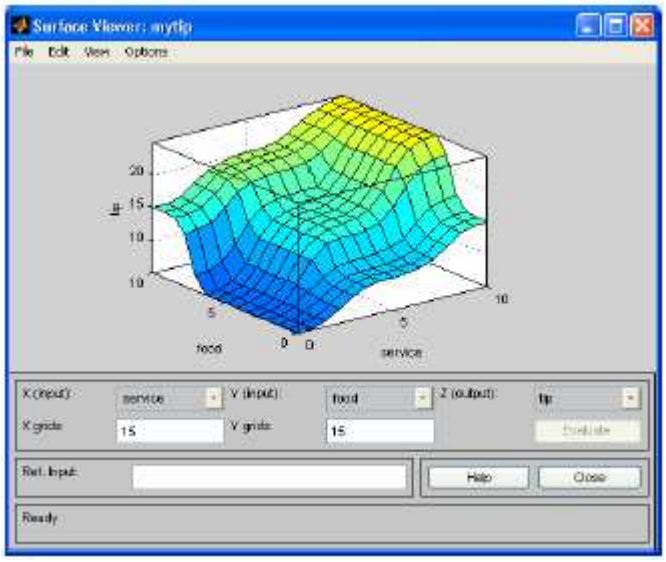


Рис. 2.9. Вид программы просмотра поверхности нечеткого вывода для разработанной нечеткой модели

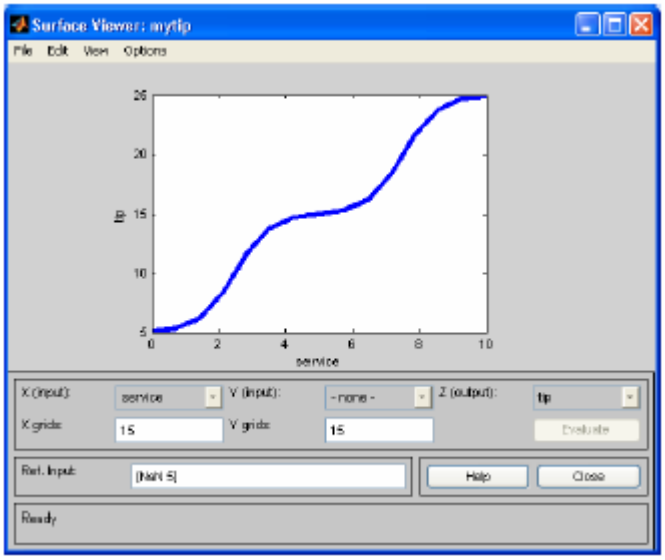


Рис. 2.10. График зависимости выходной переменной от первой из входных переменных для разработанной нечеткой модели

Заканчивая рассмотрение процесса разработки простейшей системы нечеткого вывода в интерактивном режиме, следует заметить, что наиболее эффективным этот способ оказывается для сложных нечетких моделей с большим числом переменных и правил нечеткого вывода. В этом случае задание переменных и функций принадлежности их термов в графическом режиме, а также визуализация правил позволяют существенно уменьшить трудоемкость разработки нечеткой модели, снизить количество возможных ошибок и сократить общее время нечеткого моделирования.

В процессе работы следует помнить, что количество переменных и правил в нечеткой модели, которые могут быть визуализированы, ограничено. В частности, если число входных переменных превышает 10, то их отображение в соответствующих графических редакторах происходит с искажениями.

Процесс разработки системы нечеткого вывода в режиме команд может дополнить, а в отдельных случаях и заменить процесс разработки в интерактивном режиме, предоставляя пользователю полный контроль над всеми переменными рабочей области системы MATLAB.

Лекция 12. Гибридные сети.

Краткое введение в гибридные сети

Каждая разновидность систем искусственного интеллекта имеет свои особенности, например, по возможностям обучения, обобщения и выработки выводов, что делает ее наиболее пригодной для решения одного класса задач и менее пригодной — для другого.

Так, нейронные сети хороши для задач распознавания образов, но весьма неудобны для выяснения вопроса, как они такое распознавание осуществляют. Они могут автоматически приобретать знания, но процесс их обучения зачастую происходит достаточно медленно, а анализ обученной сети весьма сложен (обученная сеть для пользователя обычно представляется как черный ящик). При этом какую-либо априорную информацию (знания эксперта) для ускорения процесса обучения в нейронную сеть ввести невозможно

Системы с нечеткой логикой, напротив, хороши для объяснения получаемых с их помощью выводов, но они не могут автоматически приобретать знания для использования их в механизмах выводов. Необходимость разбиения универсальных множеств на отдельные области, как правило, ограничивает количество входных переменных в таких системах небольшим значением.

Вообще говоря, теоретически, системы с нечеткой логикой и искусственные нейронные сети эквивалентны друг другу, однако на практике у них имеются свои собственные достоинства и недостатки. Данное соображение легло в основу аппарата гибридных сетей, где выводы делаются на основе аппарата нечеткой логики, но соответствующие функции принадлежности подстраиваются с использованием алгоритмов обучения нейронных сетей, например, алгоритма обратного распространения ошибки. Такие системы не только используют априорную информацию, но могут приобретать новые знания и для пользователя являются логически прозрачными.

Определение гибридной нейронной сети

Гибридная нейронная сеть — это сеть с четкими сигналами, весами и активизационной функцией, но с объединением сигналов и весов сети с использованием Г-нормы, Г-конормы или некоторых других непрерывных операций.

Входы, выходы и веса гибридной нейронной сети — вещественные числа, принадлежащие отрезку [0,1].

Примером подобной сети может служить система, имеющая следующую базу знаний:

- $П_1$,: если x_1 есть L_1, и x_2 есть L_2, и x_3 есть L_3, тогда z есть Н;
- $П_2$: если x_1 есть $Н_1$, и x_2 есть $Н_2$, и x_3 есть $L_ь$ тогда z есть M;
- $П_3$: если x_1 есть $Н_1$, и x_2 есть $Н_2$, и x_3 есть $Н_3$, тогда z есть S,

где x_1, x_2, $x3$ — входные переменные, z — выход системы, L_1, L_2, L_3 H_1, H_2, H_2, H, M, S — некоторые нечеткие множества с функциями принадлежности сигмоидно-го типа (для упрощения записи последующих выкладок функции принадлежности в данном случае обозначены так же, как и соответствующие нечеткие множества):

$$L_j(t) = \frac{1}{1 + \exp(b_j(t - c_j)}, \; H_j(t) = \frac{1}{1 + \exp(-b_j(t - c_j)}, \; j = 1, 2, 3,$$

$$H(t) = \frac{1}{1 + \exp(-b_4(t - c_4 + c_5)}, \; M(t) = \frac{1}{1 + \exp(-b_4(t - c_4)}, \; S(t) = \frac{1}{1 + \exp(b_4(t - c_4)}.$$

Для определения выходной переменной используется следующий алгоритм вывода:

1) подсчитываются значения истинности предпосылок для каждого правила:

$$\alpha_1 = L_1(a_1) \wedge L_2(a_2) \wedge L_3(a_3),$$

$$\alpha_2 = H_1(a_1) \wedge H_2(a_2) \wedge L_3(a_3),$$

$$\alpha_3 = H_1(a_1) \wedge H_2(a_2) \wedge H_3(a_3),$$

где a_1, a_2, a_3, — текущие значения входов системы;

2) для каждого правила определяются частные выходы:

$$z_1 = B^{-1}(\alpha_1) = c_4 + c_5 + \frac{1}{b_4} \ln \frac{1 - \alpha_1}{\alpha_1},$$

$$z_2 = B^{-1}(\alpha_2) = c_4 + \frac{1}{b_4} \ln \frac{1 - \alpha_2}{\alpha_2},$$

$$z_3 = B^{-1}(\alpha_3) = c_4 + \frac{1}{b_4} \ln \frac{1 - \alpha_2}{\alpha_2};$$

3) находится общий выход системы:

$$z0 = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3}{\alpha_1 + \alpha_2 + \alpha_3}.$$

Изложенный процесс иллюстрируется рис. 3.20, где оси абсцисс трех левых графиков соответствуют переменной x_1, трех следующих — переменной x_2, далее — x_3,, а трех правых — переменной вывода Z-

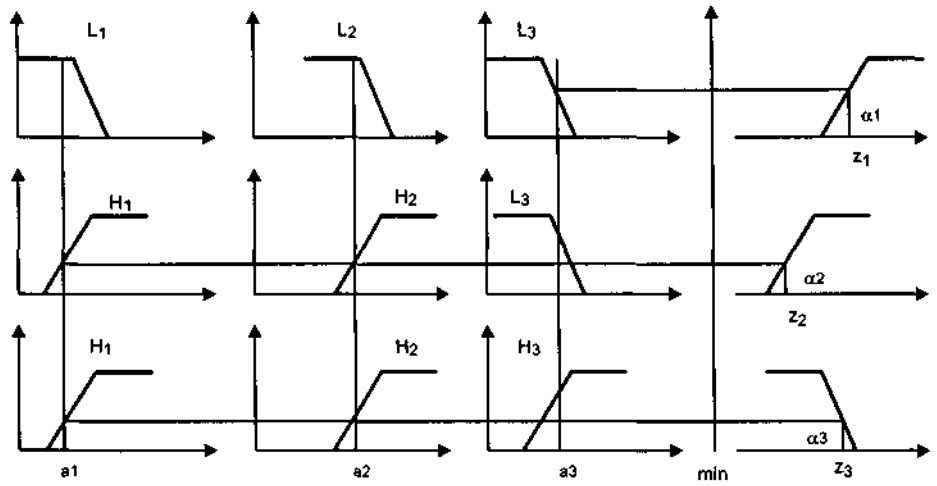


Рис. 3.20. Иллюстрация к алгоритму вывода для гибридной сети

Гибридная нейронная сеть ANFIS

Гибридная нейронная сеть, отражающая приведенный механизм вывода, представлена на рис. 3.21. Заметим, что сети с подобной архитектурой в англоязычной литературе получили название ANFIS (Adaptive Neuro-Fuzzy Inference System, то есть адаптивная нечеткая нейронная система вывода).

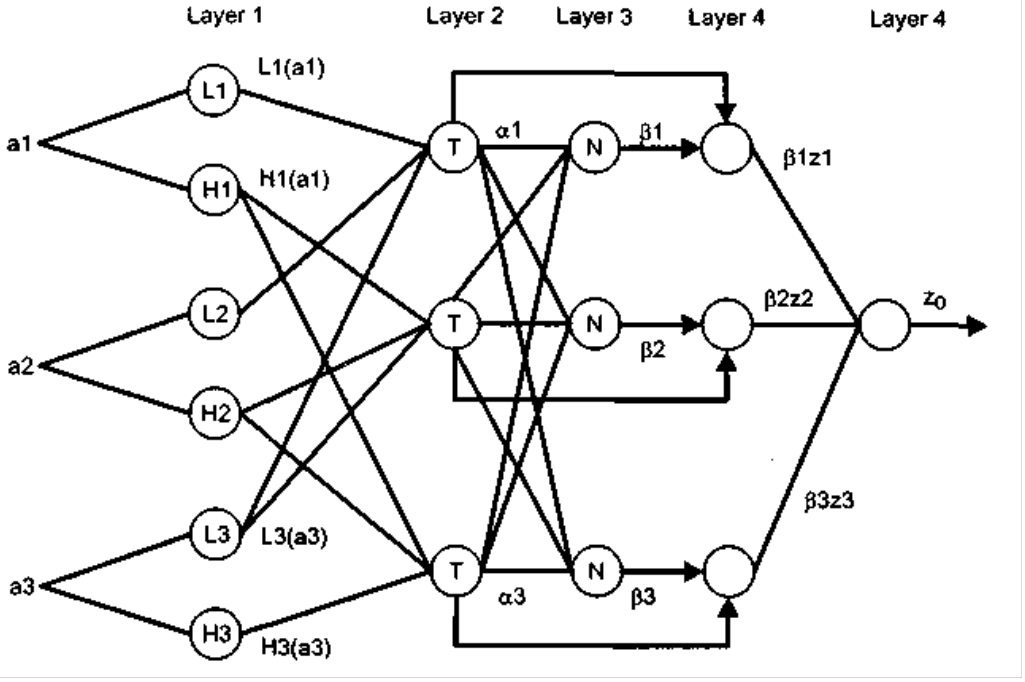


Рис. 3.21. Структура гибридной нейронной сети (архитектура ANFIS)

Данная сеть может быть описана следующим образом.

Слой 1 (Layer 1). Выходы узлов этого слоя представляют собой значения функций принадлежности при конкретных (заданных) значениях входов.

Слой 2 (Layer 2). Выходами нейронов этого слоя являются степени истинности предпосылок каждого правила базы знаний системы, вычисляемые по формулам

$$\alpha_1 = L_1(a_1) \wedge L_2(a_2) \wedge L_3(a_3),$$

$$\alpha_2 = H_1(a_1) \wedge H_2(a_2) \wedge L_3(a_3),$$

$$\alpha_3 = H_1(a_1) \wedge H_2(a_2) \wedge H_3(a_3).$$

Все нейроны этого слоя обозначены буквой T, что означает, что они могут реализовывать произвольную T-норму для моделирования операции «И».

Слой 3 (Layer 3). Нейроны этого слоя (обозначены буквой N) вычисляют величины

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}, \beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2 + \alpha_3}, \beta_3 = \frac{\alpha_3}{\alpha_1 + \alpha_2 + \alpha_3}.$$

Слой 4 (Layer 4). Нейроны данного слоя выполняют операции

$$\beta_1 z_1 = \beta_1 H^{-1}(a_1), \beta_2 z_2 = \beta_2 M^{-1}(a_2), \beta_3 z_3 = \beta_3 S^{-1}(a_3).$$

Слой 5 (Layer 5). Единственный нейрон этого слоя вычисляет выход сети:

$$z0 = \beta_1 z_1 + \beta_2 z_2 + \beta_3 z_3.$$

Корректировка параметров системы здесь производится либо в соответствии с наиболее распространенным для нейронных сетей алгоритмом обратного распространения ошибки (back propagation), либо комбинированным методом, специально разработанным для гибридных сетей.

Графический интерфейс гибридных нейронных систем

Графический интерфейс гибридных (нечетких) нейронных систем вызывается функцией anfisedit (из режима командной строки). Исполнение функции приводит к появлению окна редактора гибридных систем (ANFIS Editor, ANFIS-редактор). вид которого приведен на рис. 3.22.

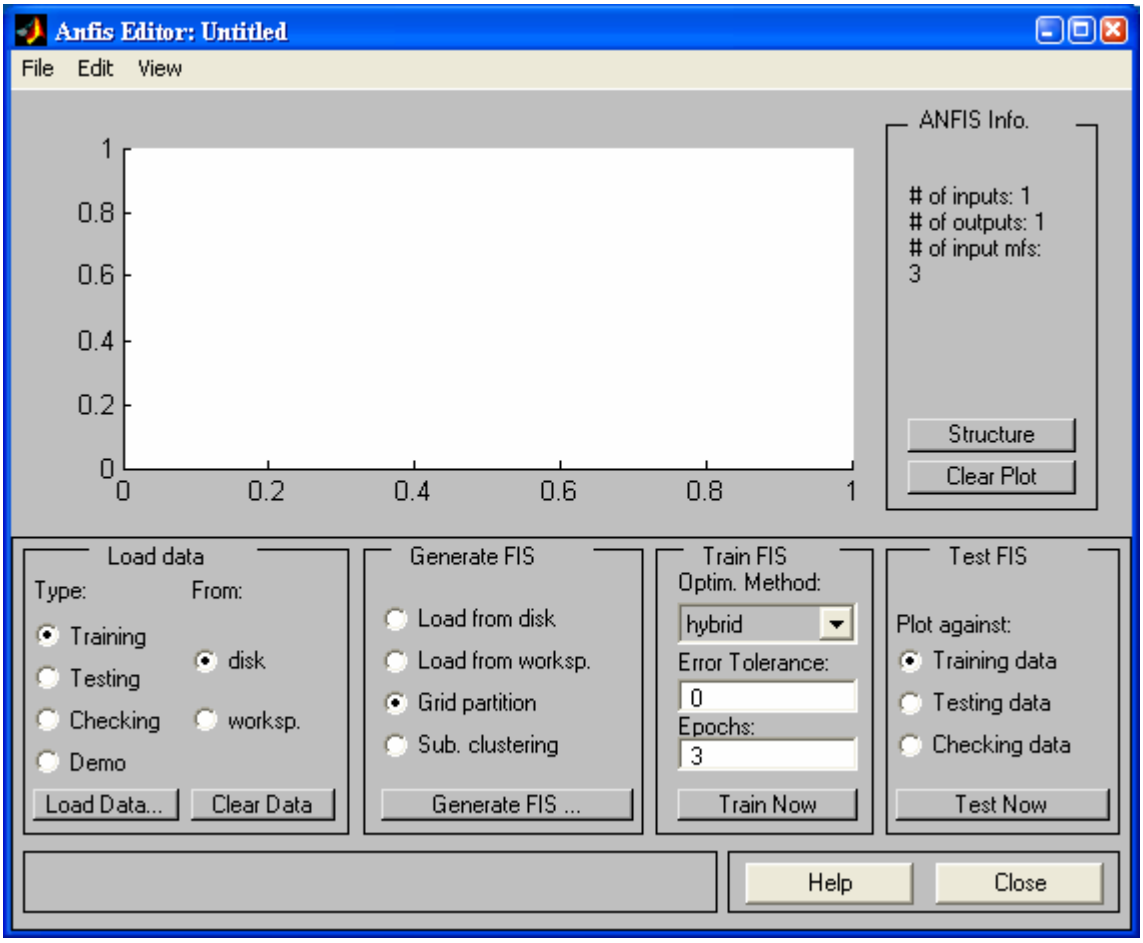


Рис 3.22. Окно редактора гибридных систем

С помощью данного редактора осуществляются создание или загрузка структуры гибридной системы, просмотр структуры, настройка ее параметров, проверка качества функционирования такой системы. Создание структуры, настройка параметров и проверка осуществляются по выборкам (наборам данных) — обучающей (Training data), проверочной (Checking data) и тестирующей (Testing data), которые

предварительно должны быть представлены в виде текстовых файлов (с расширением .dat и разделителями-табуляциями), первые столбцы которых соответствуют входным переменным, а последний (правый) — единственной выходной переменной; количество строк в таких файлах равно количеству образцов (примеров). Так, обучающая выборка, сформированная по табл. 3.2, представляется в виде

-1	**1**
-0.6	**0.36**
0.0	**0.00**
0.4	**0.16**
1	**1**

Строгих рекомендаций по объемам указанных выборок не существует, по-видимому, лучше всего исходить из принципа «чем больше, тем лучше». Обучающая и проверочная выборки непосредственно задействуются в процессе настройки параметров гибридной сети (проверочная — для выяснения ситуации, не происходит ли так называемого *переобучения* сети, при котором ошибка для обучающей последовательности стремится к нулю, а для проверочной — возрастает; впрочем, наличие проверочной выборки не является строго необходимым, оно лишь крайне желательно). Тестовая (или тестирующая) выборка применяется для проверки качества функционирования настроенной (обученной) сети.
Поясним позиции меню и опции редактора:
- Меню File и View, в общем, идентичны аналогичным меню FIS-редактора.
- Меню Edit содержит команды:
- Undo (Отменить выполненное действие);
- FIS Properties — вызов редактора нечеткой системы **(FIS Editor);**
- Membership Functions — вызов редактора функций принадлежности (Membership Function Editor);
- Rules — вызов редактора правил (Rule Editor);
- Anfis — в данном случае команда не выполняет никаких действий.

Набор опций Load data (Загрузка данных) в нижней левой части окна редактора включает в себя:
- тип (Type) загружаемых данных (для обучения — Training, для тестирования — Testing, для проверки — Checking, демонстрационные — Demo);
- место, откуда должны загружаться данные: с диска (disk) или из рабочей области MATLAB (workspace).

К данным опциям относятся две кнопки, нажатие на которые приводит к требуемым действиям — Load Data (Загрузить данные) и Clear Data (очистить, то есть стереть введенные данные).

Следующая группа опций (в середине нижней части окна ANFJS-редактора) объединена под именем Generate FIS (Создание нечеткой системы вывода). Данная группа включает в себя следующие опции:
- загрузку структуры системы с диска (Load from disk);

• загрузку структуры системы из рабочей области MATLAB (Load from worksp.);
• разбиение (деление) областей определения входных переменных (аргументов) на подобласти — независимо для каждого аргумента (Grid partition);
• разбиение всей области определения аргументов (входных переменных) на подобласти — в комплексе для всех аргументов (Sub. clustering).

Кроме того, имеется также кнопка Generate FIS, нажатие которой приводит к процессу создания гибридной системы с точностью до ряда параметров.

Следующая группа опций — Train FIS (Обучение нечеткой системы вывода) — позволяет определить метод «обучения» (Optim. Method) системы (то есть метод настройки ее параметров) — гибридный (hybrid) или обратного распространения ошибки (backpropa), установить уровень текущей суммарной (по всем образцам) ошибки обучения (Error Tolerance), при достижении которого процесс обучения заканчивается и количество циклов обучения (Epochs), то есть количество «прогонов» всех образцов (или примеров) обучающей выборки. Процесс обучения, таким образом, заканчивается либо при достижении отмеченного уровня ошибки обучения, либо после проведения заданного количества циклов.

Кнопка Train Now (Начать обучение) запускает процесс обучения, то есть процесс настройки параметров гибридной сети.

Работа с редактором гибридных нейронных систем

Работу с редактором гибридных нейронных сетей рассмотрим на примере вое становления зависимости $y = *_{\text{по}}$ данным табл. 3.2. Предположим что эти дан ные сохранены в файле Proba.dat. Создание и проверку системы как и раньше проведем по этапам.
1. В окне ANFIS-редактора выберем тип загружаемых данных Traininq и нажмем кнопку Load data. В последующем стандартном окне диалога укажем местоположение и имя файла. Его открытие приводит к появлению в графической части окна редактора набора точек, соответствующих введенным данным (рис.3.23)

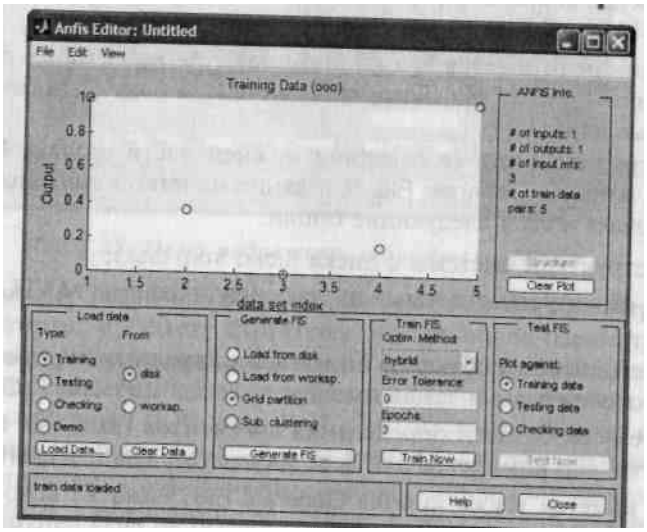


Рис. 3.23. Окно ANFIS-редактора после загрузки обучающей выборки

2. В группе опций Generate FIS по умолчанию активизирован вариант Grid partition Не будем его изменять и нажмем кнопку Generate FIS, после чего появится диалоговое окно (рис.3.24) для задания числа и типов функций принадлежности.
Сохраним все Установки по умолчанию, согласившись с ними нажатием кнопки ОК. Произойдет возврат в основное окно ANFIS-редактора Теперь структура гибридной сети создана, и её графический вид можно посмотреть помощью кнопки Structure (рис. 3.25).
3. Перейдем к опциям Train FIS. Не будем менять задаваемые по умолчанию метод настройки параметров (hybrid - гибридный) и уровень ошибки (0), но количество циклов обучения изменим на 100, после чего нажмем кнопку запуска процесса обучения (Train Now). Получившийся результат в виде графика ошибки сети в зависимости от числа проведенных циклов обучения (из которого следует, что фактически обучение закончилось после 60 циклов) представлен на рис. 3.26.

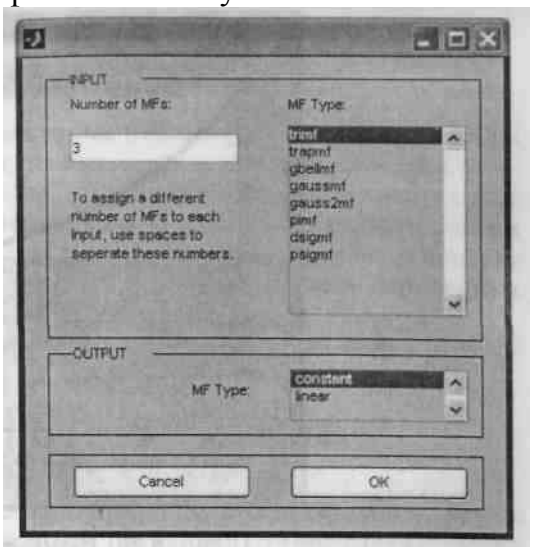


Рис. 3.24. Окно задания функций принадлежности

4. Теперь нажатием кнопки Test Now можно начать процесс тестирования обученной сети, но, поскольку использовалась только одна (обучающая) выборка, ничего особенно интересного ожидать не приходится. Действительно, выход обученной системы практически совпадает с точками обучающей выборки (рис. 3.27).

5. Сохраним разработанную систему на диске в файл с именем Probal (с расширением .fis) и для исследования разработанной системы средствами FIS-редактора из командной строки MATLAB выполним команду fuzzy, а затем через пункты меню File/Import/From disk откроем созданный файл. С созданной системой можно теперь выполнять все приемы редактирования (изменение имен переменных и т. п.) и исследования, которые были рассмотрены выше. Здесь нетрудно, кстати, убедиться, что качество аппроксимации существенно не улучшилось — слишком мало данных.

Что можно сказать про эффективность использования гибридных систем (и ANFIS-редактора)?

В данном случае используется только одна выходная переменная, всем правилам приписывается один и тот же единичный вес. Вообще говоря, возникают зна-

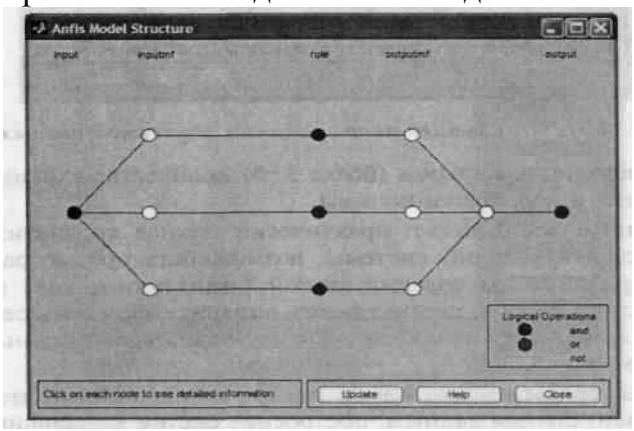


Рис. 3.25. Структура созданной гибридной сети

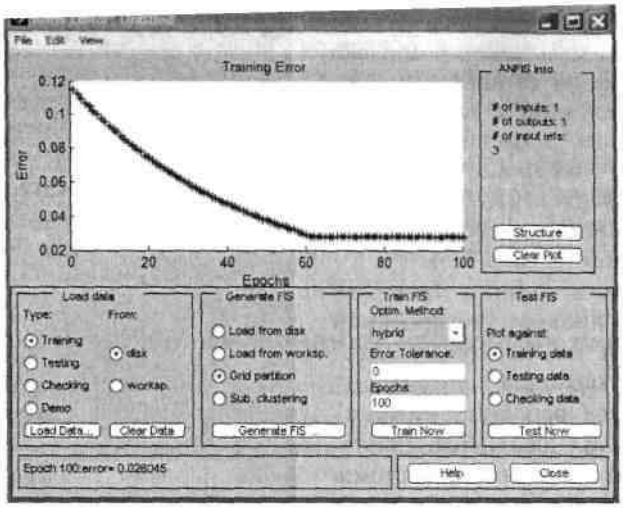


Рис. 3.26. Результат обучения сети

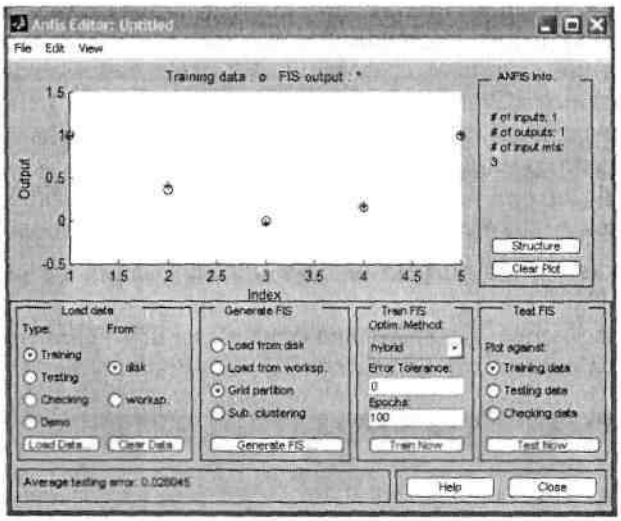


Рис. 3.27. Результат тестирования обученной системы

чительные проблемы при большом (более 5—6) количестве входных переменных. Это — **ограничения и недостатки подхода.**

Его несомненные достоинства: практически полная автоматизация процесса создания нечеткой (гибридной) системы, возможность просмотра сформированных правил и придания им содержательной (лингвистической) интерпретации, что позволяет, кстати говоря, рассматривать аппарат гибридных сетей как средство извлечения знаний из баз данных и существенно отличает данные сети от классических нейронных.
Рекомендуемая область применения: построение аппроксиматоров зависимостей по экспериментальным данным, построение систем классификации (в случае бинарной или дискретной выходной переменной), изучение механизма явлений.

Нейронечеткие системы

Гибридизация нейронных сетей с нечеткой логикой позволяет существенно повысить эффективность работы таких нейронечетких систем за счет того, что недостатки, присущие одной из технологий, компенсируются преимуществами другой. В частности, ИНС хорошо распознают образы, но процесс работы обученной сети сложен для понимания. В то же время системы НЛ хорошо объясняют выводы, но имеют ограничения на количество входных переменных. Вследствие этого возможно построение гибридных нейронечетких систем, в которых выводы формируются на основе НЛ, а ФП подстраиваются с помощью ИНС. Преимущество таких систем очевидно: построенная структура является логически прозрачной. Рассмотрим вначале обычный многослойный персептрон, в который будем вводить нечеткие величины на входной и выходной слои [6]. Левая часть правил систем НЛ (предпосылка) генерируется при обратном распространении через сетевые веса, а правая часть (заключение) — через выходы сети. Алгоритм обучения, используемый для нечеткого персептрона, является расширением традиционного метода ОРО,

только вместо среднеквадратичной ошибки применяется нечеткая ошибка, которая больше подходит для данных, включающих нечеткости между классами. Входные данные фазифицируются посредством ФП.
В отличие от обычного персептрона эта модель способна иметь дело со входными переменными, выраженными в лингвистической форме. Гибридная сеть, как и обычная ИНС, проходит через две стадии: обучение и тестирование. На стадии обучения используется супервизорное обучение для присвоения выходных значений ФП из интервала [0,1] обучающим векторам. Следовательно, каждому нейрону выходного слоя может быть присвоено ненулевое значение ФП вместо выбора единственного нейрона с наибольшей активацией. Это позволяет моделировать нечеткие данные, когда признаковое пространство включает перекрывающиеся классы так, что точка образа может принадлежать более чем одному классу с ненулевой ФП. При обучении каждая ошибка в заданной ФП вводится обратно в сеть и на обратном проходе изменяются веса сети. Ошибка обратного распространения вычисляется по отношению к каждому требуемому выходу, которым является значение ФП, определяющее степень принадлежности входного вектора к рассматриваемому классу. После ряда циклов такая сеть будет сходиться к решению с минимальной ошибкой.

Алгоритм обучения можно описать следующим образом.
Рассмотрим сеть обратного распространения, в которой общий вход x_j^{h+1}, принятый нейроном; в слое $h + 1$, определяется как:

$$x_j^{h+1} = \sum_i y_i^h w_{ji}^h - \theta_j^{h+1}, \qquad (114)$$

где y_i^h — состояние i-го нейрона в предшествующем скрытом слое;
w_{ji}^h — вес от нейрона i в слое h к j-му нейрону в слое h + 1;
Θ_j^{h+1} — порог j-го Нейрона в слое $h + 1$.
Выход нейрона в любом слое, кроме первого, определяется монотонной нелинейной функцией в виде сигмоиды от общего входа слоя и имеет вид:

$$y_j^h = 1/[1 + \exp(-x_j^h)]. \qquad (115)$$

Вследствие того, что выходные нейроны сети представляют значения ФП в диапазоне от 0 до 1 и не являются бинарными, обычная функция ошибки не может быть использована. Вместо этого здесь используется функция ошибки в виде:

$$E_p = \frac{1}{2}\sum_p \sum_k \mu(x_p)(y_{pk} - \theta_{pk})^2, \qquad (116)$$

где $\mu(x_p)$ — ФП для входного вектора x_p .

Алгоритм, как и в классическом варианте нейронных сетей, состоит из таких же шагов и принципиально не отличается от описанного ранее (см. гл. 1), за исключением, естественно, вида функции ошибки.

В противоположность описанной разновидности нейронечеткой системы была предложена адаптивная нечеткая нейронная система вывода (Adaptive Neuro-Fuzzy Inference System — ANFIS) [2]. В этой системе использовался алгоритм Сугено в качестве системы вывода. При рассмотрении ANFIS для упрощения допустим, что нечеткая система имеет два входа *x, y* и один выход f. Пусть база правил содержит всего лишь два правила вида «если..., то...»:
П1: если *x* есть A_1 и *y* есть B_1, то $f_1 = p_1x+q_1y;$
П2: если *x* есть A_2 и *y* есть B_2, то $f_2 = p_2x + q_2y$.

Соответствующая системе ANFIS архитектура нейронной сети показана на рис. 2.26.

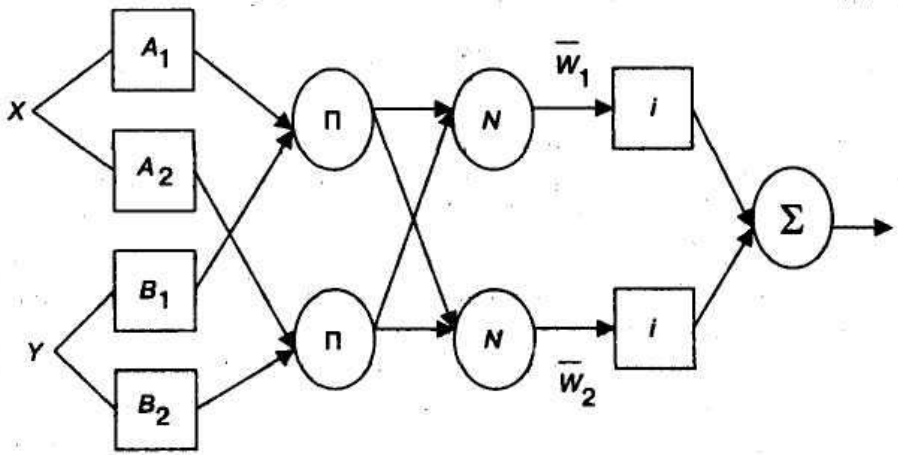


Рис. 2.26. Архитектура нейронной сети ANFIS

Каждый слой сети выполняет следующие функции.
Слой 1.
Каждый нейрон этого слоя (обозначен квадратом на рис. 2.26) есть величина, равная

$$O_i^1 = \mu_{A_i}(x), \tag{117}$$

где *x* — вход в нейрон *i;*
A_i — лингвистическая переменная (малая, большая и т. п.), ассоциированная с функцией нейрона;
O_i'— ФП переменной A_i, определяющая степень, с которой *x* удовлетворяет переменной A_i. В первоначально предложенном варианте ФП μ_{Ai} *(x)* были выбраны в виде колоколообразной кривой с максимумом, равным 1, и минимумом, равным 0. Формулы, отображающие такую кривую, имеют вид:

$$\mu_{A_i}(x) = \left\{1 + \left[\left(\frac{x - c_i}{a_i}\right)^2\right]^{b_i}\right\}^{-1}; \tag{118}$$

$$\mu_{A_i}(x) = \exp\left[-\left(\frac{x - c_i}{a_i}\right)^2\right], \tag{119}$$

где a_i, b_i, c_i — устанавливаемые параметры.

При изменении этих параметров соответственно будут меняться функции $\mu_{Ai}\ (x)$, определяющие различные формы ФП переменной A_i.
Иными словами, можно сказать, что выходы нейронов этого слоя представляют собой значения ФП при конкретных значениях входных переменных.

Слой 2.

Каждый нейрон этого слоя (обозначен кружком П) перемножает входящие сигналы и посылает полученное произведение в следующий слой. Например,

$$W_i = \mu_{A_i}(x) \times \mu_{B_i}(x),\ i = 1; 2. \tag{120}$$

Каждый выход этого слоя определяет активизацию правила.

Слой 3.

Нейрон этого слоя (обозначен кружком *N)* вычисляет отношение величин:

$$\overline{W_i} = \frac{W_i}{W_1 + W_2},\ i = 1; 2. \tag{121}$$

Выходы этого слоя могут быть названы нормированной активацией правила.

Слой 4.

Нейрон с индексом *i* (обозначен квадратом) данного слоя выполняет следующую операцию

$$O_i^4 = \overline{W_i} \times f_i = \overline{W_i} \times (p_i x + q_i y). \tag{122}$$

Слой 5.

Единственный нейрон этого слоя (обозначен кружком) вычисляет итоговый выход как сумму всех входящих сигналов

$$O_i^5 = \sum_i \overline{W_i} \times f_i. \tag{123}$$

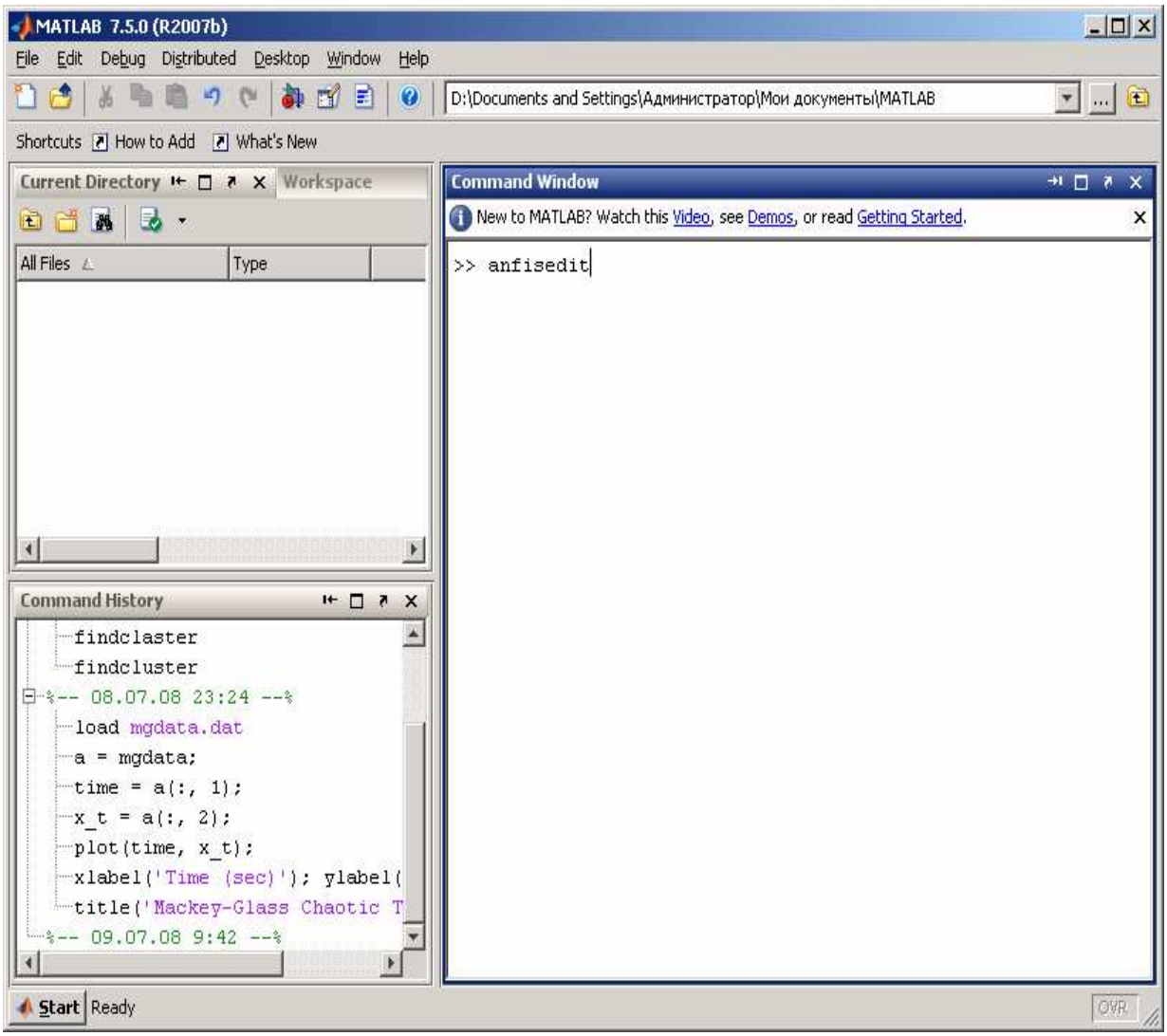
Такая адаптивная сеть функционально эквивалентна нечеткой системе вывода Сугено. Разработчик такой системы Дж. Янг (J. S.Jang) предложил применять ее для имитации функций, предсказанию в хаотической динамике, обработке сигналов. Корректировка параметров системы в такой сети производится посредством специально разработанного гибридного обучающего алгоритма.
Отметим, что выход нейрона во втором слое ANFIS представляет собой операцию «логического И» ФП каждой лингвистической переменной первого слоя. Каждый слой ANFIS эквивалентен шагу вычислений нечеткой системы вывода. Функция активации в каждом слое соответствует шагу нечеткого вывода. База знаний нечетких правил эквивалентна знаниям, находящимся в соответствующей ANFIS. При

разработке такой системы предполагалась фиксированной ее структура, а идентификация параметров осуществлялась гибридным правилом. Другая важная проблема структурной идентификации, которая касается выбора приемлемого разделения входного пространства и количества ФП на каждом входе, еще не решена. Эффективное разделение входного пространства может уменьшить число правил и, следовательно, увеличить скорость работы при обучении и применении.

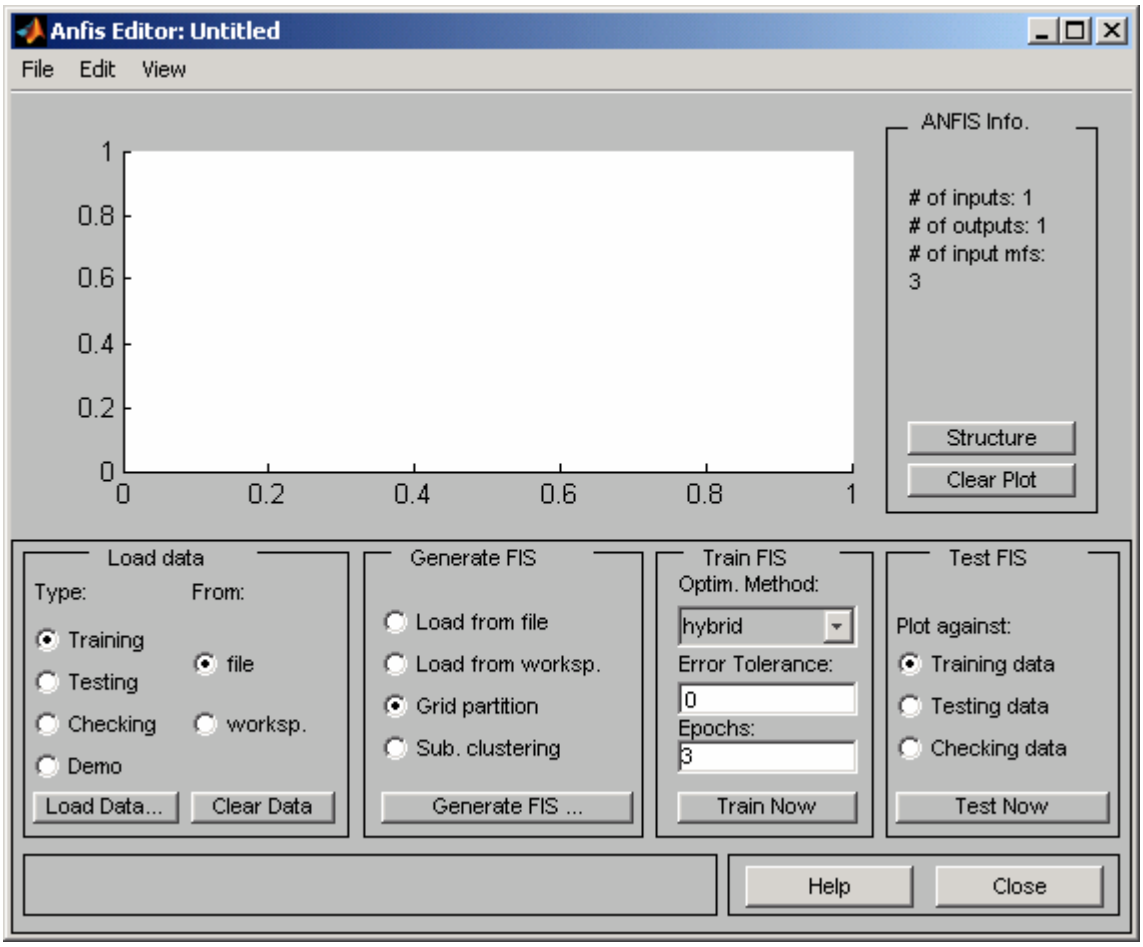
Matlab - модуль Fuzzy Logic Toolbox

- Далее рассматривается система **Matlab** - модуль **Fuzzy Logic Toolbox**

Графический интерфейс гибридных (нечетких) нейронных систем вызывается функцией anfisedit (из режима командной строки).



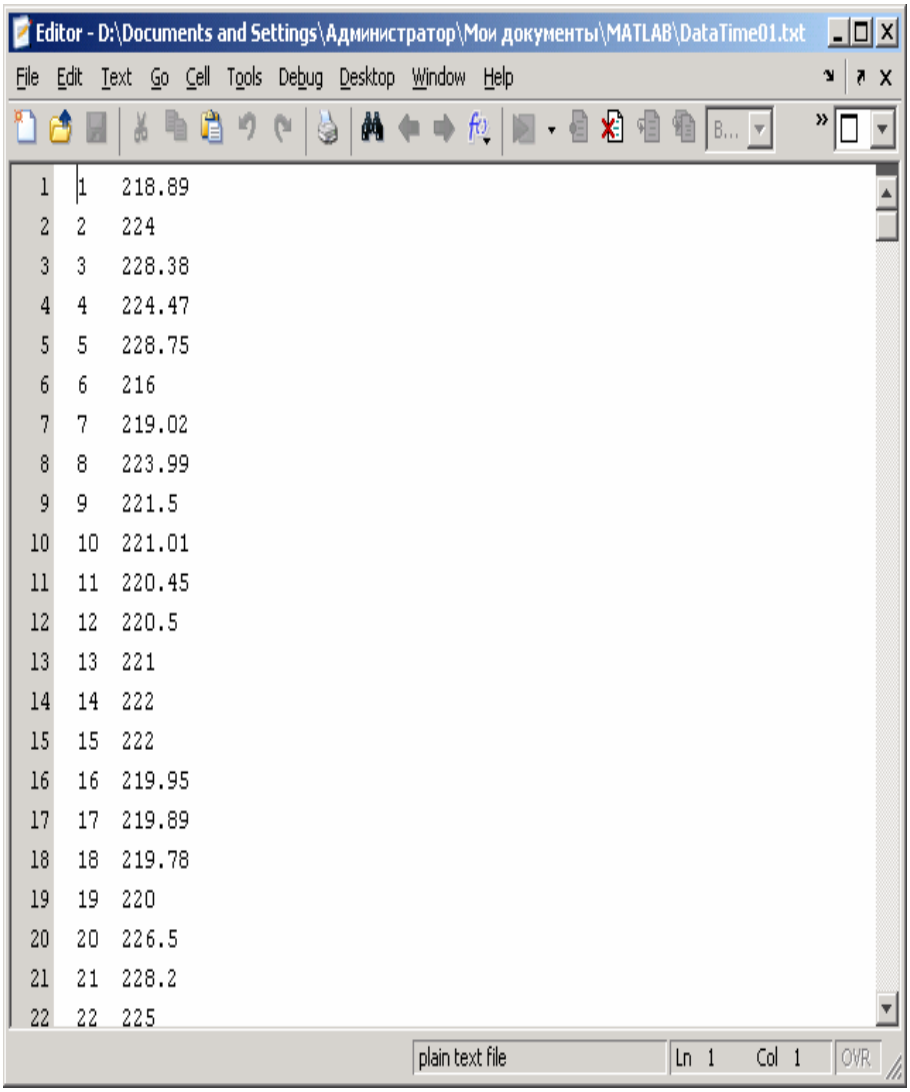
Окно редактора гибридных систем



Функции редактора

- С помощью данного редактора осуществляются создание или загрузка структуры гибридной системы, просмотр структуры, настройка ее параметров, проверка качества функционирования такой системы. Создание структуры, настройка параметров и проверка осуществляются по выборкам (наборам данных) — обучающей (Training data), проверочной (Checking data) и тестирующей (Testing data), которые предварительно должны быть представлены в виде текстовых файлов (с расширением .dat и разделителями-табуляциями), первые столбцы которых соответствуют входным переменным, а последний (правый) — единственной выходной переменной; количество строк в таких файлах равно количеству объектов (наблюдений).

Пример набора данных



Рекомендации по объему выборок

- Строгих рекомендаций по объемам указанных выборок не существует, по-видимому, лучше всего исходить из принципа «чем больше, тем лучше». Обучающая и проверочная выборки непосредственно задействуются в процессе настройки параметров гибридной сети (проверочная — для выяснения ситуации, не происходит ли так называемого *переобучения* сети, при котором ошибка для обучающей последовательности стремится к нулю, а для проверочной — возрастает; впрочем, наличие проверочной выборки не является строго необходимым, оно лишь крайне желательно). Тестовая (или тестирующая) выборка применяется для проверки качества функционирования настроенной (обученной) сети.

Позиции меню и опции редактора

- • Меню File и View, в общем, идентичны аналогичным меню FIS-редактора.
- • Меню Edit содержит команды:
- • Undo (Отменить выполненное действие);
- • FIS Properties — вызов редактора нечеткой системы **(FIS Editor);**
- • Membership Functions — вызов редактора функций принадлежности (Membership Function Editor);

- • Rules — вызов редактора правил (Rule Editor);
- • Anfis — в данном случае команда не выполняет никаких действий.

Набор опций Load data (Загрузка данных) в нижней левой части окна редактора включает в себя:

- тип (Type) загружаемых данных (для обучения — Training, для тестирования — Testing, для проверки — Checking, демонстрационные — Demo);
- место, откуда должны загружаться данные: с диска (disk) или из рабочей области MATLAB (workspace).
- К данным опциям относятся две кнопки, нажатие на которые приводит к требуемым действиям — Load Data (Загрузить данные) и Clear Data (очистить, то есть стереть введенные данные).

Группа опций под именем Generate FIS (Создание нечеткой системы вывода).

- загрузку структуры системы с диска (Load from disk);
- загрузку структуры системы из рабочей области MATLAB (Load from worksp.);
- разбиение (деление) областей определения входных переменных (аргументов) на подобласти — независимо для каждого аргумента (Grid partition);
- разбиение всей области определения аргументов (входных переменных) на подобласти — в комплексе для всех аргументов (Sub. clustering).
- Кроме того, имеется также кнопка Generate FIS, нажатие которой приводит к процессу создания гибридной системы с точностью до ряда параметров.

Группа опций — Train FIS (Обучение нечеткой системы вывода)

- позволяет определить метод «обучения» (Optim. Method) системы (то есть метод настройки ее параметров) — гибридный (hybrid) или обратного распространения ошибки (backpropa), установить уровень текущей суммарной (по всем образцам) ошибки обучения (Error Tolerance), при достижении которого процесс обучения заканчивается и количество циклов обучения (Epochs), то есть количество «прогонов» всех образцов (или примеров) обучающей выборки. Процесс обучения, таким образом, заканчивается либо при достижении отмеченного уровня ошибки обучения, либо после проведения заданного количества циклов.

Эффективность использования гибридных систем

- **Его несомненные достоинства:** практически полная автоматизация процесса создания нечеткой (гибридной) системы, возможность просмотра сформированных правил и придания им содержательной (лингвистической) интерпретации, что позволяет, кстати говоря, рассматривать аппарат гибридных сетей как средство извлечения знаний из баз данных и существенно отличает данные сети от классических нейронных.
- **Ограничения и недостатки подхода:** незначительное число входных переменных (5-10).

Рекомендуемая область применения:

- построение аппроксиматоров зависимостей по экспериментальным данным, построение систем классификации (в случае бинарной или дискретной выходной переменной), изучение механизма явлений.

Литература

1. Круглов В.В., Дли М.И., Голунов Р.Ю. Нечеткая логика и искусственные нейронные сети: Учеб.пособие.- М.: Изд-во физ.-мат. литературы, 2001.
2. Аверкин А.Н., Батыршин И.З., Блишун А.Ф., Тарасов В.Б., Силов В.Б. Нечеткие множества в моделях управления и искусственного интеллекта/Под ред.Д.А.Поспелова.-М.:Наука,1986.
3. Батыршин И.З. Основные операции нечеткой логики и их обобщения.- Казань: Отечество, 2001.
4. Прикладные нечеткие системы/Под ред. Т.Тэрано, К.Асаи, М.Сугано.-М.:Мир, 1993.
5. Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры: Учеб. пособие для вузов.- М.: Изд-во МГТУ им. Н.Э.Баумана, 2002 (Сер. Информатика в техническом университете).

Лекция 13. Постпроцессорная обработка автоматической классификации (кластеризации, распознавания)

При разработке искусственных когнитивных систем образы, полученные в результате кластеризации, Классификации, распознавания нуждаются в

- трактовке,
- объяснении,
- описании иерархического строения и структуры всего множества
- и отдельно – каждого из них,
- выявлении характерных черт,
- существенных признаков,
- типовых свойств,
- описании эталонных объектов
- и отличительных черт.

В этом заключается цель постпроцессорной обработки.

Для её проведения наиболее подходят пакеты программ математической статистики, такие, как Excel, BMDP, SPSS, SAS, Статистика, Deductor и другие. От нейросетевых имитаторов требуется только вывод соответствующих комбинаций входных и выходных переменных в приемлемом для обработки виде.

Постпроцессорная обработка операций по автоматической кластеризации, классификации, распознавания образов использование средств типа Описательных статистик позволяет сформулировать типовые и отличительные черты полученных кластеров.

Выполнение статистического анализа

В меню Сервис пакета Excel выберите команду Анализ данных.

Если эта команда недоступна, загрузите пакет анализа:

В меню Сервис выберите команду Надстройки.

В списке надстроек выберите Пакет анализа и нажмите кнопку OK.

Выполните инструкции программы установки, если это необходимо.

Выберите нужную функцию в диалоговом окне Анализ данных и нажмите кнопку ОК.

Установите параметры анализа в соответствующем диалоговом окне.

Чтобы получить дополнительные сведения о параметрах, нажмите в диалоговом окне кнопку Справка.

Основной состав средств анализа данных.

В состав Microsoft Excel входит набор средств анализа данных (так называемый пакет анализа), предназначенный для решения сложных статистических и инженерных задач. Для анализа данных с помощью этих инструментов следует указать входные данные и выбрать параметры; анализ будет выполнен с помощью подходящей статистической или инженерной макрофункции, а результат будет помещен в выходной диапазон. Другие средства позволяют представить результаты анализа в графическом виде.

- Дисперсионный анализ
- Корреляционный анализ
- Ковариационный анализ
- Описательная статистика
- Экспоненциальное сглаживание
- Двухвыборочный F-тест для дисперсии
- Анализ Фурье
- Гистограмма
- Скользящее среднее
- Генерация случайных чисел
- Ранг и персентиль
- Регрессия
- Выборка
- T-тест
- Z-тест

Описательная статистика

Это средство анализа служит для создания одномерного статистического отчета, содержащего информацию о центральной тенденции и изменчивости входных данных.

Статистический отчёт включает в себя 16 характеристик, часть из которых – не обязательная, заказывается дополнительно:

Среднее,
Стандартная ошибка (среднего),
Медиана,
Мода,
Стандартное отклонение,
Дисперсия выборки,
Эксцесс,
Асимметричность,
Интервал,
Минимум,
Максимум,
Сумма,

Счет,
Наибольшее (#),
Наименьшее (#),
Уровень надежности

Пример постпроцессинга и вывода о некоторых существенных различиях полученных кластеров.

Получим описательную характеристику для каждого из трех кластеров.

SETOSA

SEPALLEN		*SEPALWID*		*PETALLEN*		*PETALWID*	
Среднее	4,985366	Среднее	3,434146	Среднее	1,448778	Среднее	0,246341
Стандартная ошибка	0,054946	Стандартная ошибка	0,056005	Стандартная ошибка	0,028181	Стандартная ошибка	0,017143
Медиана	5	Медиана	3,4	Медиана	1,4	Медиана	0,2
Мода	5	Мода	3,4	Мода	1,4	Мода	0,2
Стандартное отклонение	0,351825	Стандартное отклонение	0,358894	Стандартное отклонение	0,180447	Стандартное отклонение	0,109767
Дисперсия выборки	0,123778	Дисперсия выборки	0,128805	Дисперсия выборки	0,032561	Дисперсия выборки	0,012049
Эксцесс	-0,123602	Эксцесс	0,219736	Эксцесс	1,147706	Эксцесс	1,785837
Асимметричность	0,163851	Асимметричность	0,784916	Асимметричность	0,235467	Асимметричность	1,349605
Интервал	1,5	Интервал	1,5	Интервал	0,9	Интервал	0,5
Минимум	4,3	Минимум	2,9	Минимум	1	Минимум	0,1
Максимум	5,8	Максимум	4,4	Максимум	1,9	Максимум	0,6
Сумма	204,4	Сумма	140,8	Сумма	59,4	Сумма	10,1
Счет	41	Счет	41	Счет	41	Счет	41

VERSICOL

SEPALLEN		*SEPALWID*		*PETALLEN*		*PETALWID*	
Среднее	5,834783	Среднее	2,717391	Среднее	4,326087	Среднее	1,354348
Стандартная ошибка	0,062802	Стандартная ошибка	0,041969	Стандартная ошибка	0,077237	Стандартная ошибка	0,035264
Медиана	5,8	Медиана	2,75	Медиана	4,4	Медиана	1,3
Мода	5,6	Мода	2,8	Мода	4,5	Мода	1,3
Стандартное отклонение	0,425946	Стандартное отклонение	0,284647	Стандартное отклонение	0,523847	Стандартное отклонение	0,239171
Дисперсия выборки	0,181403	Дисперсия выборки	0,081024	Дисперсия выборки	0,274415	Дисперсия выборки	0,057203
Эксцесс	-0,229707	Эксцесс	0,274915	Эксцесс	0,570203		

	0,07287		79		87		75
	-		-		-		
Асимметричность	0,26632	Асимметричность	0,33491	Асимметричность	0,14177	Асимметричность	0,611184
Интервал	1,8	Интервал	1,4	Интервал	2,6	Интервал	1
Минимум	4,9	Минимум	2	Минимум	3	Минимум	1
Максимум	6,7	Максимум	3,4	Максимум	5,6	Максимум	2
Сумма	268,4	Сумма	125	Сумма	199	Сумма	62,3
Счет	46	Счет	46	Счет	46	Счет	46

VIRGINIC

SEPALLEN		*SEPALWID*		*PETALLEN*		*PETALWID*	
	6,75714		3,06904		5,57857		2,01904
Среднее	3	Среднее	8	Среднее	1	Среднее	8
Стандартная ошибка	0,08362 7	Стандартная ошибка	0,04397 3	Стандартная ошибка	0,09148	Стандартная ошибка	0,04259 8
Медиана	6,7	Медиана	3	Медиана	5,6	Медиана	2
Мода	6,7	Мода	3	Мода	5,6	Мода	1,8
Стандартное отклонение	0,54196 7	Стандартное отклонение	0,28498	Стандартное отклонение	0,59285 8	Стандартное отклонение	0,27606 8
Дисперсия выборки	0,29372 8	Дисперсия выборки	0,08121 4	Дисперсия выборки	0,35148 1	Дисперсия выборки	0,07621 4
	-				-		-
Эксцесс	0,66299	Эксцесс	1,00446 6	Эксцесс	0,50322	Эксцесс	0,74594
	-						-
Асимметричность	0,50541 5	Асимметричность	0,46854 5	Асимметричность	0,36720 4	Асимметричность	0,14953
Интервал	2	Интервал	1,3	Интервал	2,2	Интервал	1,1
Минимум	5,9	Минимум	2,5	Минимум	4,7	Минимум	1,4
Максимум	7,9	Максимум	3,8	Максимум	6,9	Максимум	2,5
Сумма	283,8	Сумма	128,9	Сумма	234,3	Сумма	84,8
Счет	42	Счет	42	Счет	42	Счет	42

4. Сравним некоторые характеристики трех кластеров, чтобы понять их различия и сходства. Посмотрим на средние значения характеристик SEPALLEN, SEPALWID, PETALLEN, PETALWID каждого и трех кластеров:

	Setosa	***Versicolor***	***Virginica***
Sepallen	5	5.9	6.6
Sepalwid	3.4	2.8	3
Petallen	1.4	4.2	5.6
Petalwid	0.2	1.3	2

Заметим, что, как и сказано в условии, ***класс Setosa выделяется из трех сразу***. В самом деле, наиболее яркое отличие видно в характеристике *Petalwid*, где ее значение близко к нулю, в то время как у двух других кластеров они 1.3 и 2 соответственно. Такое же резкое отличие видно и в характеристике *Petallen*, где значение ***Setosa*** 1.4, в то время как в двух других кластерах в 3-4 раза больше, соответственно. По характеристике *Sepalwid* этот класс выделить не так-то и просто, так как оба кластера кучкуются возле значения 3. Чуть легче, но все равно не такое очевидное различие, видно в *Sepallen*, где ***Setosa*** имеет характеристику 5, а остальные два 6 и больше.

Имея эти четыре характеристики выделить первый класс не составляет труда, что нельзя сказать о двух других. Рассмотрим их.

Характеристика *Sepalwid* в этих классах практически идентична, поэтому использовать ее при кластеризации не имеет смысла, так как это может только увеличить ошибку и привести к неправильной классификации. Остальные три характеристики *Sepallen*, *Petallen*, *Petalwid* имеют бОльшие значения в классе ***Virginica***, чем в ***Versicolor***, причем особенно отчетливо разница чувствуется по характеристике *Petallen*, где пересечение множеств ***Versicolor*** и ***Virginica*** практически равно нулю, так как разница составляет почти 1.5, тогда как в *Sepallen* и *Petalwid* она 0.7.

Таким образом, три вида цветка имеют следующие типовые сходства и различия:
1. *Ширина чашелистика* у всех трех типов в среднем одинаковая, она колеблется возле тройки, поэтому использовать это данное для кластеризации неуместно
2. Наименьшую *длину чашелистика* имеет класс ***Setosa***, в то время как два оставшихся класса имеют длину на 1 и 1.5 в среднем больше, чем у этого класса. Чисто теоретически это данное можно использовать с целью кластеризации, хотя некие пересечения множеств все-таки могут возникать, и ошибка точно будет.
3. *Ширина лепестка* – основная отличительная особенность класса ***Setosa***. Она близка к нулю в этом кластере и к 1.3 и 2 в классах ***Versicolor*** и ***Virginica***. Чтобы выделить первый класс эта характеристика хороша, но, чтобы различить два последних, – менее нужная, так как пересечения ее будут возникать.
4. Пожалуй, главная из характеристик, позволяющая различить сразу все три класса - это характеристика *длина лепестка*. Значение этой характеристики в каждом из классов имеет довольно большой интервал разницы, поэтому пересечений если и будет возникать, то пренебрежимо малое количество. В первом классе ее значение около 1.4, в то время как в следующих двух классах соответственно в 3 и в 4 раза больше.

Таким образом, для легкой кластеризации достаточно взять одну характеристику, при этом полученный результат будет нисколько не хуже, а даже лучше, чем с использованием всех четырех данных.

Особенности постпроцессорной обработки в различных нейроимитаторах.

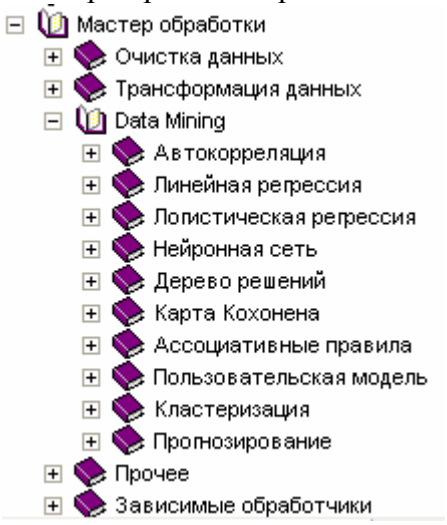
В Deduktor после загрузки пакета доступен только один элемент управления - Мастер импорта:

С помощью которого можно только загрузить в Deductor текстовый файл. При успешном завершении импорта файла определяется способ отображения введённых данных.

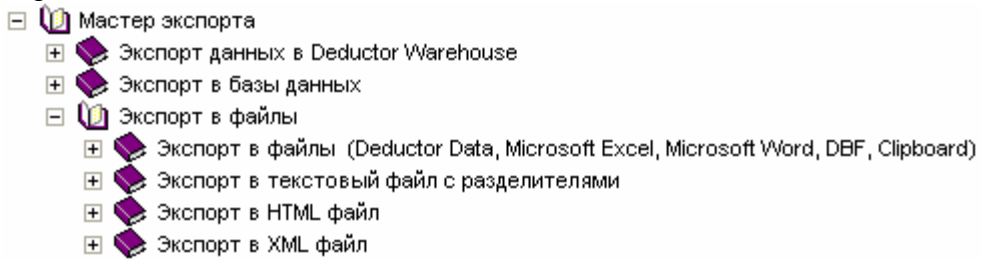
Для выборки данных, полученных в результате обработки имеющимися программными средствами или с помощью карт Кохонена, доступны следующие виды отображения:

- Обучающий набор;
- Диаграмма рассеяния;
- Самоорганизующаяся карта;
- Что-если;
- Таблица сопряженности.
- Таблица;
- Статистика;
- Диаграмма;
- Гистограмма;

Мастер обработки предоставляет программы математической статистики:



Мастер экспорта предоставляет различные средства для вывода информации после обработки:



В Пермском симуляторе

Программа «Нейросимулятор» позволяет создавать и применять нейронные сети перцептронного типа.

Программа «Нейросимулятор» может работать в четырех режимах: «Проектирование сети», «Обучение», «Проверка» и «Прогноз». Переход из одного режима в другой осуществляется путем нажатия соответствующих закладок, расположенных в верхней части рабочего окна.

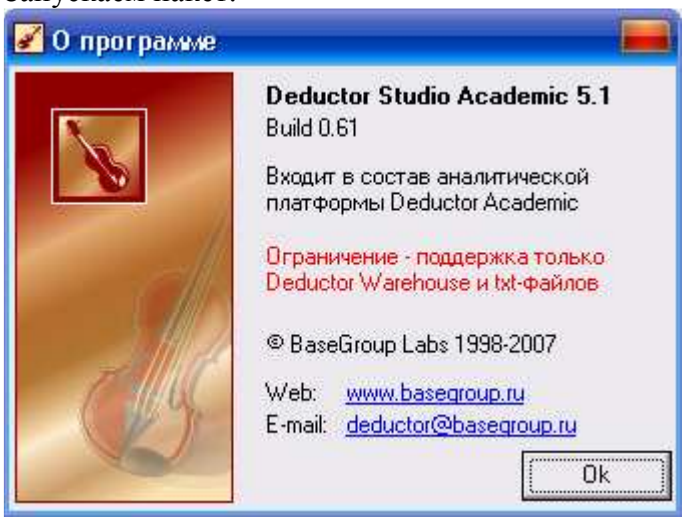
Обмен данными с приложением Excel проводится по командам.

Лекция 14. Отображение данных в разных пакетах.

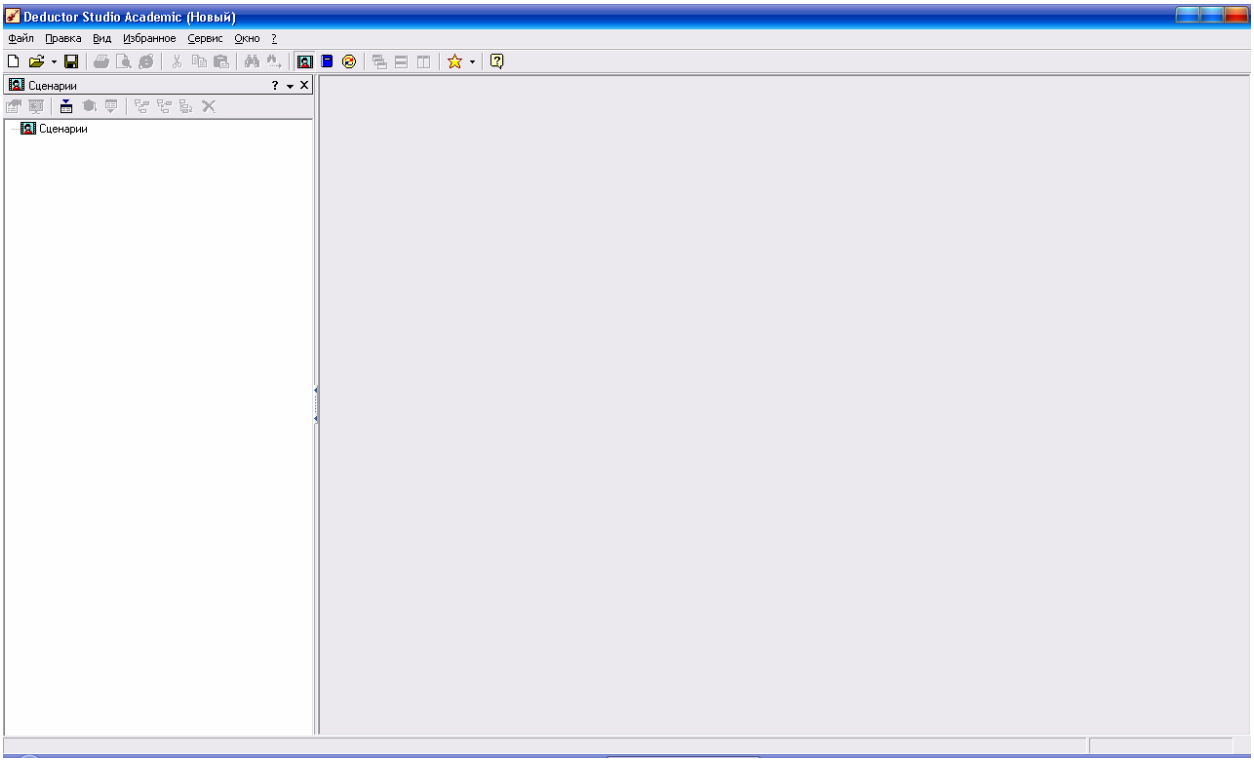
Deductor.

1. Подготовка текстового файла для нейросетевого исследования.

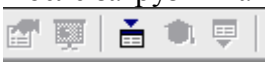
Запускаем пакет:



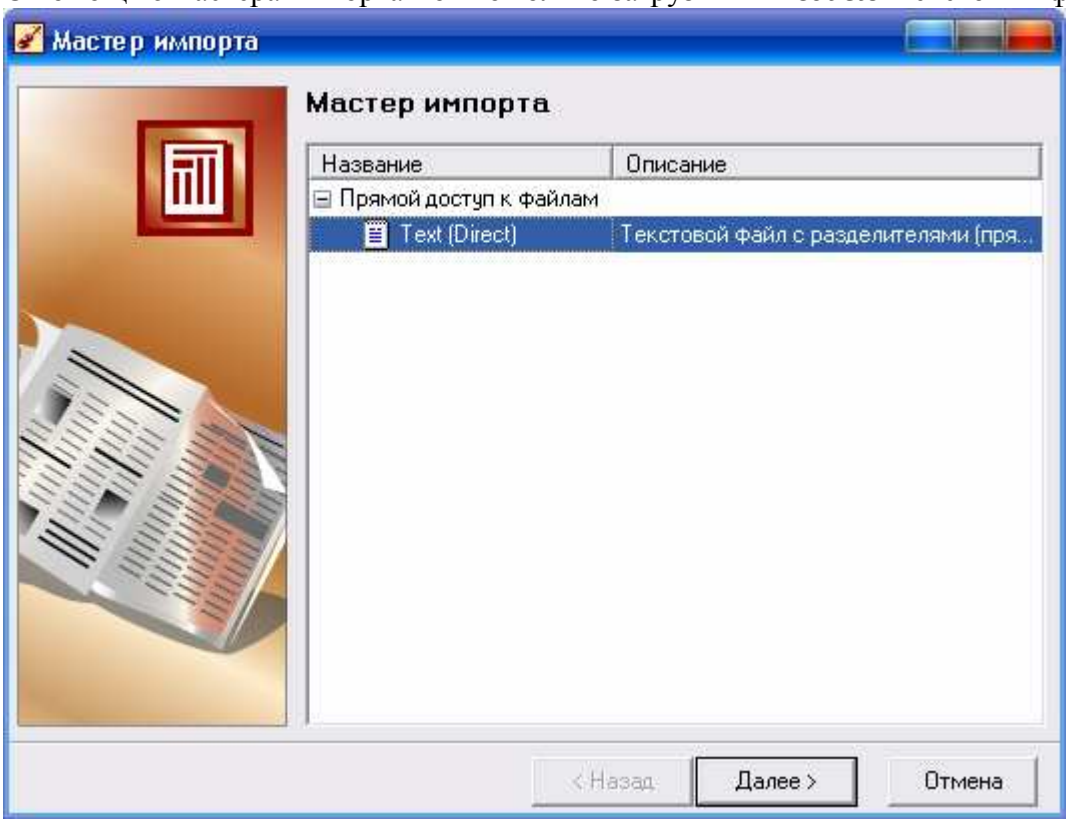
Появляется первое окно, предоставляющее очень мало возможностей:



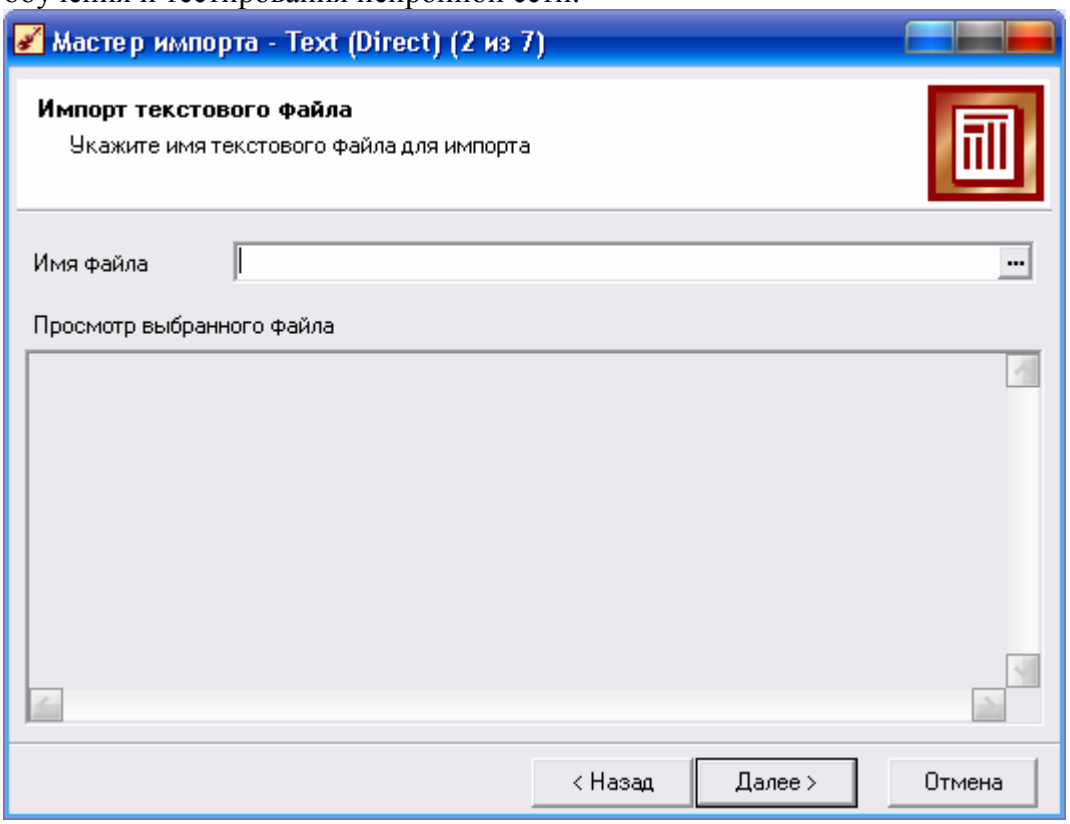
После загрузки пакета доступен только один элемент управления - Мастер импорта:



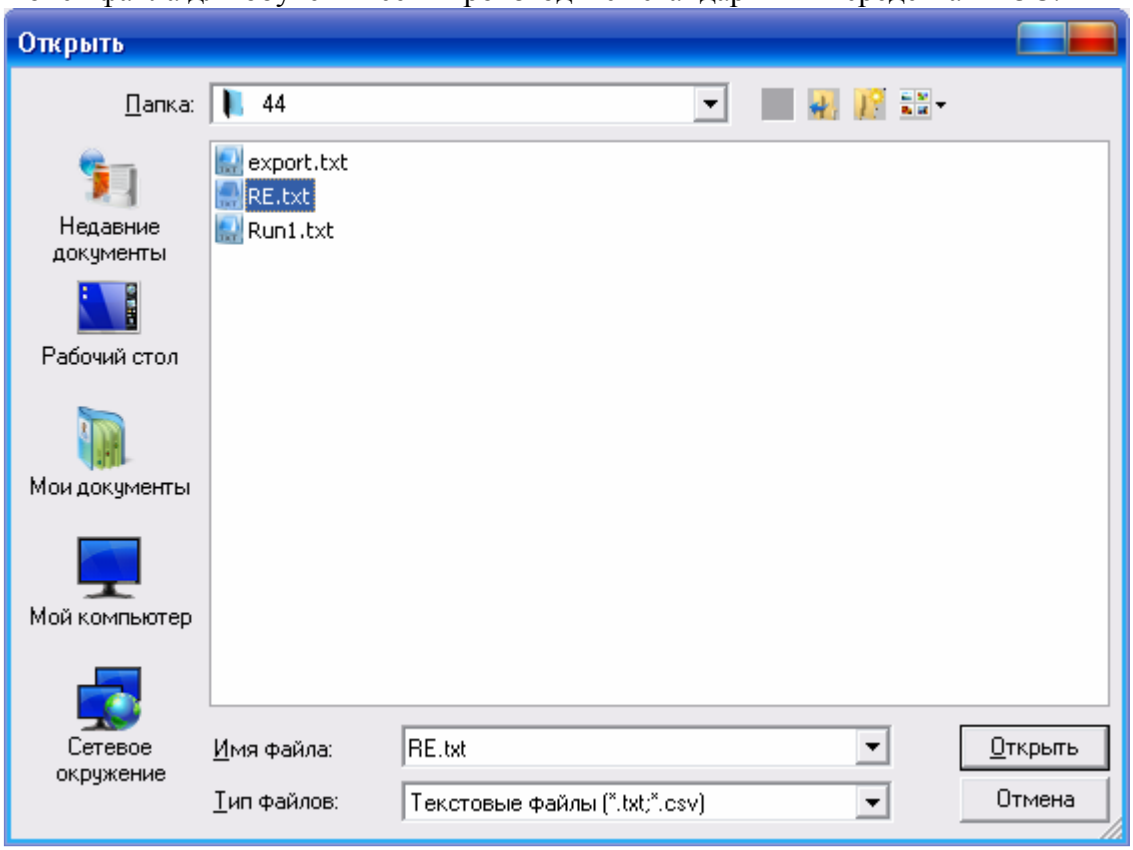
С помощью мастера импорта можно только загрузить в Deductor текстовый файл.



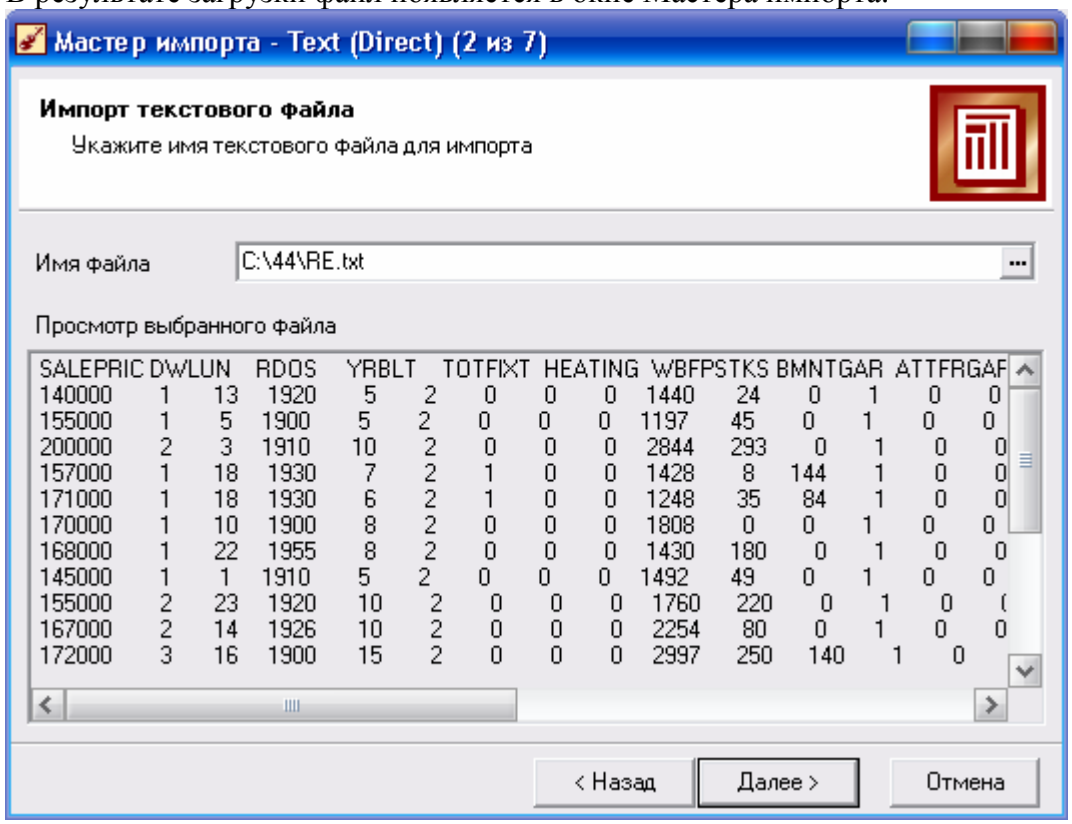
Следующее окно Мастера импорта позволяет начать поиск файла, содержащего примеры для обучения и тестирования нейронной сети:



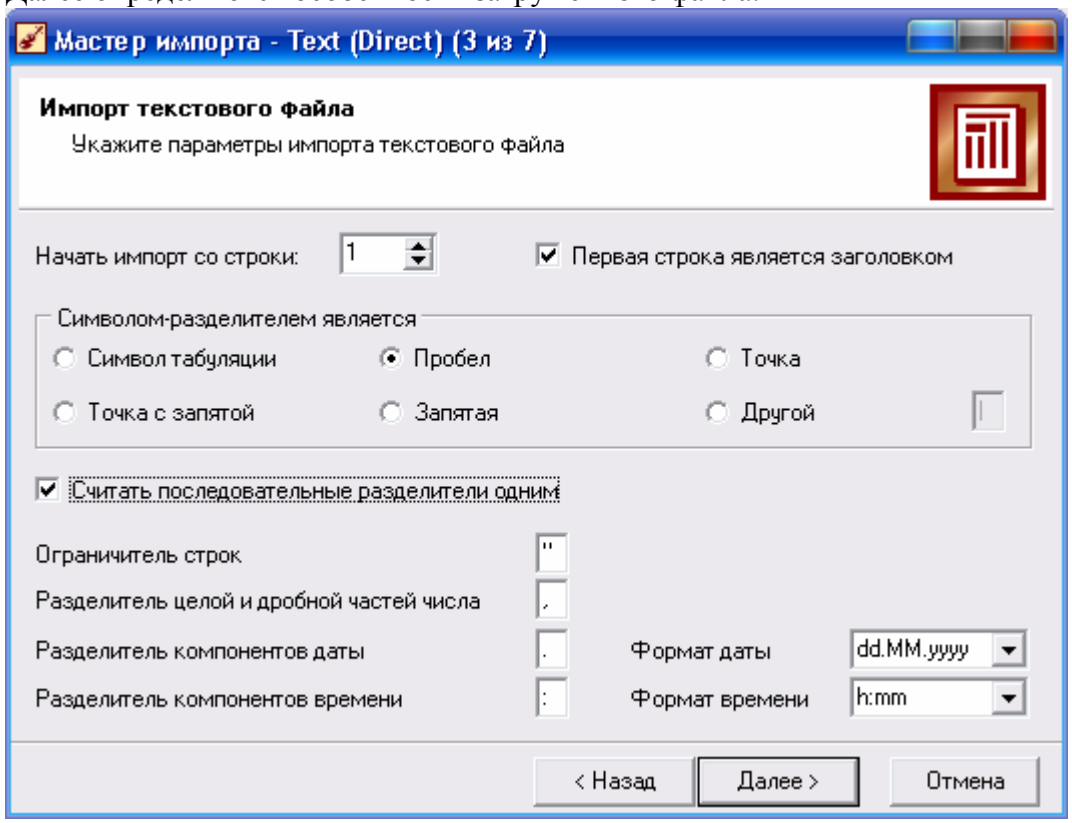
Поиск файла для обучения сети производится стандартными средствами ОС:



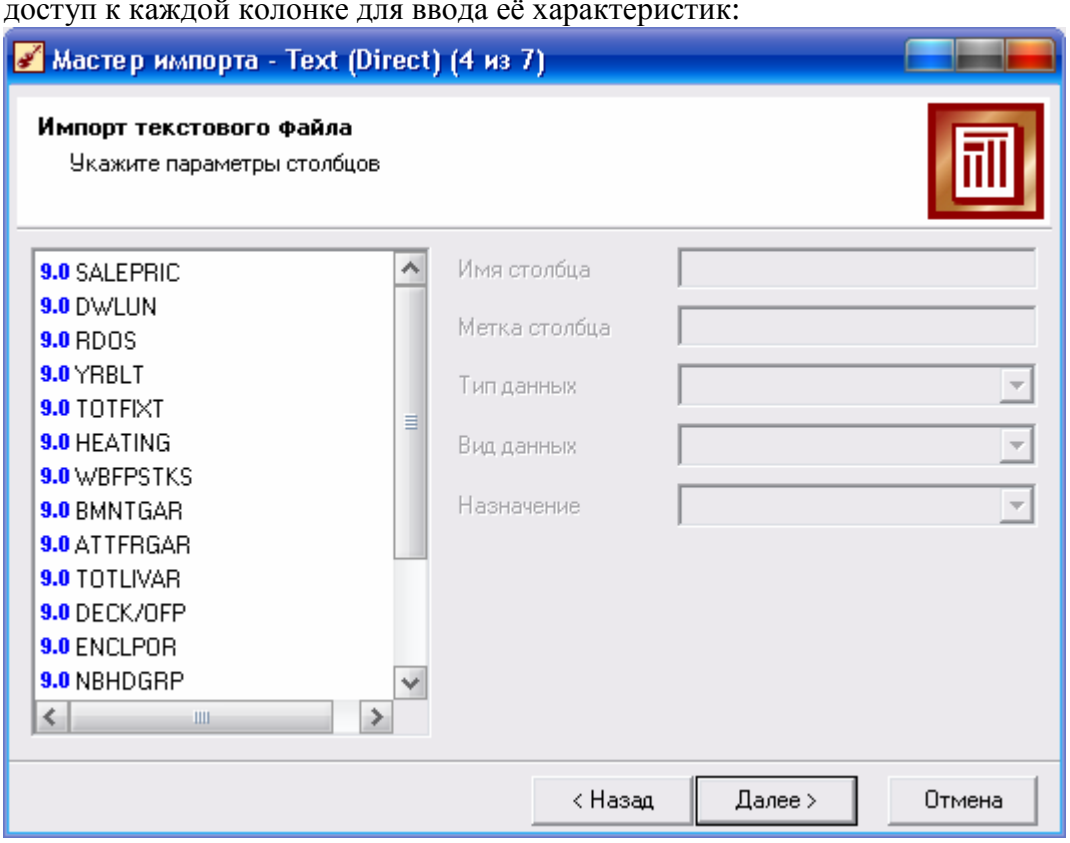
В результате загрузки файл появляется в окне Мастера импорта:



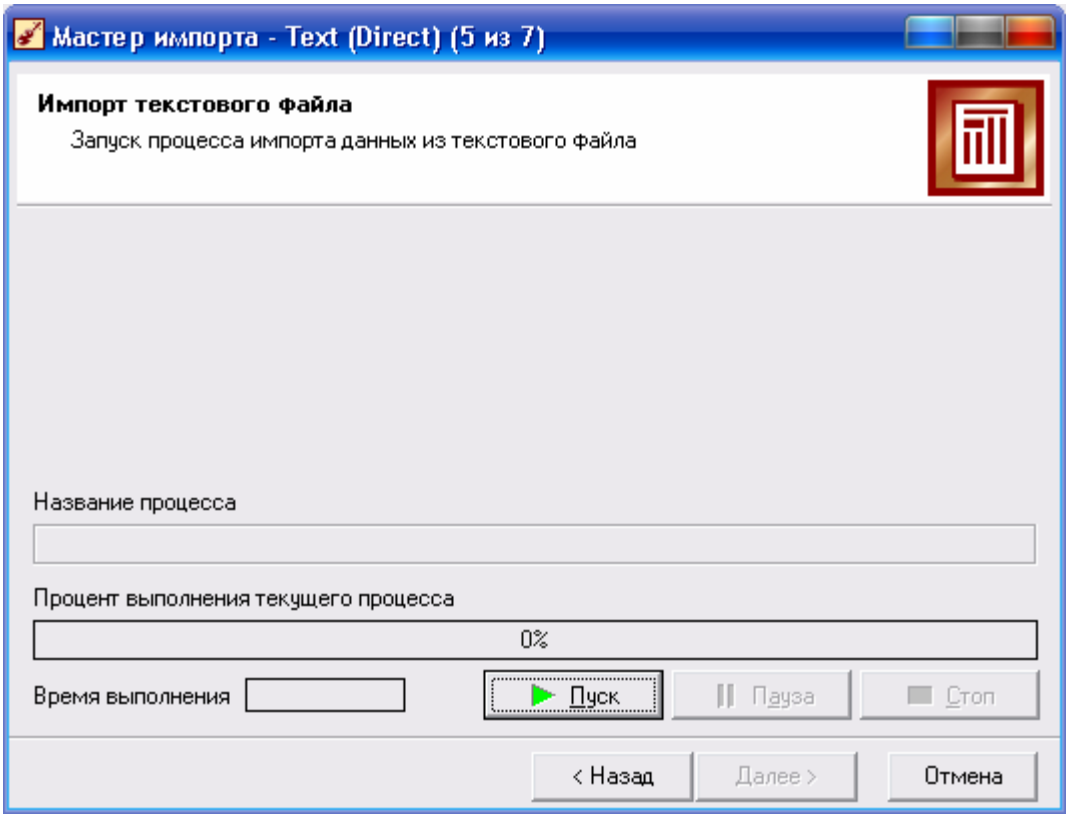
Далее определяются особенности загруженного файла:



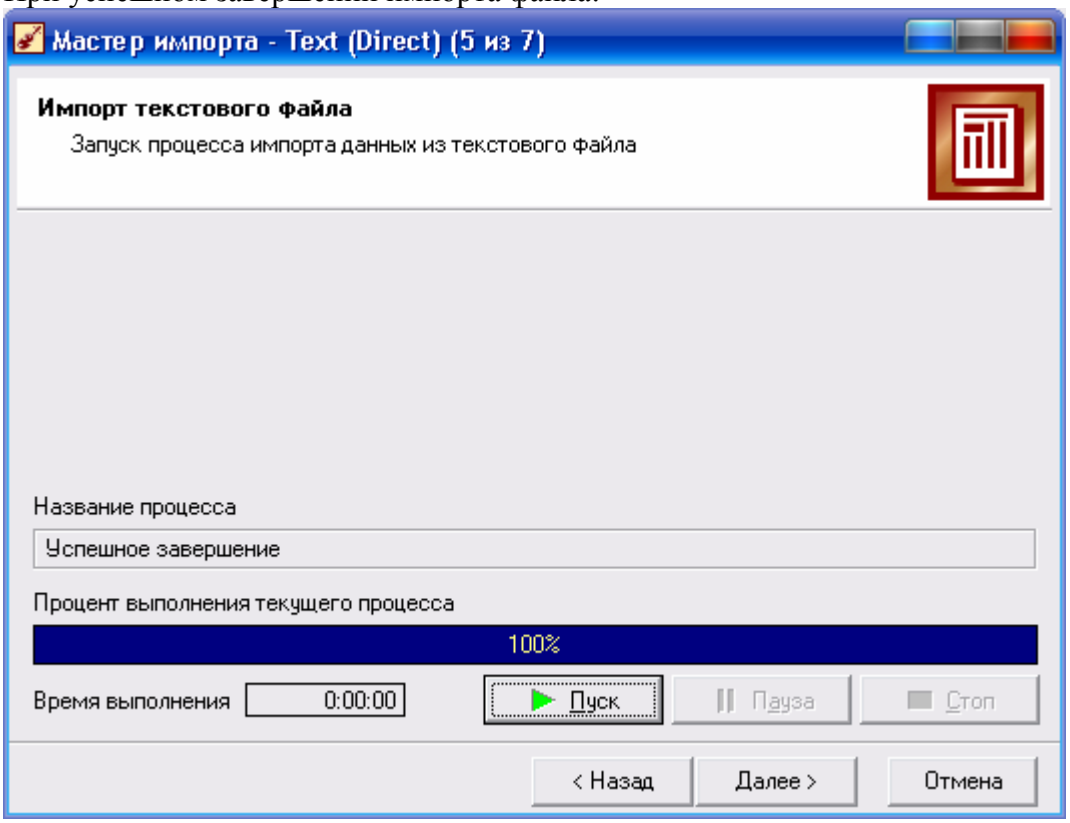
При правильной характеристике файла Мастер импорта выводит состав колонок загружаемого файла. При этом имена колонок выводятся таким образом, чтобы облегчить доступ к каждой колонке для ввода её характеристик:



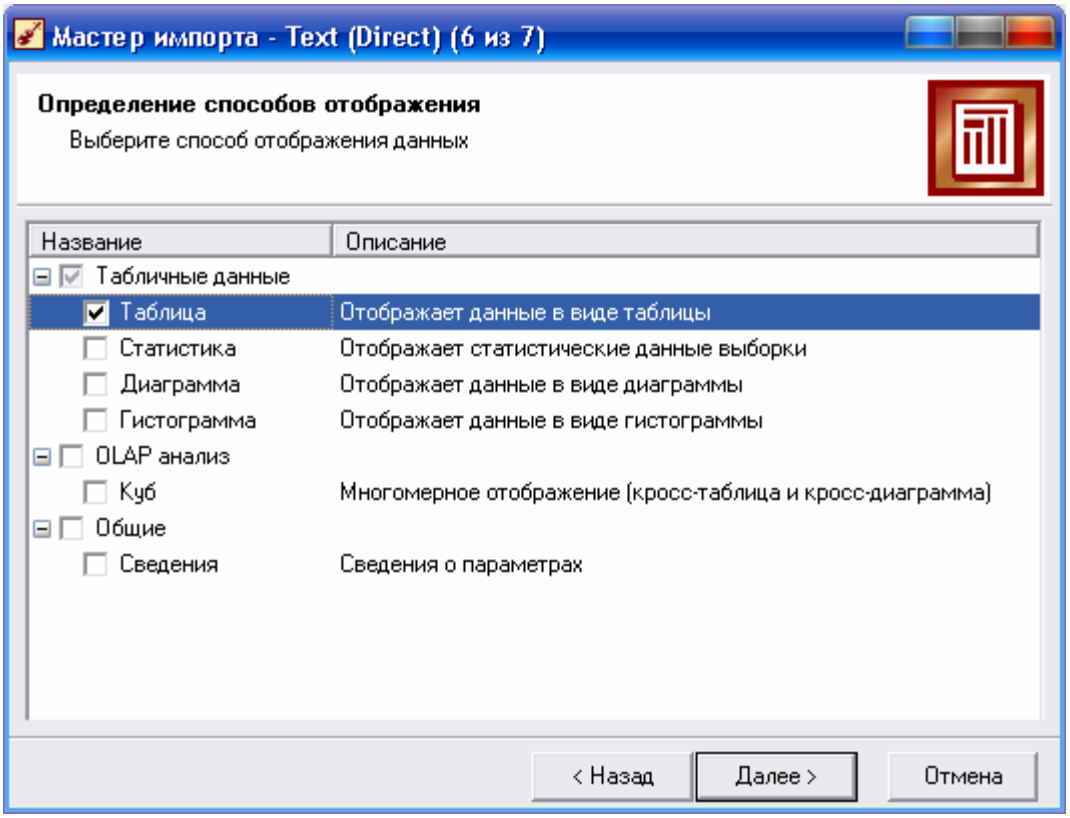
Для ввода характеристик какой-нибудь колонки надо активизировать её имя, щёлкнув по нему.
На данном этапе характеризовать колонки не надо. Можно ограничиться импортом файла в пакет Deductor:



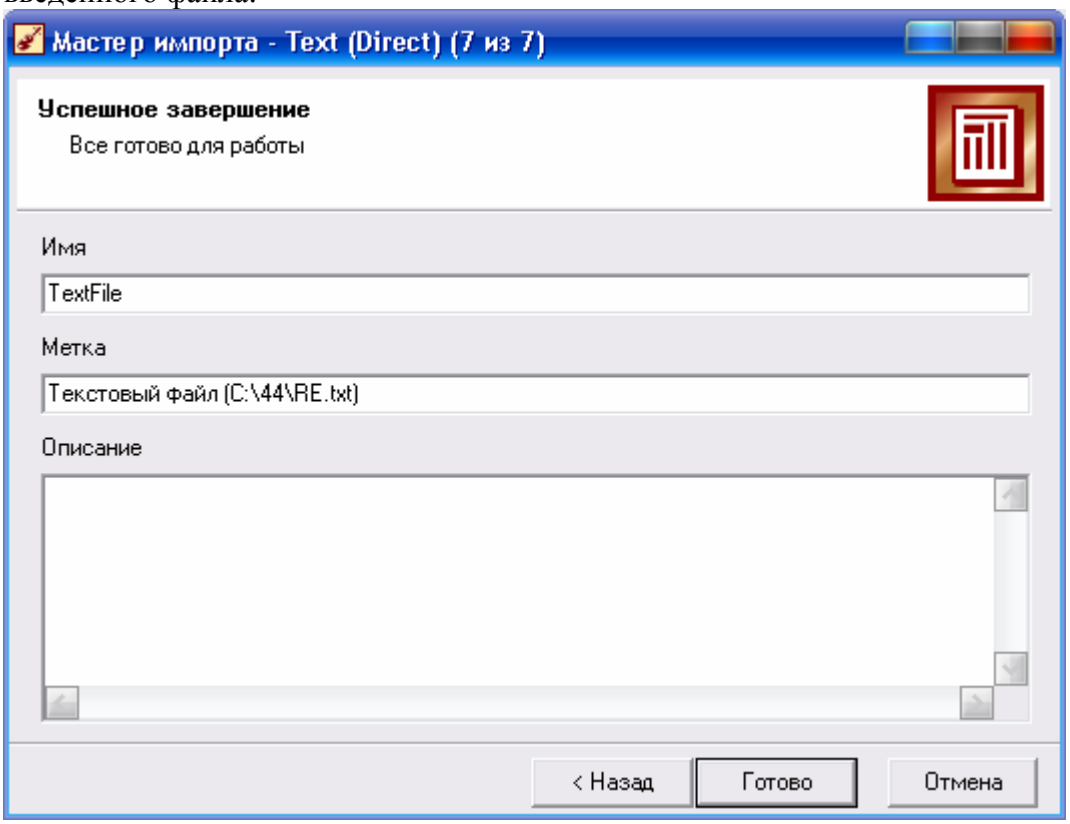
При успешном завершении импорта файла:



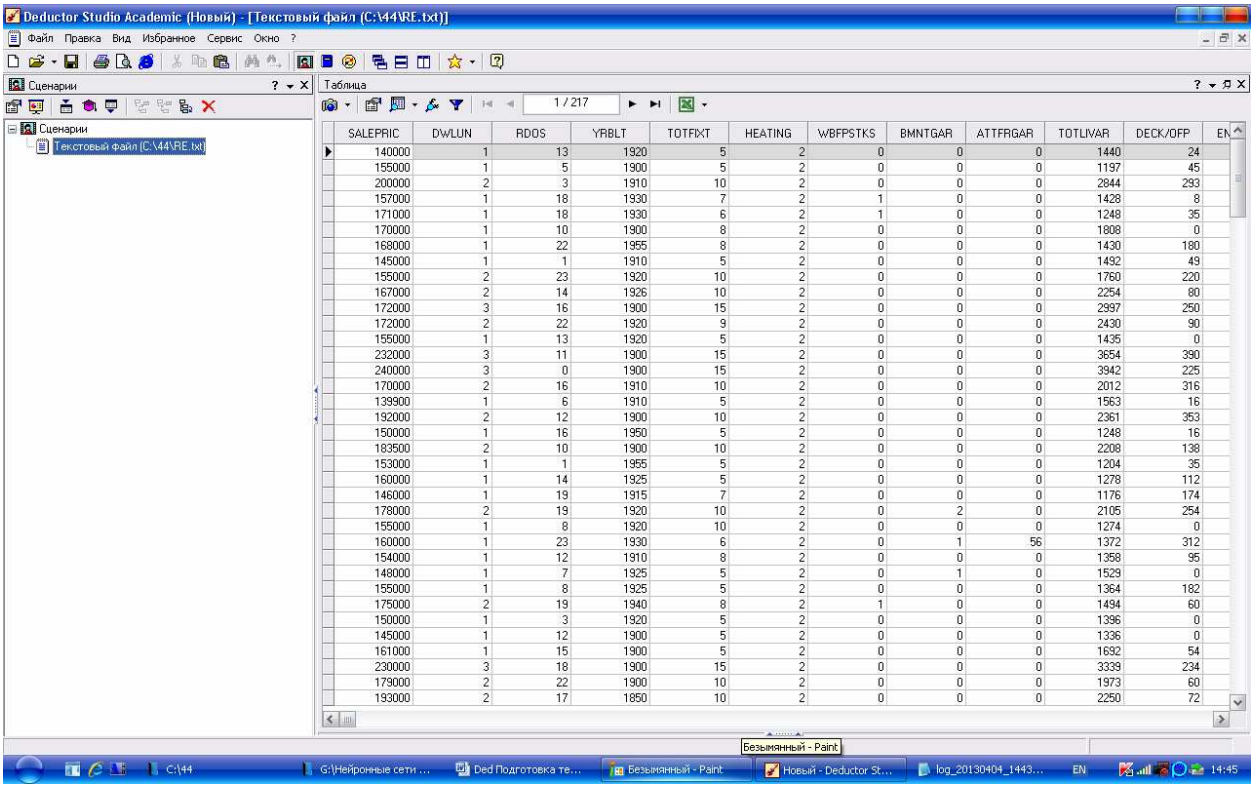
После завершения импорта определяется способ отображения введённых данных:



При успешном завершении загрузки данных система сообщает имя и местоположение введённого файла:



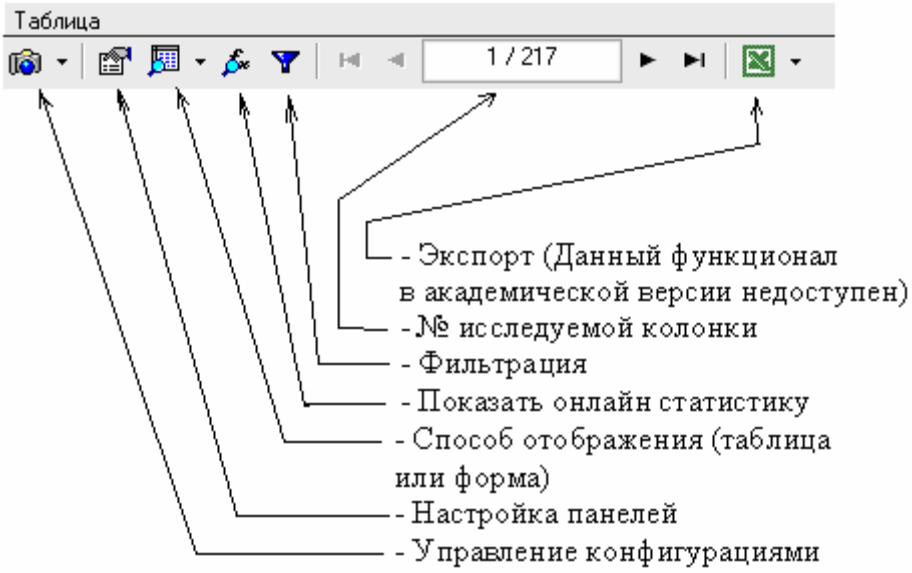
На экран выводятся заявленные в п. 6 из 7 введённые данные:



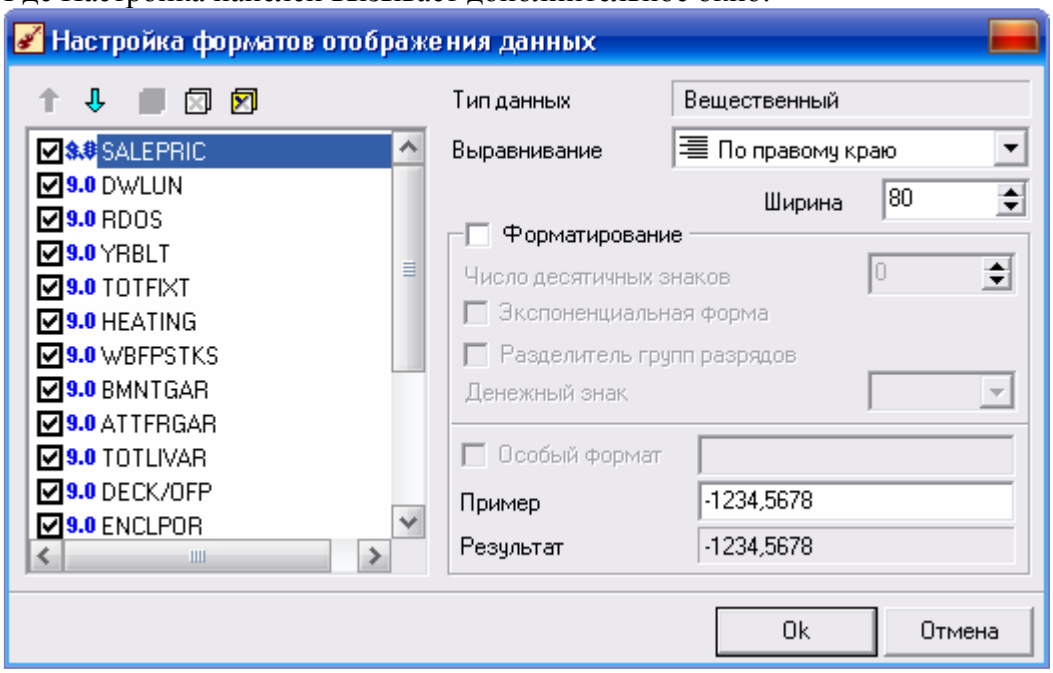
Открываются следующие элементы управления, которые ранее были недоступными:

С их помощью можно продолжить работу с нейронной сетью.

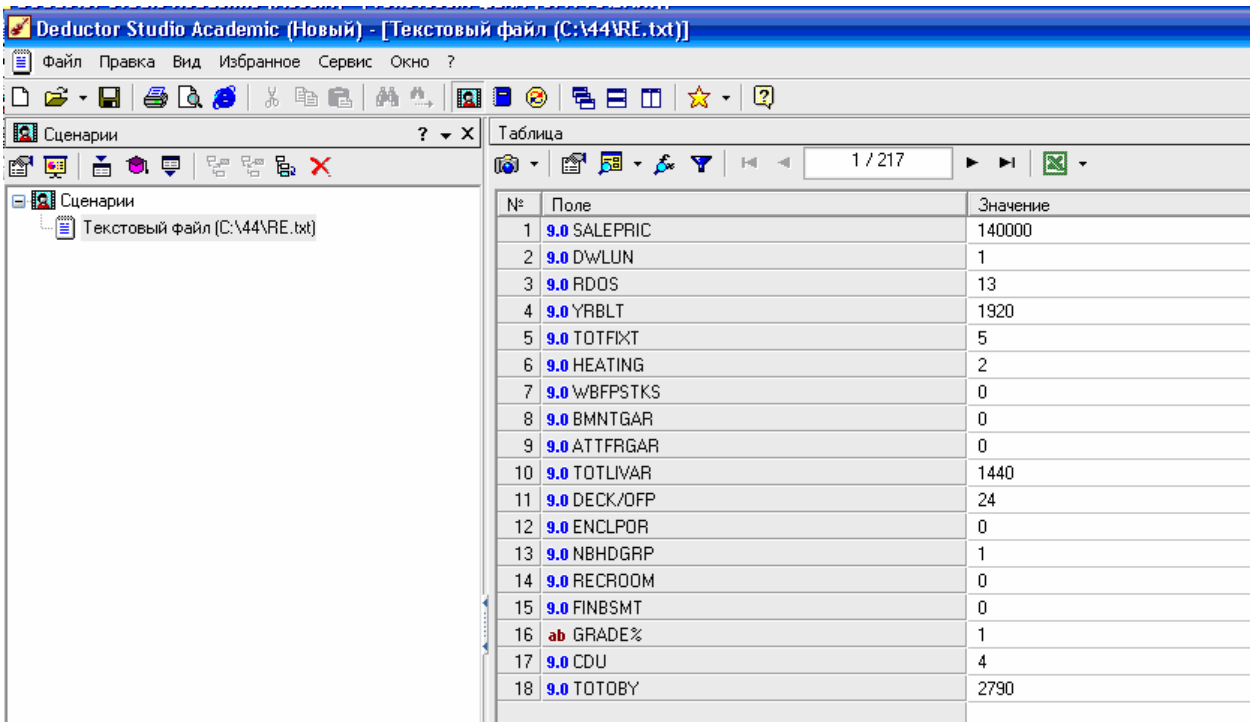
Кроме того, сформированная таблица может быть изучена и преобразована с помощью следующего меню:



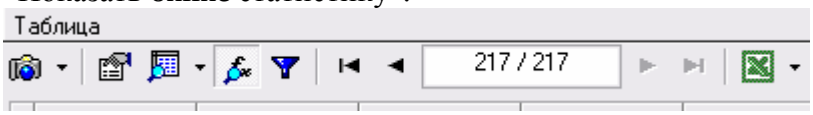
Где Настройка панелей вызывает дополнительное окно:



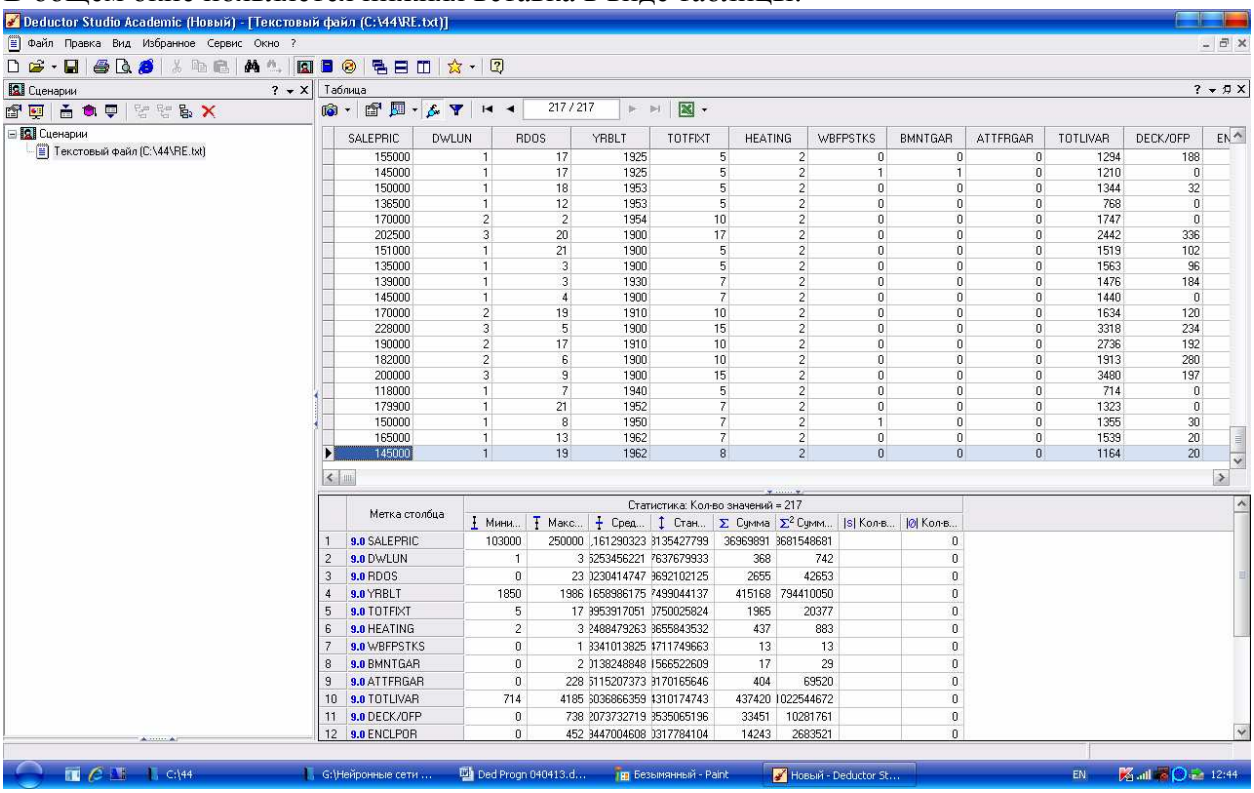
“Форма” вместо полной таблицы выводит на экран одну её строку



“Показать online статистику”:



В общем окне появляется нижняя вставка в виде таблицы:

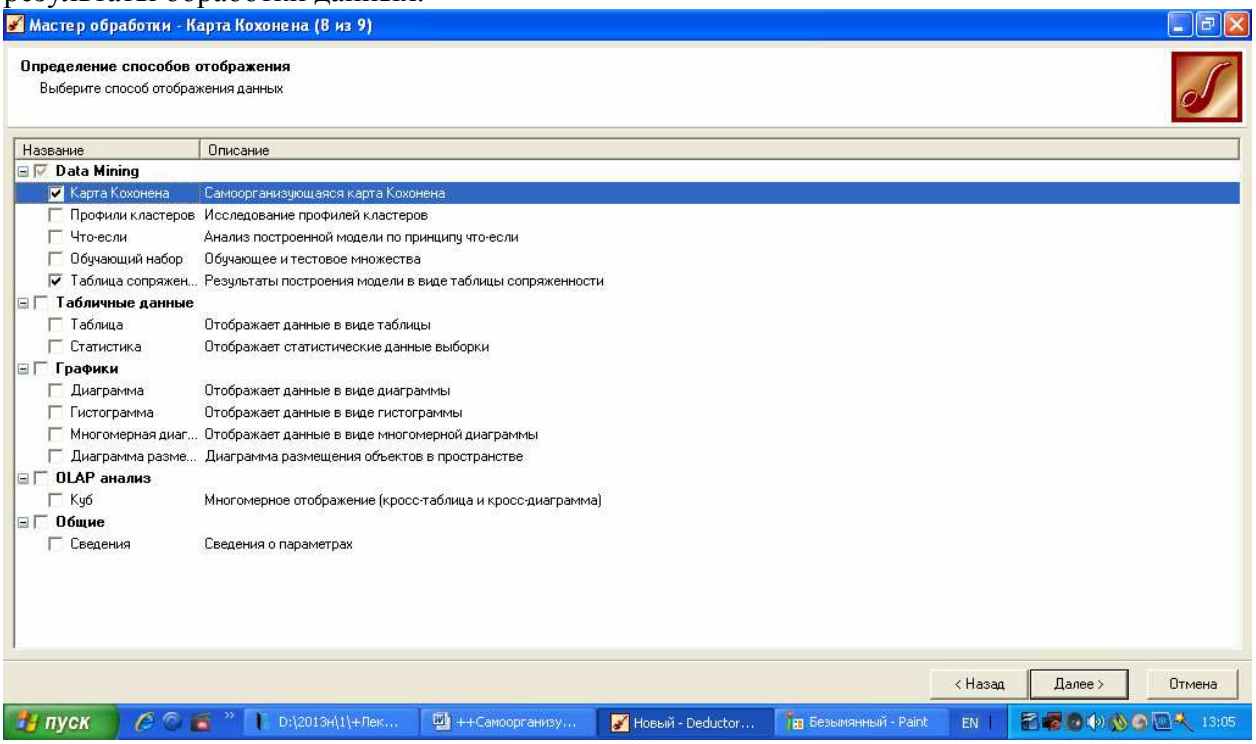


Онлайн-статистика из нижней вставки выводится в виде следующей таблицы:

	Метка столбца	Статистика: Кол-во значений = 217							
		Мини...	Макс...	Сред...	Стан...	Сумма	Σ^2 Сумм...	|s| Кол-в...	|∅| Кол-в...
1	9.0 SALEPRIC	103000	250000	,161290323	3135427799	36969891	3681548681		0
2	9.0 DWLUN	1	3	5253456221	7637679933	368	742		0
3	9.0 RDOS	0	23	0230414747	9692102125	2655	42653		0
4	9.0 YRBLT	1850	1986	1658986175	7499044137	415168	794410050		0
5	9.0 TOTFIXT	5	17	8953917051	0750025824	1965	20377		0
6	9.0 HEATING	2	3	2488479263	3655843532	437	883		0
7	9.0 WBFPSTKS	0	1	3341013825	4711749663	13	13		0
8	9.0 BMNTGAR	0	2	0138248848	1566522609	17	29		0
9	9.0 ATTFRGAR	0	228	5115207373	9170165646	404	69520		0
10	9.0 TOTLIVAR	714	4185	5036866359	4310174743	437420	1022544672		0
11	9.0 DECK/OFP	0	738	2073732719	3535065196	33451	10281761		0
12	9.0 ENCLPOR	0	452	3447004608	0317784104	14243	2683521		0
13	9.0 NBHDGRP	1	2	0783410138	4711749662	230	256		0
14	9.0 RECROOM	0	672	5451612903	5389480845	6202	2400522		0
15	9.0 FINBSMT	0	810	4470046083	9061728605	4852	2815528		0
16	ab GRADE%	1	4	2,327	1,493	505	1657	4	0
17	9.0 CDU	3	5	0783410138	0588774742	881	3613		0
18	9.0 TOTOBY	0	16400	8548387097	7583873783	247800	1383750200		0

Выбор способа отображения

На данном шаге пользователь должен выбрать, в каком виде будут отображены результаты обработки данных.



Для этого достаточно пометить нужные виды отображения флажками и щелкнуть по кнопке "Далее".

Приложение 2. Пермский симулятор.

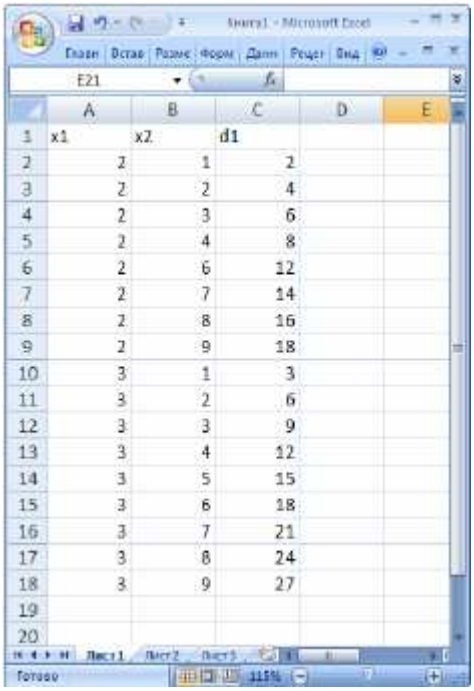
Программа «Нейросимулятор» позволяет создавать и применять нейронные сети персептронного типа.

Программа «Нейросимулятор» может работать в четырех режимах: «Проектирование сети», «Обучение», «Проверка» и «Прогноз». Переход из одного режима в другой осуществляется путем нажатия соответствующих закладок, расположенных в верхней части рабочего окна.

Обмен данными с приложением Excel.

В программе «Нейросимулятор» предусмотрена возможность обмена данными с приложением Microsoft Excel. Для этого в режимах «Обучение», «Проверка» и «Прогноз» имеются кнопки «Загрузить из Excel-файла» и «Отправить в Excel-файл». Благодаря этому, имеется возможность использовать редактор Microsoft Excel, в частности – готовить в нем обучающие и тестирующие примеры, а также обрабатывать результаты нейропрогнозов, используя графические возможности Microsoft Excel.

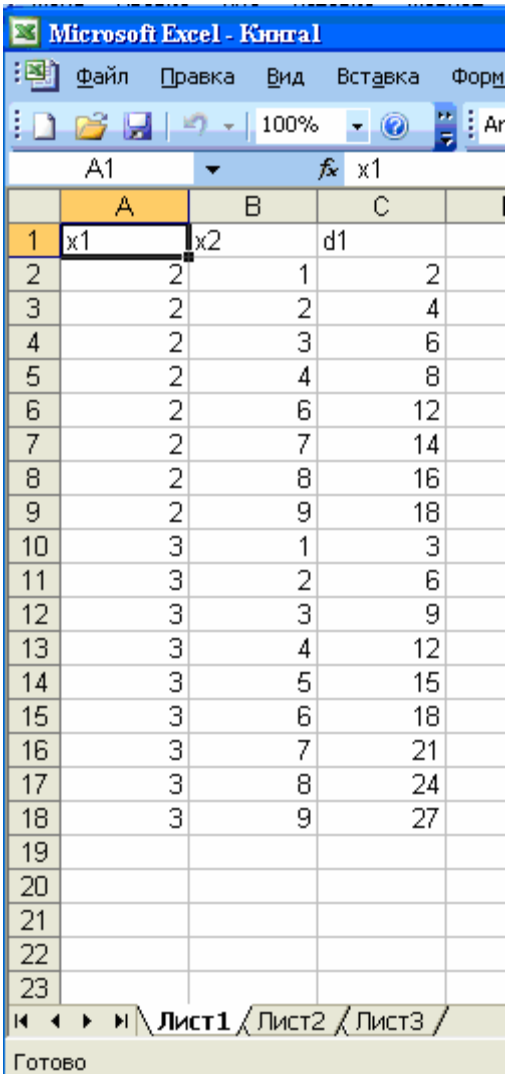
Например, множество обучающих примеров таблицы умножения, можно предварительно подготовить в редакторе Microsoft Excel, а затем загрузить его в программу «Нейрсимулятор» в режиме «Обучение» нажав клавишу «Загрузить из Excel-файла».



	A	B	C	D	E
1	x1	x2	d1		
2	2	1	2		
3	2	2	4		
4	2	3	6		
5	2	4	8		
6	2	6	12		
7	2	7	14		
8	2	8	16		
9	2	9	18		
10	3	1	3		
11	3	2	6		
12	3	3	9		
13	3	4	12		
14	3	5	15		
15	3	6	18		
16	3	7	21		
17	3	8	24		
18	3	9	27		
19					
20					

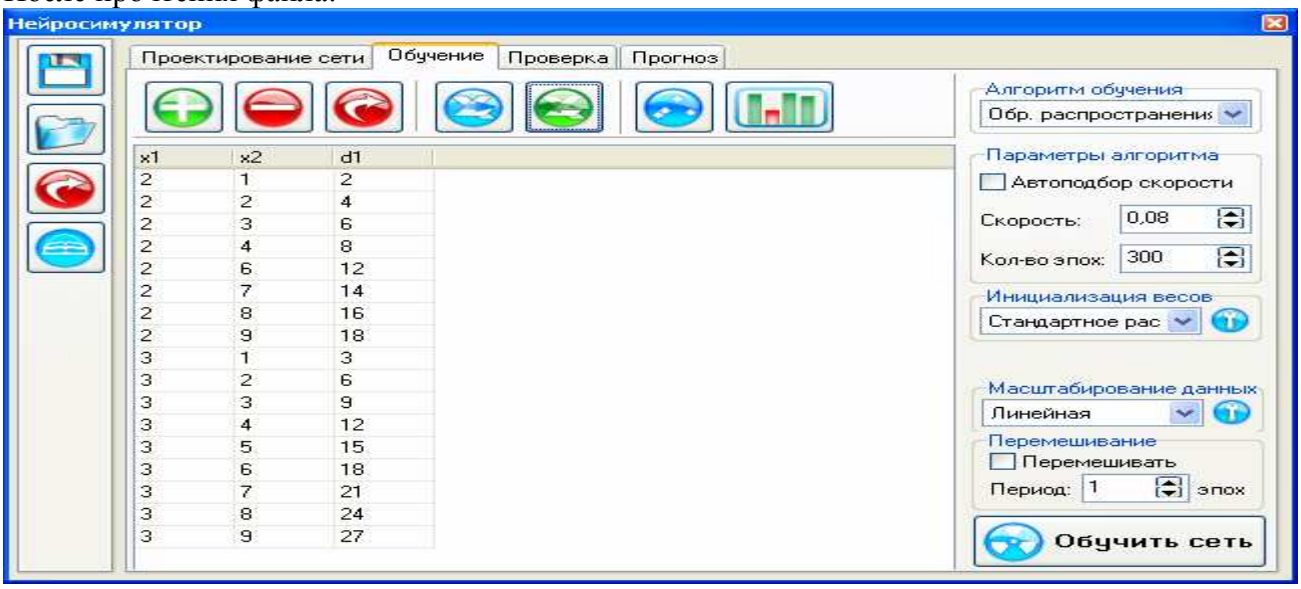
Рис. 10. Множество обучающих примеров таблицы умножения на 2 и на 3, подготовленное в редакторе Microsoft Excel

Клавиши  предназначены для обмена информацией с Excel – ввести в нейросимулятор подготовленную информацию:



	A	B	C
1	x1	x2	d1
2	2	1	2
3	2	2	4
4	2	3	6
5	2	4	8
6	2	6	12
7	2	7	14
8	2	8	16
9	2	9	18
10	3	1	3
11	3	2	6
12	3	3	9
13	3	4	12
14	3	5	15
15	3	6	18
16	3	7	21
17	3	8	24
18	3	9	27
19			
20			
21			
22			
23			

После прочтения файла:



При выводе информации из Excel запись происходит в файл «Книга1.xls».

Приложение 3. Обмен данными с MemBrain.

Managing I/O Data

ДАННЫЕ Input/Output в MemBrain организованы в так назвавшее "Разделы (Lessons)".

Раздел (**Lesson**) является коллекцией комплектов (так называемых Образцов (**Patterns)**) данных входа/выхода для специфической сети

Pattern представляет собой один набор данных входа-выхода, для сети, содержащей совместно - одну величину для каждого входного нейрона сети и одну величину для каждого выходного нейрона.

В MemBrain возможность администрирования разделов реализуется Редактором разделов (Lesson Editor).

С этого инструмента можно создать шаблоны Разделов из существующей сети или переименовать сетевые входные/выходные нейроны.

Можно так же собирать и редактировать собранные образцы текущей нейросети с помощью одного щелчка мышкой. Несомненно разделы могут также быть сохранены или загружены из файлов, использовавшихся Редактором Раздела.

Одной из наиболее важных возможностей Lesson Editor является способность перекодирования данных в разделе.

Т.е. если вы хотите проверить результаты обучения нейросети в MemBrain, можно использовать Lesson Editor для преобразования каждого шага в каждом образце раздела (опция 'Think On Lesson') и записать результат из нейросети в другой раздел. Позднее при экспорте сохранённого раздела в csv-файл можно будет использовать стандартные программы обработчика электронных таблиц.

Данные входных нейронов всегда хранятся в разделе, пока данные выходных нейронов не будут деактивированы. Это важно, если вы хотите обучить нейросеть без учителя.

В этом случае вам не нужны данные для выходных нейронов, так как эти данные неизвестны во время обучения нейросети.

Если вы хотите создать, вывести или модифицировать свой образец с помощью внешнего программного обеспечения, вы будете иметь возможность экспорта или импорта раздела с помощью csv-файла.

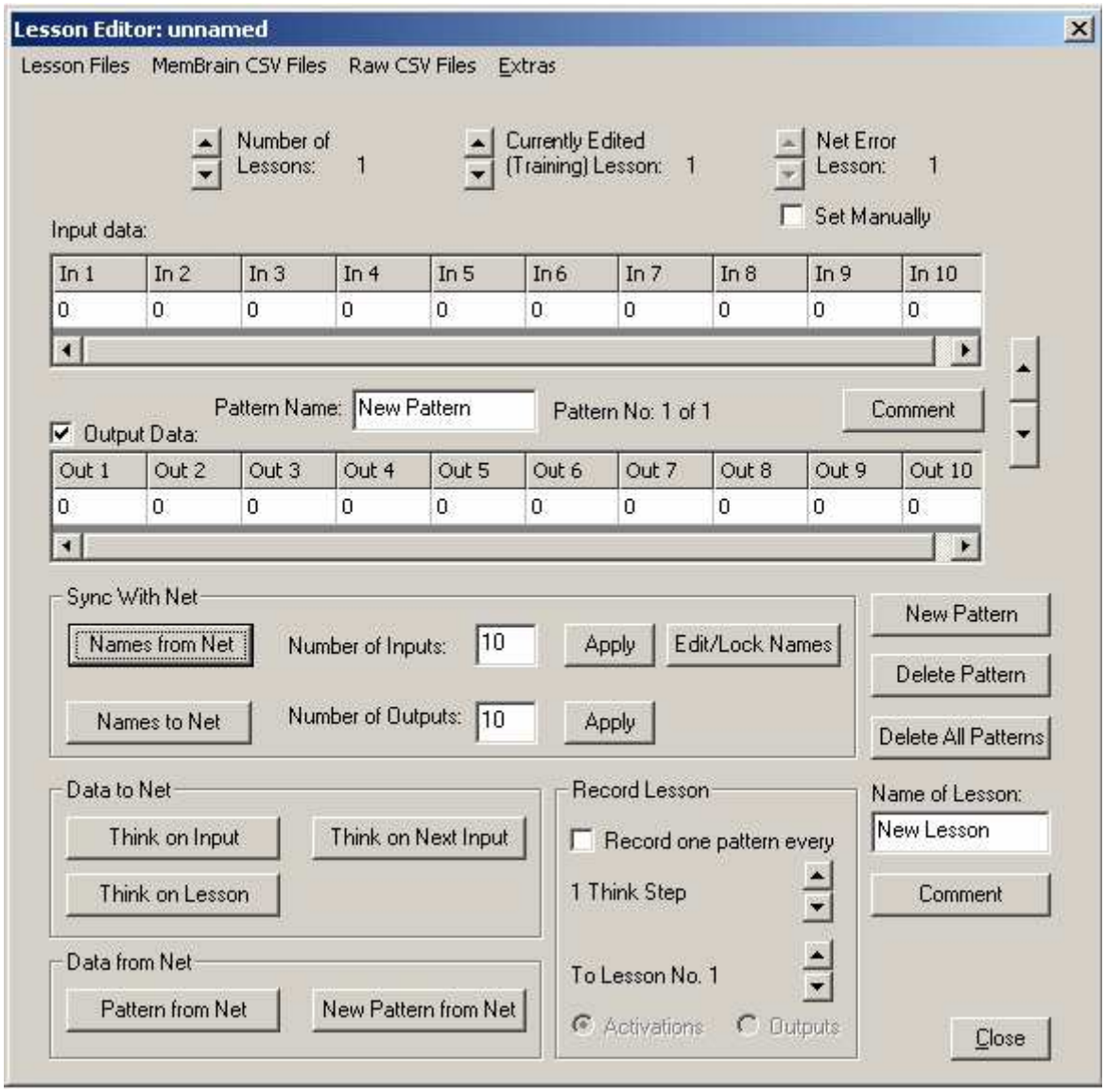
Для получения дополнительной информации о возможностях файловых форматов кликните переход на раздел Sectioned Lesson CSV File.

Есть так же возможность нормализовать входные-выходные данные в подходящие диапазоны отдельно для каждого входного или выходного нейрона.

The Lesson Editor.

Во время работы с нейронными сетями, вы будете часто иметь дело с созданием, управлением сохранением и загрузкой входных и выходных данных, с проверкой нейросети. Для этой цели в состав MemBrain добавлен Редактор разделов, который открывается функцией <Teach><Lesson Editor> из главного меню или простым щелчком на панели инструментов по кнопке:

Редактор разделов откроет своё окно:



Редактор разделов

Когда редактор разделов открывается после старта MemBrain, он всегда готов загрузить раздел, содержащий один пустой образ (все данные в нуле).

Редактор разделов всегда показывает один образ (если текущий раздел не является пустым), т.е. один набор данных, содержащий входные нейроны, и один соответствующий набор данных для выходных нейронов текущей нейросети.

Обе колонки данных отображены в одной строке таблицы каждый, каждый столбец таблицы представляет один входной и один выходной нейрон, соответственно.

Если ваша нейросеть открыта в главном окне, а входные и выходные нейроны не имеют дублирующих имён, то имена входных и выходных колонок данных остаются без изменения.

Если сначала открывается редактор разделов при отсутствии загруженной нейросети, или если имена входных-выходных нейронов не извлечены из нейросети из-за того, что входные и выходные имена дублируются (из-за чего будет получено сообщение об ошибке), то имена входных и выходных нейронов не создаются в редакторе разделов.

Тем не менее, можно синхронизировать имена столбцов данных с сетью позже, используя командные кнопки на Редакторе Раздела.

Количество разделов, администрируемых Редактором Раздела в настоящее время, отмечено на верху, слева от диалога ("Number of Lessons: ###"). Вы можете добавить

новые пустые разделы, увеличивая это число up/вниз под стрелками. Если число уменьшается, то разделы удаляются.

При попытке удалить не пустой раздел (имеет по крайней мере один образец) Вы потребуетесь Редактору Раздела прежде, чем удаление произойдет. Вы не можете удалять раздела с номером "1".

Редактор раздела всегда открывает по крайней мере один раздел. Номер текущего редактируемого раздела показан в верхнем углу справа от Редактора Раздела ("Текущий тредактируемый Раздел (Обучение): ###").

Модифицируйте это число стрелками под ним. Отображаемые данные скорректируются согласно разделу, который имеет фокус.

Все операции доступные через Редактор Разделов выполнены в этом разделе с одним исключением – нет записи данных во время работы, которая продолжается в этой главе. Это очень важно отметить, что раздел 'находящийся в редактировании' всегда - раздел, который используется для обучения нейросети!

Редактор Раздела является не модальным диалогом, он может быть использован параллельно главному действию. Нельзя закрыть Lesson Editor, когда он работает с нейросетью. Закрытие Редактора Раздела даже не влияет на свои введенные данные. Это проявится даже когда переоткрывается Редактор Раздела.

Просто не забудьте сохранять ваши Разделы перед тем, как закрывать основное окно MemBrain (то есть при выходе из MemBrain), так как при закрытии MemBrain нет предупреждения, что Редактор Раздела связавший данные сохранен! Данные Unsaved текущего раздела будут отвергнуты без уведомления!

С Редактором Раздела можно выполнить следующие функции:

Добавить-Удалить раздел:

Используйте вверх/вниз стрелки в верхнем левом углу диалога, чтобы добавлять/удалять целые разделы. Общее количество управляющих разделов отображено под стрелками.

Выберите активный раздел (Обучение):

С вверх/вниз стрелками в верхней средней области Редактора Раздела Вы можете выбрать который из управляющих разделов - к настоящему времени активный. Все действия (за исключением записи образцов), эффективны только в активном разделе.

Выбрать раздел Ошибки нейросети:

В верхнем правом углу редактора раздела выбрать, который из загруженных разделов должен быть основой для учителя, чтобы вычислять ошибку нейросети в результате обучения. Нормально, этот раздел должен быть такой же, как и активный раздел, то есть при обучении Вы увидите ошибку полученную в текущем разделе. Однако, можно вручную установить раздел сетевой ошибки, чтобы выбать другой контрольный ящик ниже соответствующих стрелок и затем изменять величину, щелкая стрелки вверх/вниз.

Таким образом, возможно использовать один раздел для подготовки (активный раздел) и вычислять сетевую ошибку на основе другого раздела, который обычно содержит неподготовленные данные подтверждения. Это позволяет проверять сеть 'НА ЛЕТУ' в течение обучения.

Выбрать активный образец:

Текущий активный образец раздела может быть изменен кнопками стрелок на правой стороне диалога. Можно также использовать колесо мышки, чтобы перемещать активный образец вверх или вниз.

Редактировать Имя образца:

Вы можете назначить уникальное имя в каждый образец раздела. Просто наберите имя в theedit область между вводной и выходной колонками данных.

Добавить Комментарий Образца:

Вы можете добавить комментарий к любому образцу раздела. Пока комментарии не начнут использоваться в MemBrain но будут включены в файл Экспорта Раздела, где они возможно окажутся полезными для вашей будущей ссылки на данные.

Редактировать имя раздела:

Вы можете назначить уникальное имя разделу. Просто наберите имя в области редактирования в правом нижнем углу Редактора Раздела

Добавить Комментарий Раздела:

Вы можете добавить комментарий к разделу. Пока комментарии не начнут использоваться в MemBrain но будут включены в файл Экспорта Раздела, где они могут быть использованы для вашей будущей ссылки на данные.

Простое Редактирование Данных:

Вы можете редактировать текущий активный вводной/выходной образец просто щелкая в области редактирования входной или выходной области таблицы данных и изменяя значения их величин.

Удалить образец:

Для начального удаления текущего образца загруженного раздела щёлкните по кнопке <Delete Pattern>.

Замечание: Подтверждения действий не потребуется и обычно нет никакой возможности Undo/Redo в Редакторе Раздела. Поэтому будьте внимательны при использовании этой кнопки.

Удалить все образцы:

Для удаления всех образцов загруженного раздела щёлкните по кнопке <Delete All Patterns>. Вам

потребуется подтверждение этого действия. Помните, что это не может отменено!

Добавить новый Образец:

Для того, чтобы добавить новый образец к разделу кликните по кнопке <New Pattern>. Новый образец, заполненный весь нулями, будет создан и добавлен в хвосте текущего раздела. Новый образец получит текущее отображенное (то есть активное) значение, независимо от того, какой образец отображался прежде.

Синхронизировать Раздел с Сетью:

Для того, чтобы использовать раздел вместе с нейросетью, раздел и сеть должны быть синхронизированы. Это означает, что входные и выходные имена и значения должны быть одинаковыми и в разделе, и в сети. Для активации этого есть несколько возможностей:

1. ***Вручную определите значения В/В и имена:***

Вы можете вручную изменить количество входов и выходов текущего раздела, печатая их величины в соответствующие области редактирования Редактора Раздела. Нажмите <Apply> кнопки под областями редактирования, чтобы позволить вашим изменениям выполниться.

Помните: Изменение этих чисел повлияет на все образцы раздела! Вы будете предупреждены при уменьшении одного из этих чисел, если эти результаты уменьшают количество данных (при удалении столбцов). Также Вы можете изменить входное и выходное имена раздела. Щелчок в <Edit/Lock именах> и имена станут отредактированными. После того, как Вы завершили изменения имён, щелкните <

2. Имена из Сети:

Когда Вы щелкаете на кнопку <Имена из Сети>, раздел автоматически будет отформатирован в соответствии с текущей загрузкой нейросети в основном окне MemBrain's. Вы будете запрошены о подтверждении действия, поскольку оно может закончиться уменьшением данных, если сеть имеет меньше входных или выходных нейронов, чем имеется в текущем разделе редактора. Обычно это - действие, которое выполняется при создании новых разделов в существующей сети или когда В/В имена сети не устанавливают их в разделе но регулируют число и порядок колонок.

Важное замечание: когда I/O-имена синхронизируются, они будут извлечены из сети и добавлены в колонки раздела слева направо для достижения той же геометрии размещения в разделе и сети.

Алгоритм будет следующим: берётся самый верхний входной нейрон и его имя размещается в самой левой входной колонке раздела. Затем ищите крайнего правого соседа этого входного нейрона и присвойте его имя следующему соседу справа в редактируемом разделе. Если больше нет в сети соседа справа, ищите следующего соседнего снизу нейрона и продолжите добавление его имени следующему правому соседу в разделе.

Этот алгоритм выбирается, чтобы автоматически передавать матрицу отформатировавшую входные области нейрона (иными словами – представления картинки, содержащей строки и столбцы) в используемом способе раздела.

Когда все входные нейроны будут синхронизированы, такая же процедура должна быть выполнена для выходных нейронов.

3. *Имена в Сеть:*

Это действие передаёт I/O-имена из текущего раздела в I/O-пейроны нейросети, открытой в главном окне. Функция удачно будет выполнена только если количество входных и выходных нейронов будет одинаковым для обоих – и раздела, и сети. Можно получить сообщение об ошибке, если это не соблюдается. Алгоритм используется для идентификации нейронов, которые присоединяют имена колонок из разделов, соответствующих описанным в предшествующей секции.

Подумайте в Текущем Образце входа:

Если кликнуть по кнопке <Think on Input>, то входные данные из активного текущего образца будут приложены входным нейронам открытой сети и сеть выполнит один шаг размышления (Think Step).

Можно получить сообщение об ошибке, если действие ошибочно. Можно использовать эту кнопку во время режима Auto Think.

Размышление о Следующем Входном Образце:

Если щёлкнуть кнопку <Think on Next Input>, входные данные следующего образца активного раздела будут приложены к входным нейронам открытой сети и сеть выполнит один Шаг Мышления.

Если действие ошибочное, появится сообщение об ошибке. Эту кнопку можно так же использовать в режиме Auto Think. Используйте колёсико мышки для перемещения вверх и вниз через раздел и выполните Шаг Мышления в выбранном образце при перемещении. Для этого нажмите и удерживайте кнопку <CTRL> клавиатуры во время перемещения.

Дополнительные возможности:

Подумайте в Разделе:

При нажатии на кнопку <Think on lesson> все входные образы данных текущего раздела будут приложены ко входным нейронам открытой нейросети и сеть выполнит один шаг мышления для каждого образа. Процедура начинается с первого образца раздела и заканчивается с последним образцом.

Если действие неправильное, вырабатывается сообщение об ошибке.

Возьмите Образец из Сети:

При нажатии кнопки <Pattern from Net> текущий образец будет переписан с текущей активизацией входа соответственно выходным нейронам нейросети.

Вы потребуетесь, чтобы подтверждать действие.

Добавьте Новый Образец из Сети:

При кликировании кнопки <New Pattern from Net> создаётся новый образец с текущей активацией входов соответственно выходным нейронам нейросети. Новый образец помещается в конец раздела и будет высвечен, как активный образец.

Это действие может быть использовано для очень эффективного создания нового образца с текущей активацией входа относительно выходных нейронов сети. Вы не потребуетесь, чтобы подтверждать действие, так как не найдется никакой убыток данных. Новый образец будет установлен в конце раздела и будет отображен как активный образец.

Запись раздела:

Вы можете автоматически записать новые образцы раздела проверяя ящик на правой нижней правой стороне редактора раздела. Если он активизирован, редактор раздела добавит один образец в скорректированный раздел каждый раз, когда выбранный номер мыслительного шага выполняется.

Вы можете отрегулировать записывающий интервал образец, щелкая на соответствующих вверх/вниз (up/down) стрелках. Раздел на который новые образцы должны быть добавлены, также может быть скорректирован.

С помощью этой возможности это легко приобретать данные подтверждения о вашей сети: Создайте новый раздел и регулируйте запись в этот раздел. Затем установите активный раздел на то, что обеспечивают ваши тестирующие образцы. Разрешите сети думать обо всех образцах в тестирующем разделе кнопкой <Думать в Разделе> (<Think on Lesson>). Ответ вашей сети вместе с приложенными входными данными будет записан в новый раздел после каждое шага «думать». Не имеет значения, как выполнены Шаги Think (Think Steps): в режиме Авто Думать (Auto Think), при ручном запуске режима, или во время обучения сети.

Если запись производилась в режимах Auto Think или Auto Teach, редактор разделов не будет актуализирован новыми данными, пока Вы снова не остановите процесс моделирование/обучение (simulation/teach).

С помощью двух радиокнопок <Activations> и <Outputs> можно выбрать, какие данные из выходных нейронов должны быть сохранены в разделах. Значение по умолчанию есть <Activations>. Это вызывает величины активизации выходных нейронов, которые нужно записывать. Если Вы изменяете выбор на <Outputs>, тогда выходные величины (пожарный уровень (fire level)) выходных нейронов записаны вместо величин активизации. Это полезно, если Вы используете установку '1" вместо 'Активизация" как выходной пожарный уровень выходных нейронов: Вы можете отрегулировать пожарные пороги выходных нейронов в величину, которая соответствует вашим критериям решения

- когда нейрон должен считаться как активный и когда как неактивный. Нормально нужно устанавливать оба, как нижний так и верхний пожарный порог в ту же величину напр.. 0.5. Это должно вустанавливать на выходе нейрона величину '1 когда активизация выше чем 0.5 и 0 в противном случае. Эти величины будут записаны в раздел если Вы выбираете упомянутый выбор <Outputs>. Важно, помнить, что возможно также записывать данные в текущий активный раздел. В большинстве случаев это не желательно, важно не забыть выбрать раздел, отличный от активного раздела для записи. Также, проверьте, что запись выведена однажды, как только Вы завершили действие, для того, чтобы предохраниться от невнимательной записи во время дальнейших действий!

Сохранить раздел:

Для того, чтобы сохранить текущий редактируемый раздел в формате внутреннего раздела надо нажать <Lesson Files><Save Current Lesson> или соответственно <Lesson Files><Save Current Lesson As...> в меню Lesson Editor's.

Сохранить все разделы:

Для сохранения всех разделов в одном коротком, надо нажать <Lesson Files><Save all Lessons> в меню Lesson Editor's.

Загрузить раздел:

Для загрузки раздела в Редактор раздела использовать в меню Lesson Editor's <Lesson Files><Load Current Lesson...>. Имейте в виду, что это сотрёт все ранее введённые данные текущего раздела. Если надо добавить образы (patterns) из другого раздела в текущий редактируемый раздел, нажмите <Lesson Files><Append to Current Lesson...> взамен.

Добавление раздела в текущий раздел:

Можно добавить все образы из файла разделов MemBrain в текущий редактируемый раздел, т.е. объединить два небольших радела в один, нажав <Lesson Files><Append Lesson to Current Lesson...> в меню Lesson Editor's.

Вывод раздела в виде CSV файла:

Текущий редактируемый раздел может быть экспортирован как файл данных MemBrain, отформатированный в CSV-формате для дальнейшего использования, например, в электронных таблицах. Для этого надо использовать меню редактора разделов <MemBrain CSV Files><Export Current Lesson...>.

Ввод раздела в виде CSV файла:

Содержимое текущего редактируемого раздела может быть получено из MemBrain файла данных, отформатированного в формате CSV, созданного или модифицированного в другом приложении. Для этого надо использовать функцию <MemBrain CSV Files><Import Current Lesson...> из меню редактора разделов.

Вывод раздела как необработанного (Raw) CSV файла:

Содержимое текущего редактируемого раздела может быть получено из необработанного MemBrain файла данных, отформатированного в формате Raw CSV, для дальнейшей обработки в другом приложении, например, в электронной таблице. Для этого воспользуйтесь функцией <Raw CSV Files><Export Current Lesson (Raw CSV)...> из меню редактора разделов. Формат Raw CSV описан отдельно.

Ввод раздела из необработанного (Raw) CSV файла:

Содержимое текущего редактируемого раздела может поступить из файла данных, отформатированного в формате Raw CSV, созданного или модифицированного в другом приложении. Для этого надо использовать функцию <Raw CSV Files><Import Current Lesson (Raw CSV)...> из меню редактора разделов. Формат Raw CSV описан отдельно.

Вывод входных данных раздела в виде Raw CSV файла:

Вводные данные текущего редактируемого раздела может быть выведен в виде данных, отформатированных в Raw CSV-формате для дальнейшего использования в других приложениях, как например в электронных таблицах. Для этого надо использовать функцию <Raw CSV Files><Export

Current Lesson Inputs (Raw CSV)...> из меню редактора разделов. Формат Raw CSV описан отдельно.

Ввод входных данных раздела из Raw CSV файла:

Входные данные текущего редактируемого раздела могут быть импортированы в виде Raw CSV формата данных, созданных или модифицированных в других приложениях. Для этого надо использовать функцию <Raw CSV Files><Import Current Lesson Inputs (Raw CSV)...> из меню редактора раздела. Формат Raw CSV описан отдельно.

Вывод выходных данных раздела в виде Raw CSV файла:

Выходные данные текущего редактируемого раздела могут быть экспортированы в виде Raw CSV формата данных для дальнейшего использования в других приложениях, например – в электронных таблицах. Для этого надо использовать функцию <Raw CSV Files>><Export Current Lesson Outputs (Raw CSV)...> из меню редактора раздела. Формат Raw CSV описан отдельно.

Ввод выходных данных раздела из Raw CSV файла:

Выходные данные текущего редактируемого раздела могут быть импортированы в виде Raw CSV формата данных, созданных или модифицированных в других приложениях. Для этого надо использовать функцию <Raw CSV Files><Import Current Lesson Outputs (Raw CSV)...> из меню редактора раздела. Формат Raw CSV описан отдельно.

Разделить текущий раздел:

Часто необходимо разделять раздел на две части: одна для обучения, и одна для тестирования. В MemBrain можно выполнить это разделение автоматически в редакторе разделов. Для этого надо выбрать <Extras><Split Current Lesson...> и ввести процент, согласно которому должно быть выполнено разделение. Образец разделения будет выбран произвольный и перенесен в другой раздел, который создаётся автоматически. Новый раздел всегда будет последним разделом Редактора разделов. После разделения можно заметить, что текущий раздел содержит меньше образов, чем раньше. Используйте стрелку «вверх» в правой стороне Редактора разделов для навигации по вновь созданному разделу. Образцы для вновь созданного раздела упорядочены в соответствии с их исходными позициями в исходном разделе.

Работа только со входными данными:

Если вы хотите работать только со входными данными, можно деактивировать выходные данные раздела: переместите защёлку под текстом 'Output Data:' выше колонки, показывающей данные выходных нейронов. Данные потускнеют. Редактор разделов

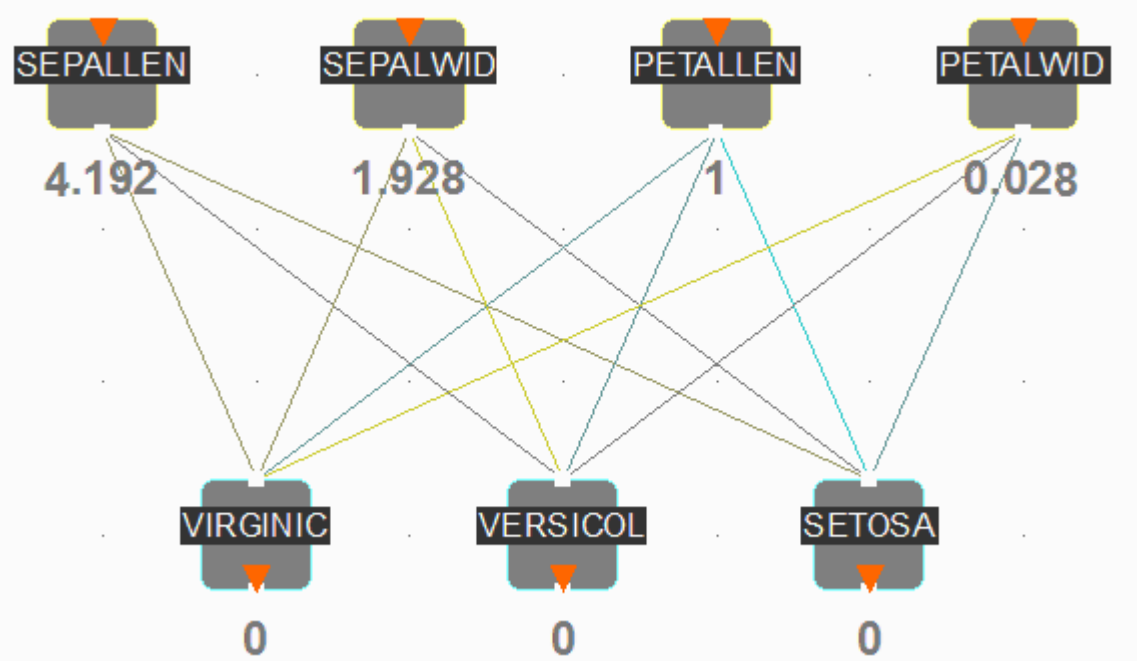
теперь будет работать только с входными данными и перестанет уважать выходные данные. Это бывает необходимо, когда вы хотите обучить сеть, используя неуправляемое обучение (без учителя). В этом случае вы не нуждаетесь в данных для выходных нейронов, как будто эти данные неизвестны, до окончания обучения нейросети.

Кластеризация при помощи нейропакета MemBrain

Постановка задачи: в пакете MemBrain выполнить кластеризацию ирисов Фишера при помощи нейронной сети.

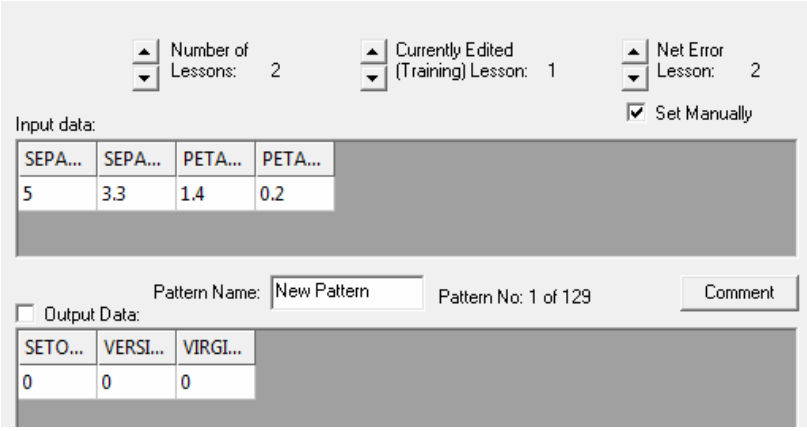
Подготовка входных данных: в Excel сохраним входные данные из обучающей выборки в виде CSV-файла. Тестовую выборку оставим с выходными данными для проверки нейронной сети; классы SETOSA, VERSICOL и VIRGINIC при этом закодируем как единичный выход на 1, 2 и 3 нейронах соответственно. См. файлы *input.csv* и *test.csv*.

Создание нейронной сети: в нейропакете MemBrain построим нейронную сеть с 4 входами и 3 выходами (по количеству классов). Для выходных нейронов при этом поставим функцию активации **MIN EUCLID DISTANCE**.



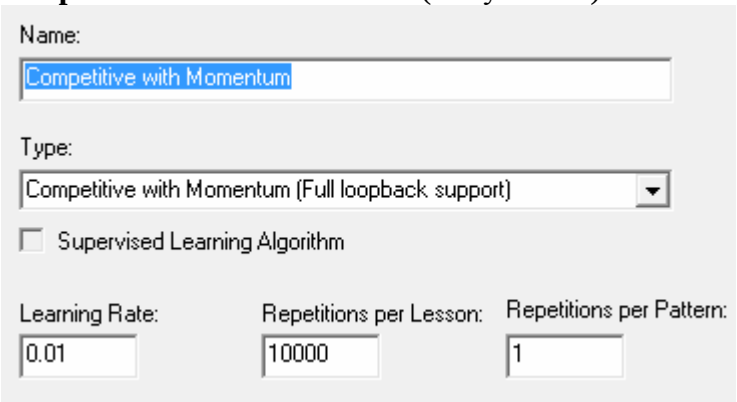
Вид построенной сети.

Обучение нейронной сети: в **“Lesson Editor…”** импортируем созданные ранее файлы как два урока, причем у обучающего отключим выходные данные (которых в файле нет).



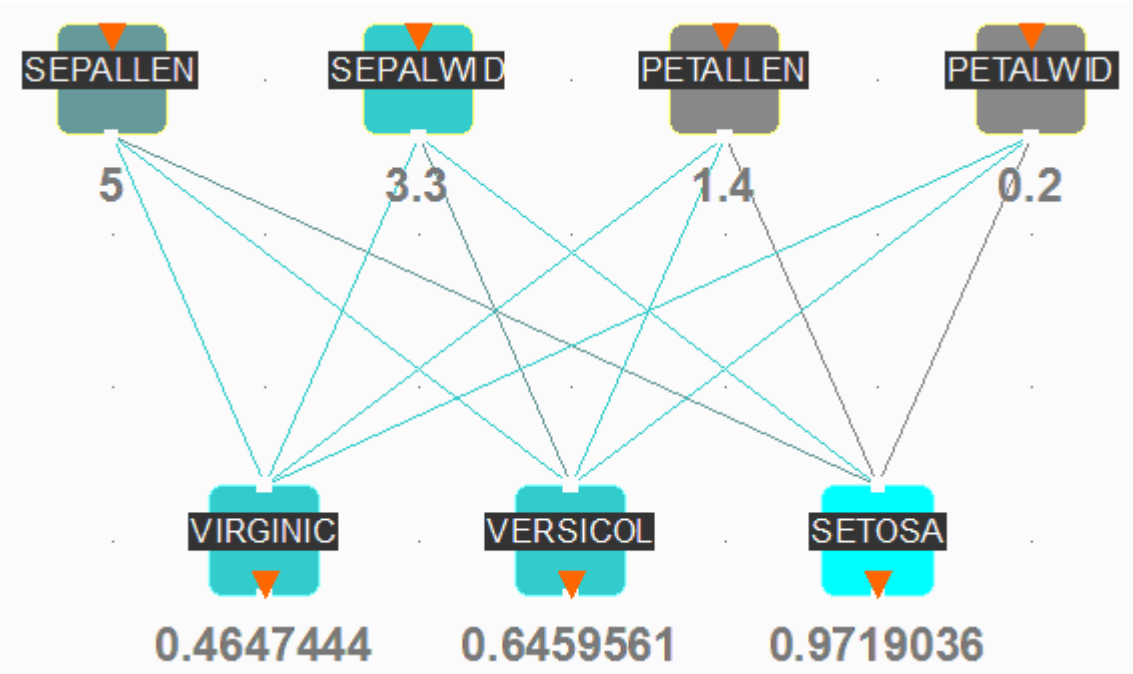
Параметры уроков: 1 – обучающий, 2 – тестовый.

При помощи **Extras > Normalization Wizard** выставим граничные значения входных нейронов по обучающей выборке, после чего в окне **“Teaching Manager…”** выберем алгоритм обучения **“Competitive with Momentum”** (без учителя).



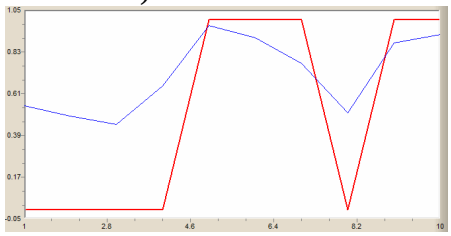
Чтобы в процессе обучения не переименовались выходные нейроны (т.к. обучающая выборка не содержит названий, они все будут названы “New Pattern”), нужно отключить опцию **“Rename Winner Neurons According to Patterns”**.

После чего нажимаем кнопку “Teach Step” чтобы обучить нейронную сеть. По итогам обучения может потребоваться переименовать выходные нейроны чтобы названия соответствовали обнаруживаемым классам.

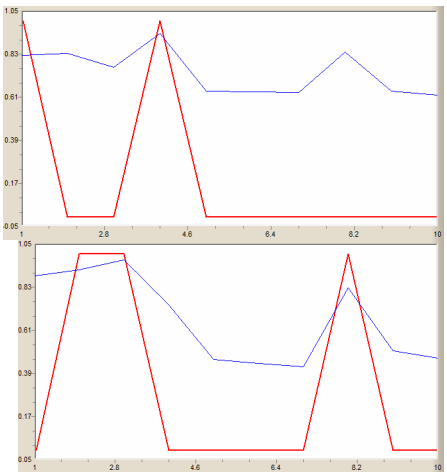


Пример распознавания нейронной сетью цветка из класса SETOSA.

Чтобы протестировать нейронную сеть, переключим учителя обратно на RPROP (т.к. иначе MemBrain не позволит открыть окно, отображающее ошибку), выставим 2 – тестовый – урок как активный и нажмем кнопку **“Think on Lesson”**. Теперь в окне **Pattern Error Viewer** будет отображаться сравнение вывода нейронной сети (синия линия) с ожидаемым (красная линия).



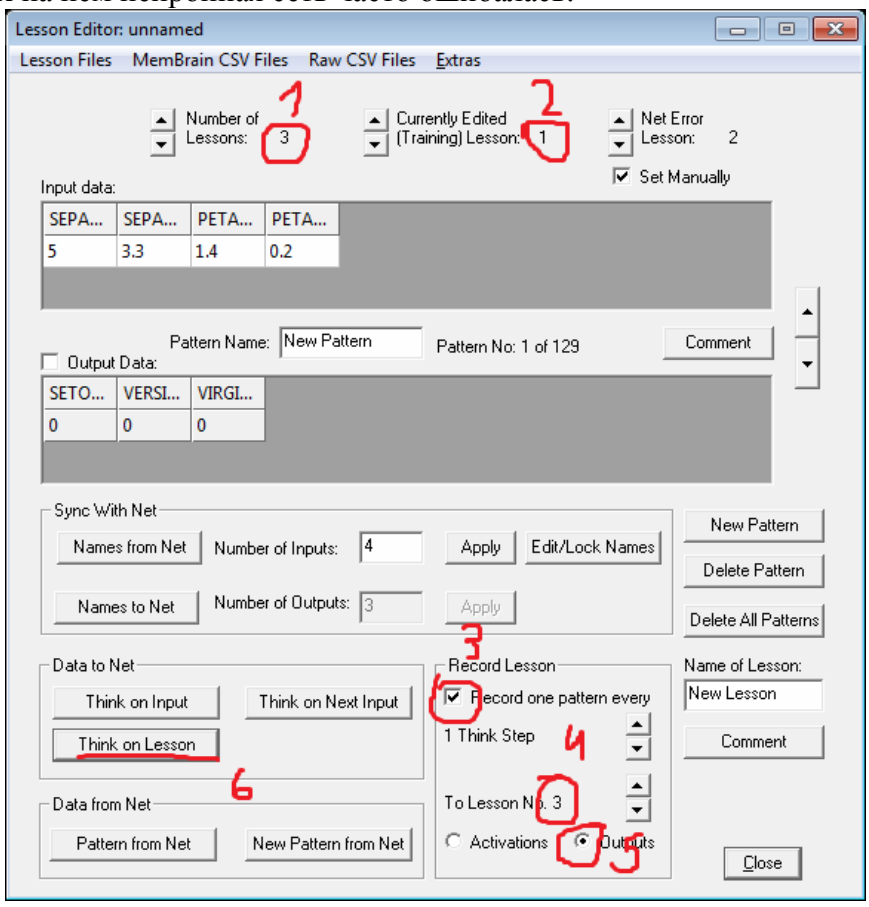
SETOSA



VERSICOL

VIRGINIC

Видно, что нейронная сеть действительно обнаружила требуемые классы, хотя и с гораздо меньшей точностью, чем в режиме классификации. Особую трудность для сети составил класс VERSICOL, что не удивительно, т.к. даже при известных во время обучения ответах на нем нейронная сеть часто ошибалась.



Последовательность шагов для экспорта результатов.

Экспортируем результаты кластеризации тестовой выборки в Excel. Для этого сначала запишем результаты работы нейронной сети на обучающей выборке в новый урок: увеличим число **“Number of Lessons”** до 3, после чего включим режим **“Record Lesson”** с параметром **“To Lesson No. 3”**. Теперь после выполнения **“Think on Lesson”** в ранее пустом третьем уроке появятся входные данные из первого, которым сопоставлены выходы сети. Весь этот урок можно экспортировать в Excel.

По экспортированным данным можно, например, посчитать точность кластеризации. Если за ответ нейронной сети взять класс, соответствующий выходу с максимальным значением, точность на обучающей выборке составляет 91.5%; см. файл *result.xls*.

Чтобы построить описательную статистику, скопируем результаты на новый лист, отсортируем по ответу нейронной сети, разобьем на три группы и удалим лишнюю информацию, после чего воспользуемся командой **«Анализ данных»**. Результат приведен в файле *result.xls* в таблице *STATS*. Легко увидеть, с чем связаны трудности разделения классов VERSICOL и VIRGINIC при помощи нейронной сети: все 4 параметра у обоих классов близки, в то время как у SETOSA параметры PETALLEN и PETALWID значительно меньше.

Лекция 15. Нейросети крупного калибра.

Библиотека машинного обучения ENCOG и пакет Azur

1. Общая характеристика пакета ENCOG

https://www.mql5.com/ru/articles/252
Библиотека ENCOG - пакет для работы с нейросетями и системой машинного обучения, разработанной Heaton Research.

Пакет ENCOG используется в различных коммерческих торговых системах. Он существует в двух вариантах: один написан на языке C#, другой на Java.

1. Библиотека ENCOG является бесплатной, ее исходные коды доступны. Если вы захотите посмотреть что происходит внутри нейросети, можно посмотреть исходники.
2. Пакет ENCOG очень хорошо документирован. Mr. Heaton, основатель компании Heaton Research, предоставил в открытый доступ онлайн курс по нейронным сетям, машинному обучению и использованию ENCOG для предсказания будущих данных. В дополнение к этому, на сайте Heaton Research доступны книги по программированию с использованием ENCOG на языках Java и C#. На сайте разработчика также доступна онлайн-документация по ENCOG.
3. ENCOG не является "мертвым" проектом. Разработана версия ENCOG 2.6, опубликована карта разработки (roadmap) ENCOG 3.0.
4. Библиотека ENCOG является робастной. Она очень хорошо спроектирована, поддерживает работу с несколькими процессорами/потоками для ускорения нейросетевых расчетов. Некоторые коды уже начали портироваться в OpenCL для расчета на графических процессорах (GPU).

На текущий момент ENCOG предоставляет следующие возможности:

Архитектуры нейронных сетей (Neural Network Architectures)

- Адаптивная линейная нейросеть (ADALINE Neural Network)
- Адаптивная резонансная теория (Adaptive Resonance Theory 1, ART1)
- Двунаправленная ассоциативная память (Bidirectional Associative Memory, BAM)
- Машина Больцмана (Boltzmann Machine)

- Нейросети встречного распространения (Counterpropagation Neural Network, CPN)
- Рекуррентные нейросети Элмана (Elman Recurrent Neural Network)
- Сеть прямого распространения/Многослойный персептрон (Feedforward Neural Network, Perceptron)
- Нейронные сети Хопфилда (Hopfield Neural Network)
- Рекуррентные сети Джордана (Jordan Recurrent Neural Network)
- Нейроэволюция нарастающей топологии (Neuroevolution of Augmenting Topologies, NEAT)
- Нейросеть на базе радиальных базисных функций (Radial Basis Function Network)
- Рекуррентные самоорганизующиеся карты (Recurrent Self Organizing Map, RSOM)
- Самоорганизующиеся карты Кохонена (Kohonen Self Organizing Map)

Методы обучения (Training Techniques)

- Backpropagation
- Resilient Propagation (RPROP) – обучение при помощи эластичного распространения
- Scaled Conjugate Gradient (SCG)
- Manhattan Update Rule Propagation
- Competitive Learning
- Hopfield Learning
- Levenberg Marquardt, LMA
- Genetic Algorithm Training
- Instar Training
- Outstar Training
- ADALINE Training
- Training Data Models
- Supervised
- Unsupervised
- Temporal (Prediction)
- Financial (загрузка данных с Yahoo Finance)
- SQL
- XML
- CSV
- Image Downsampling

2. Методы машинного обучения (Machine Learning Types)

- Прямое и простейшее рекуррентное обучение Элмана-Джордана (Feedforward and Simple Recurrent, Elman/Jordan)
- Генетические алгоритмы (Genetic Algorithms)
- Алгоритм нейроэволюции нарастающей топологии (NEAT)
- Вероятностная нейронная сеть/Нейросеть с общей регрессией (Probablistic Neural Network/General Regression Neural Network, PNN/GRNN)
- Самоорганизующиеся карты (Self Organizing Map, SOM/Kohonen)
- Метод Метрополиса/Алгоритм имитации отжига (Simulated Annealing)
- Метод опорных векторов (Support Vector Machine)

Активационные функции (Activation Functions)

- Competitive
- Sigmoid
- Hyperbolic Tangent
- Linear
- SoftMax
- Tangential
- Sin Wave
- Step
- Bipolar
- Gaussian

Генераторы случайных чисел (Randomization Techniques)

- Range Randomization
- Gaussian Random Numbers
- Fan-In
- Nguyen-Widrow

Планируются к реализации (Planned features):

- HyperNEAT
- Restrictive Boltzmann Machine (RBN/Deep Belief)
- Spiking Neural Networks

учебники по ENCOG, доступные на сайте Heaton Research

- Neural Network Calculation (Part 1): Feedforward Structure
- Neural Network Calculation (Part 2): Activation Functions & Basic Calculation
- Neural Network Calculation (Part 3): Feedforward Neural Network Calculation
- статья по нормализации

Подготовка входных данных

Для обучения нейросети библиотека Encog принимает файлы формата CSV.

Первая строка входного файла представляет собой заголовок, разделенный запятыми:

```
DATE,TIME,CLOSE,Indicator_Name1,Indicator_Name2,Indicator_Name3
```

В заголовке первые три колонки содержат дату, время и цену закрытия бара, следующие колонки содержат имена переменных. Строки файла с данными для обучения нейросети представляют собой данные, разделенные запятыми.

Для правильного расчета функций активации необходимо нормализовать исходные данные. Нормализация данных представляет собой математическое преобразование данных в диапазон [0..1] или [-1,1]. Нормализованные данные могут быть денормализованы, т.е. преобразованы обратно в начальный диапазон.

Денормализация требуется для преобразования выходных данных нейросети в вид, воспринимаемый человеком. ENCOG может самостоятельно нормализовать и денормализовать данные:

```
/**
         * Возвращает нормализованное значение числа.
         * @param value Число для нормализации.
         * @return Нормализованное значение.
         */
        public static double normalize(final int value) {
                return ((value - INPUT_LOW)
                                / (INPUT_HIGH - INPUT_LOW))
                                * (OUTPUT_HIGH - OUTPUT_LOW) + OUTPUT_LOW;
        }

        /**
         * Возвращает денормализованное значение числа.
         * @param value Число для денормализации.
         * @return Денормализованное значение.
         */
        public static double deNormalize(final double data) {
                double result = ((INPUT_LOW - INPUT_HIGH) * data - OUTPUT_HIGH
                                * INPUT_LOW + INPUT_HIGH * OUTPUT_LOW)
                                / (OUTPUT_LOW - OUTPUT_HIGH);
                return result;
        }
```

Результирующий файл, готовый для обучения нейросети, должен выглядеть следующим образом:

```
DATE,TIME,CLOSE,BullsPower
20110103,0000,0.93377000,-7.8970208860e-002
20110104,0000,0.94780000,-6.4962292188e-002
20110105,0000,0.96571000,-4.7640374727e-002
20110106,0000,0.96527000,-4.4878854587e-002
20110107,0000,0.96697000,-4.6178012364e-002
20110110,0000,0.96772000,-4.2078647318e-002
20110111,0000,0.97359000,-3.6029181466e-002
20110112,0000,0.96645000,-3.8335729509e-002
20110113,0000,0.96416000,-3.7054869514e-002
20110114,0000,0.96320000,-4.4259373120e-002
```

```
20110117,0000,0.96503000,-4.4835729773e-002
20110118,0000,0.96340000,-4.6420936126e-002
20110119,0000,0.95585000,-4.6868984125e-002
20110120,0000,0.96723000,-4.2709941621e-002
20110121,0000,0.95810000,-4.1918330800e-002
20110124,0000,0.94873000,-4.7722659418e-002
20110125,0000,0.94230000,-5.7111591557e-002
20110126,0000,0.94282000,-6.2231529077e-002
20110127,0000,0.94603000,-5.9997865295e-002
20110128,0000,0.94165000,-6.0378312069e-002
20110131,0000,0.94414000,-6.2038328069e-002
20110201,0000,0.93531000,-6.0710334438e-002
20110202,0000,0.94034000,-6.1446445012e-002
20110203,0000,0.94586000,-5.2580791504e-002
20110204,0000,0.95496000,-4.5246755566e-002
20110207,0000,0.95730000,-4.4439392954e-002
```

Выбор архитектуры нейросети

Для новичка выбор правильной архитектуры нейросети представляет собой сложную задачу. Рассмотрим трехслойную архитектуру нейронной сети прямого распространения: входной слой, один скрытый слой и выходной слой. Вы можете экспериментировать с большим числом слоев.

Для входного и выходного слоя мы сможем точно рассчитать количество требуемых нейронов. Для скрытого слоя мы попытаемся минимизировать ошибку нейросети при помощи алгоритма "forward selection". Вы можете попробовать использовать другие методы для определения количества нейронов в скрытом слое, например, генетические алгоритмы.

Другой метод, используемый в ENCOG называется "backward selection", или "обрезкой", в основном он имеет дело с оптимизацией связей между слоями и удаляет нейроны скрытых слоев с нулевыми весами, вы также можете его попробовать.

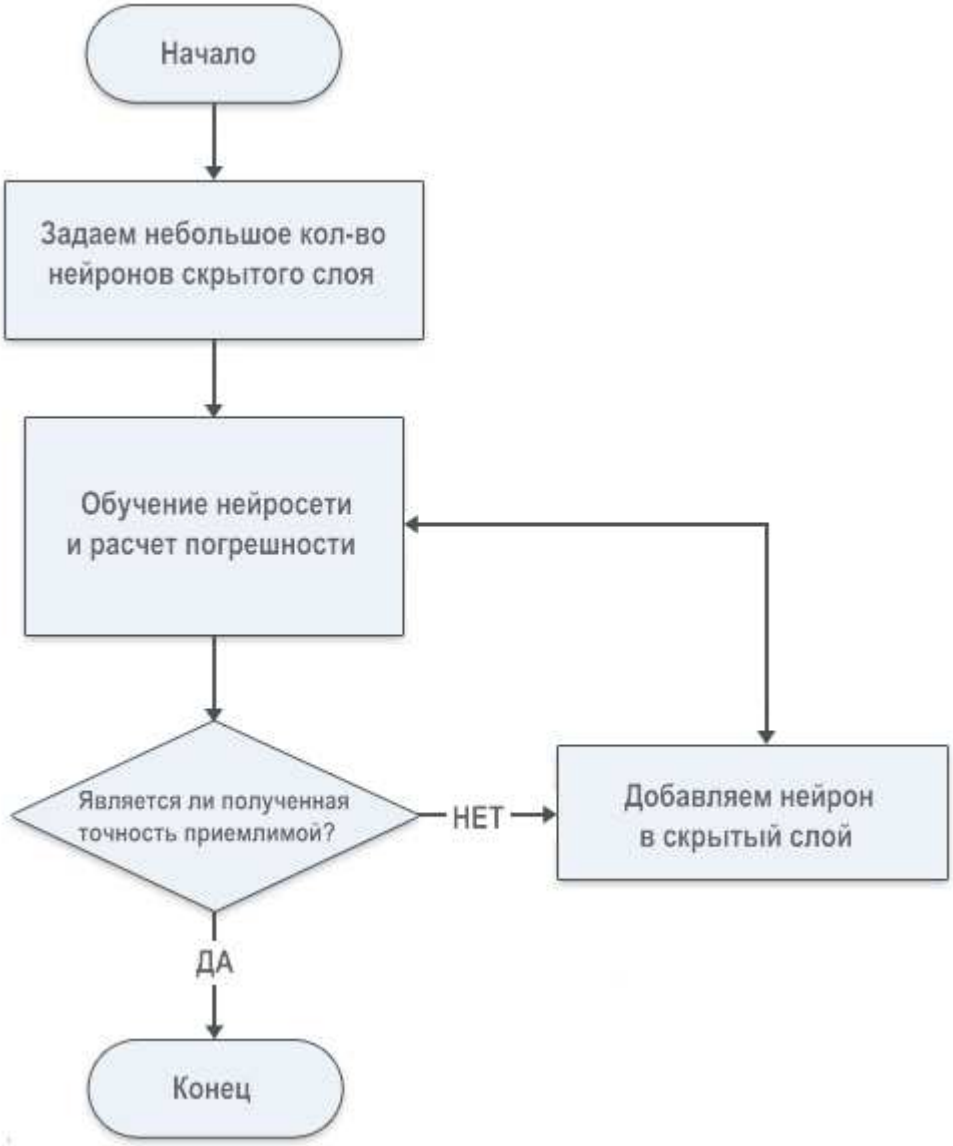
Входной слой

Поскольку данные подготавливались алгоритмом timeboxing, число нейронов входного слоя должно быть равно числу индикаторов, умноженному на число баров, используемых для предсказания следующего бара.

Если мы используем значения 3-х индикаторов и размер входного окна в 6 баров, входной слой нейросети будет содержать 18 нейронов. Данные, подготовленные алгоритмом timeboxing, затем предоставляются нейросети в качестве входных параметров.

Скрытый слой

Количество нейронов скрытого слоя должно определяться на основе погрешности расчета нейронной сети. Явного математического выражения для числа нейронов скрытого слоя не существует. До написания этой статьи, методом проб и ошибок я испробовал множество различных подходов, затем на сайте Heaton Research я нашел алгоритм, который помогает понять метод "forward selection":



Алгоритм "Forward selection" для определения числа нейронов скрытого слоя

Выходной слой

Для наших целей число нейронов выходного слоя - это количество баров, которые мы пытаемся предсказать. Запомните, что с увеличением количества нейронов скрытого и выходного слоев увеличивается и время обучения сети. В этой статье мы пытаемся предсказать один бар будущего, поэтому выходной слой содержит один нейрон.

Обучение нейронной сети

Алгоритм обучения нейросети уже реализован разработчиками ENOG на языке C#.

ENCOG 2.6 использует пространство имен Encog.App.Quant в качестве основы для предсказания финансовых временных рядов.

Архитектура нейросети и параметры обучения могут быть легко настроены изменением следующих переменных:

```
        /// <summary>

        /// Размер входного окна. Представляет собой количество баров, используемых
для предсказания следующего бара.

        /// </summary>
```

```
        public const int INPUT_WINDOW = 6;


        /// <summary>
        /// Количество баров в будущем, которые пытаемся предсказать. Обычно это всего
1 бар.
         /// Предсказание будущего на 1 бар работает лучше всего.
        /// </summary>
        public const int PREDICT_WINDOW = 1;


        /// <summary>
        /// Количество баров вперед, используемых для получения лучшего результата.
        /// </summary>
        public const int RESULT_WINDOW = 5;


        /// <summary>
        /// Количество нейронов в первом скрытом слое.
        /// </summary>
        public const int HIDDEN1_NEURONS = 12;


        /// <summary>
        /// Ошибка обучения нейросети.
        /// </summary>
        public const double TARGET_ERROR = 0.01;
```

Код не нуждается в комментариях, поэтому лучше внимательно его посмотреть.

В зависимости от сложности нейросети и размера обучающей выборки, процесс обучения может занимать от нескольких минут до часов или даже дней.

Имейте ввиду, что возможны случаи, когда нейронная сеть не может быть обучена до некоторой точности из-за того, что количество нейронов слишком мало.

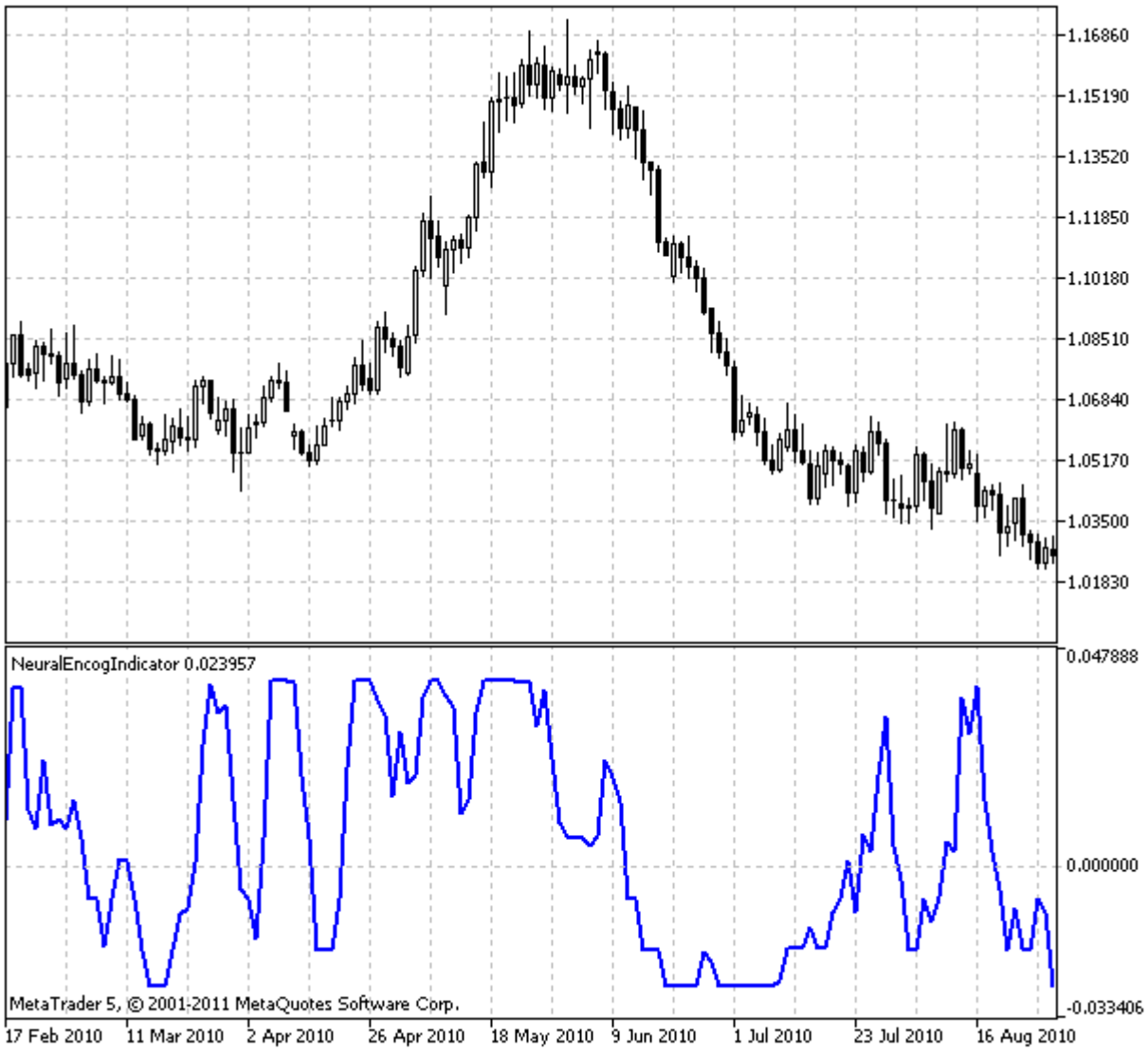
Прогнозирование временных рядов в MetaTrader 5 при помощи библиотеки машинного обучения ENCOG

Обученная нейронная сеть может быть использована в нейросетевом индикаторе, который будет пытаться предсказать изменение цены (best return on investment).

Нейросетевой индикатор ENCOG для MetaTrader 5 состоит из двух частей. Первая часть написана на MQL5, главным образом, она берет значения тех же самых индикаторов, которые использовались нами для обучения нейросети и отдает их нейронной сети в качестве входных параметров.

Вторая часть написана на C#, она подготавливает входные данные, осуществляет timeboxing, производит расчет и возвращает результат работы нейросети в MQL5.

Результат работы индикатора, обученного на дневных данных пары USDCHF и данных индикаторов Stochastic и Williams %R приведен на рисунке:



Нейросетевой индикатор на базе Encog

Индикатор показывает предсказанные приращения цены (returns on investment) для следующего бара.

Нейросетевой индикатор готов, теперь мы готовы для создания советника на базе этого индикатора.

Советник, основанный на нейросетевом индикаторе

Советник берет выходные значения нейросетевого индикатора и принимает решение о покупке или продаже финансового инструмента. Сначала предполагалось, что покупать следует, пока индикатор больше нулевой линии, и продавать при отрицательных значениях индикатора. Смысл заключается в покупке при предсказании положительного приращения цены и продаже в случае предсказания отрицательного изменения цены.

После некоторого тестирования выяснилось, что эффективность можно повысить, поэтому я ввел переменные "strong uptrend" and "strong downtrend", используемые для того чтобы не выходить с рынка в случаях сильных трендов, в соответствии с правилом "trend is your friend".

Кроме того, на форуме Heaton Research мне посоветовали использовать ATR для скользящих уровней stop-loss, поэтому я использовал индикатор Chandelier ATR, опубликованный на форуме по MQL5.

Код советника приведен ниже.

Советник был протестирован на дневных барах пары USDCHF. Для обучения нейросети было использовано около 50% данных.

Результаты тестирования советника

Ниже я приведу результаты тестирования на истории. Тест проводился с 2000.01.01 по 2011.03.26.

Начальный депозит	10 000.00				
Бары	2900	Тики	14109128		
Чистая прибыль	14 369.15	Общая прибыль	80 573.12	Общий убыток	-66 203.97
Прибыльность	1.22	Матожидание выигрыша	35.66		
Фактор восстановления	2.13	Коэффициент Шарпа	0.07		
Абсолютная просадка по балансу	3 558.60	Максимальная просадка по балансу	5 647.61 (22....	Относительная просадка по балансу	41.64% (4 596...
Абсолютная просадка по средствам	3 776.66	Максимальная просадка по средствам	6 760.14 (26....	Относительная просадка по средствам	47.47% (5 623...
Всего трейдов	403	Короткие трейды (% выигравших)	202 (43.56%)	Длинные трейды (% выигравших)	201 (40.30%)
Всего сделок	806	Прибыльные трейды (% от всех)	169 (41.94%)	Убыточные трейды (% от всех)	234 (58.06%)
Самый большой		прибыльный трейд	3 258.69	убыточный трейд	-973.72
Средний		прибыльный трейд	476.76	убыточный трейд	-282.92
Максимальное количество		непрерывных выигрышей (прибыль)	8 (1 490.15)	непрерывных проигрышей (убыток)	11 (-3 749.50)
Максимальная		непрерывная прибыль (число выигрышей)	3 698.71 (3)	непрерывный убыток (число проигрышей)	-3 749.50 (11)
Средний		непрерывный выигрыш	2	непрерывный проигрыш	2

Рисунок. Результаты тестирования нейросетевого советника на исторических данных

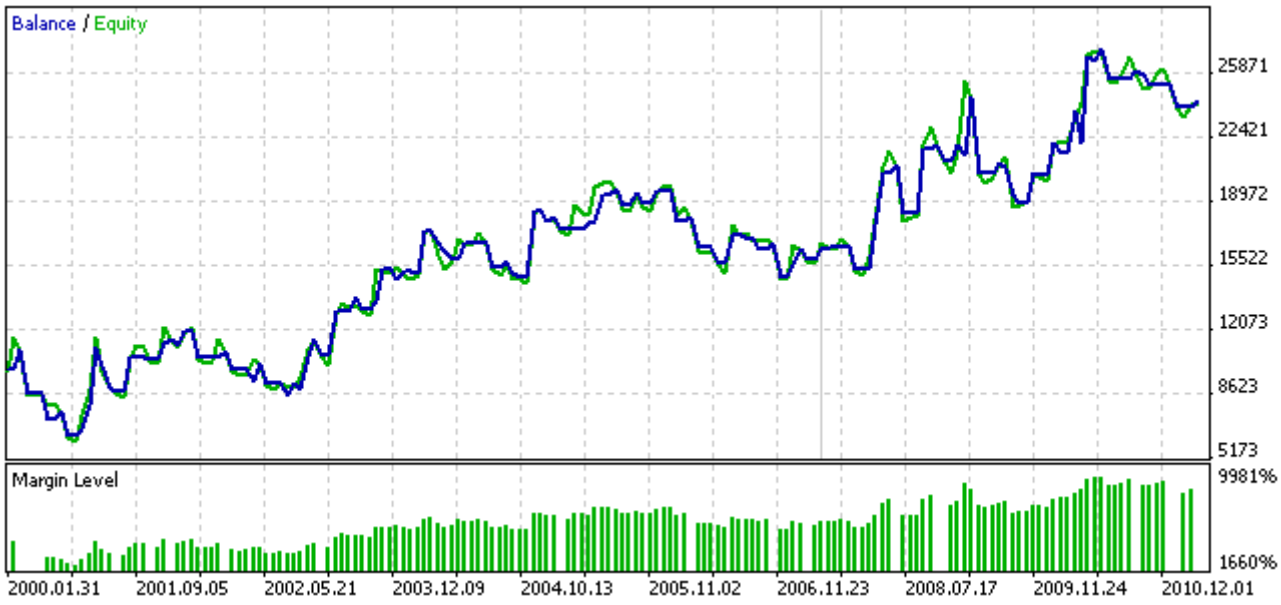


График Баланса/Средств нейросетевого советника

Просьба использовать данный советник лишь в образовательных целях как отправную точку для дальнейших исследований.

Выводы

Здесь представлен способ создания нейросетевого прогнозирующего индикатора, созданного при помощи библиотеки машинного обучения ENCOG и советника, построенного на базе данного индикатора.

Литература.

1. https://www.mql5.com/ru/articles/252
2. Neural Network Calculation (Part 1): Feedforward Structure
3. Neural Network Calculation (Part 2): Activation Functions & Basic Calculation
4. Neural Network Calculation (Part 3): Feedforward Neural Network Calculation
5. статья по нормализации

2. Пакет Windows Azure для Windows Server

https://azure.microsoft.com/ru-ru/documentation/articles/machine-learning-azure-ml-netsharp-reference-guide/

Пакет Windows Azure для Windows Server представляет собой набор технологий Windows Azure, доступных клиентам Microsoft без дополнительных затрат по их установке в центре обработки данных. Они выполняются на базе Windows Server 2012 R2 и System Center 2012 R2 и на основе технологий Windows Azure позволяют предложить клиентам полнофункциональную, облачную, мультитенантную систему с самообслуживанием, соответствующую стилю работы в Windows Azure.

Пакет Windows Azure предоставляет следующие возможности.

- Портал с самообслуживанием для подготовки, наблюдения и управления службами, например, службами облаков веб-сайтов, виртуальных машин и шины обслуживания.
- Портал управления, где можно настраивать облака ресурсов, учетные записи пользователей, предложения клиентов, квоты и расценки, а также управлять ими.
- API управления службами — API REST, который позволяет реализовать набор сценариев интеграции, включая построение пользовательских порталов и систем выставления счетов.
- Служба облаков веб-сайтов, которая позволяет предоставить масштабируемую, общую платформу веб-хостинга высокой плотности для веб-приложений ASP.NET, PHP и Node.js. Служба облаков веб-сайтов содержит настраиваемую коллекцию веб-приложений из проектов с открытым исходным кодом.
- Служба облаков виртуальных машин, предоставляющая возможности инфраструктуры как услуги (IaaS) для виртуальных машин Windows и Linux.
- SQL и MySQL — это службы, предоставляющие экземпляры базы данных. Эти базы данных можно использовать совместно со службами веб-сайтов.

Терминология Windows Azure Pack

Язык спецификаций нейронных сетей Net# для машинного обучения Azure

Обзор

- Net# — это язык, разработанный Майкрософт для использования при определении архитектур нейронных сетей для модулей нейронных сетей, которые применяются в Машинном обучении Microsoft Azure.

Для служб машинного обучения Azure доступна бесплатная пробная версия.

Ни кредитная карта, ни подписка на Azure не требуются. **Начните работу прямо сейчас >**

Основы нейронных сетей

Структура нейронной сети состоит из ***узлов***, организованных в ***слои***, и взвешенных ***подключений*** (или ***переходов***) между узлами. Подключения при этом являются направленными, и для каждого из них задан узел ***источника*** и узел ***назначения***.

У каждого ***обучаемого слоя*** (скрытого или выходного) есть один или несколько ***пакетов подключений***. Пакет подключений состоит из слоя источника и спецификации подключений из этого слоя источника. Все подключения в любом пакет совместно используют один и тот же ***слой источника*** и тот же ***слой назначения***. В Net# считается, что пакет подключений относится к слою назначения пакета.

Net# поддерживает различные виды пакетов подключений, которые позволяют настраивать способ сопоставления входов со скрытыми слоями и выходами.

Пакет по умолчанию или стандартный пакет называется **полным пакетом**. В таком пакете каждый узел в слое источника подключается к каждому узлу в слое назначения.

В дополнение к этому, Net# поддерживает следующие четыре типа дополнительных пакетов подключения:

- **Фильтрованные пакеты**. Пользователь может задавать предикат с использованием мест назначения узла слоя источника и узла слоя назначения. Узлы подключаются, если предикат имеет значение True.
- **Сверточные пакеты**. Пользователь может задавать небольшие окрестности узлов в слое источника. Каждый узел в слое назначения подключается к одной окрестности узлов в слое источника.
- **Группирующие пакеты** и **пакеты нормализации ответов**. Эти пакеты аналогичны сверточным пакетам в том смысле, что пользователь определяет небольшие окрестности узлов в слое источника. Различие заключается в том, что в таких пакетах не поддерживается обучение по весам переходов. Вместо этого для определения значения узла назначения к узлу источника применяется предопределенная функция.

Использование Net# для определения структуры нейронной сети позволяет задавать такие сложные структуры, как глубокие нейронные сети или свертки произвольных размеров для улучшения обучения на основе данных — изображений, аудио и видео.

Поддерживаемые настройки

Архитектура моделей нейронных сетей, создаваемых в Машинном обучении Azure, может широко настраиваться с помощью Net#. Вы можете:

- Создавать скрытые слои и управлять количеством узлов в каждом слое.
- Задавать способ подключения слоев друг к другу.
- Определять специальные структуры подключения, такие как свертки и пакеты с распределением весов.
- Задавать различные функции активации.

Подробную информацию о синтаксисе языка спецификаций см. в разделе Спецификация структуры.

Примеры определения нейронных сетей для некоторых общих задач машинного обучения от простого к сложному см. в разделе Примеры.

Общие требования

Примеры использования Net

В этом разделе приводится несколько примеров использования Net# для добавления скрытых слоев, определения способов взаимодействия скрытых слоев с другими слоями и построения сверточных сетей.

Определение простой настраиваемой нейронной сети: пример «Привет, мир!»

В этом простом примере показано, как создавать модель нейронной сети с одним скрытым слоем.

```
input Data auto;
hidden H [200] from Data all;
output Out [10] sigmoid from H all;
```

В примере проиллюстрированы некоторые базовые команды и их порядок.

- В первой строке определяется входной слой (с именем *Data*). При использовании ключевого слова **auto** в нейронную сеть автоматически включаются все столбцы функций в примерах ввода.
- Вторая строка создает скрытый слой. Имя *H* назначается скрытому слою с 200 узлами. Этот слой полностью подключен ко входному слою.
- Третья строка определяет выходной слой (с именем *O*), который содержит 10 выходных узлов. Для классификации нейронных сетей используется по одному выходному узлу на класс. Ключевое слово **sigmoid** указывает выходную функцию, примененную к выходному слою.

Определение нескольких скрытых слоев: пример машинного зрения

В следующем примере показано, как определять немного более сложную нейронную сеть с несколькими настраиваемыми скрытыми слоями.

```
// Define the input layers
input Pixels [10, 20];
```

```
input MetaData [7];

// Define the first two hidden layers, using data only from the Pixels input
hidden ByRow [10, 12] from Pixels where (s,d) => s[0] == d[0];
hidden ByCol [5, 20] from Pixels where (s,d) => abs(s[1] - d[1]) <= 1;

// Define the third hidden layer, which uses as source the hidden layers
ByRow and ByCol
hidden Gather [100]
{
  from ByRow all;
  from ByCol all;
}

// Define the output layer and its sources
output Result [10]
{
  from Gather all;
  from MetaData all;
}
```

В примере проиллюстрировано несколько признаков языка спецификаций нейронных сетей

- Структура имеет два входных слоя: *Pixels* и *MetaData*.
- Слой *Pixels_ — это слой источника для двух пакетов подключений со слоями назначения: _ByRow* и *ByCol*.
- Слои *Gather* и _Result_ — это слои назначения в нескольких пакетах подключений.
- Выходной слой *Result* представляет собой слой назначения в двух пакетах подключений: один — со скрытым слоем второго уровня (Gather) в качестве слоя назначения и другой — с входным слоем (MetaData) в качестве слоя назначения.
- Скрытые слои *ByRow* и *ByCol* определяют отфильтрованные подключения с использованием выражений предиката. Если говорить точнее, то узел в *ByRow* с координатами [x, y] подключается к тем узлам в *Pixels*, у которых первая координата индекса равна первой координате узла (x). Аналогично узел ByCol с координатами [x, y] подключается к тем узлам в *Pixels*, у которых вторая координата индекса находится в пределах единицы от второй координаты узла (y).

Определение сверточной сети для многоклассовой классификации: пример распознавания цифр

Определение следующей сети, разработанной для распознавания цифр, показывает некоторые усовершенствованные методы настройки нейронных сетей.

```
input Image [29, 29];
hidden Conv1 [5, 13, 13] from Image convolve
{
   InputShape  = [29, 29];
   KernelShape = [ 5,  5];
   Stride      = [ 2,  2];
   MapCount    = 5;
}
hidden Conv2 [50, 5, 5]
from Conv1 convolve
{
   InputShape  = [ 5, 13, 13];
   KernelShape = [ 1,  5,  5];
   Stride      = [ 1,  2,  2];
```

```
    Sharing     = [false, true, true];
    MapCount    = 10;
}
hidden Hid3 [100] from Conv2 all;
output Digit [10] from Hid3 all;
```

- Структура имеет один входной слой — *изображение*.
- Ключевое слово **convolve** указывает, что *Conv1* и *Conv2* — сверточные слои. За объявлением каждого из этих слоев следует список атрибутов свертки.
- У сети имеется третий скрытый слой *Hid3*, который полностью подключен ко второму скрытому слою *Conv2*.
- Выходной слой *Digit* подключен к третьему скрытому слою *Hid3*. Ключевое слово **all** указывает, что выходной слой полностью подключен к *Hid3*.
- Арность свертки — три (длина кортежей **InputShape**, **KernelShape**, **Stride** и **Sharing**).
- Количество весов на ядро равно: *1 + **KernelShape**[0] * **KernelShape**[1] * **KernelShape**[2] = 1 + 1 * 5 * 5 = 26. Или 26 * 50 = 1300*.
- Количество узлов в каждом скрытом слое можно вычислить следующим образом:
- **NodeCount**[0] = (5 - 1) / 1 + 1 = 5.
- **NodeCount**[1] = (13 – 5) / 2 + 1 = 5.
- **NodeCount**[2] = (13 - 5) / 2 + 1 = 5.
- Общее количество узлов можно вычислить, используя объявленную размерность слоя [50, 5, 5] следующим образом: ***MapCount** * **NodeCount**[0] * **NodeCount**[1] * **NodeCount**[2] = 10 * 5 * 5 * 5*
- Так как **Sharing**[d] имеет значение False только для *d == 0*, количество ядер равно ***MapCount** * **NodeCount**[0] = 10 * 5 = 50*.

Эксперимент 5 шагов.

Литература.

1. Нейронные сети в Azure ML. Введение в Net#.
2. https://azure.microsoft.com/ru-ru/documentation/articles/machine-learning-create-experiment/
3. https://technet.microsoft.com/ru-ru/library/dn296435.aspx
4. Библиотека TechNet
5. Windows Azure Pack для Windows Server
6. Миграция со служб Windows Azure для Windows Server
7. Развертывание Windows Azure Pack для Windows Server
8. Подготовка и настройка служб в Windows Azure Pack
9. Администрирование Windows Azure Pack для Windows Server
10. Терминология Windows Azure Pack
11. Скрипты Windows Azure Pack и командлеты PowerShell
12. Windows Azure Pack troubleshooting
13. Сведения о специальных возможностях для Windows Azure Pack
14. https://azure.microsoft.com/ru-ru/documentation/articles/machine-learning-azure-ml-netsharp-reference-guide/
15. В.О. Сафонов **Платформа облачных вычислений Microsoft Windows Azure** http://www.intuit.ru/department/se/pmsazure/1/

Машинное обучение, инициализация модели и регрессии

Машинное обучение, инициализация модели и классификация

Машинное обучение, инициализация модели и кластеризация

Text analytics modules

Модули библиотеки OpenCV

Описание модуля обучения компьютера

Лекция 16. Современные направления развития нейрокомпьютерных технологий

Современные направления развития нейрокомпьютерных технологий
(Пространства имён. Свёрточные сети, Deep learning и Caffe)

Пространство имён NeuronDotNet

http://www.innovasys.com/products/dx2010/csdocumentation.aspx?cpid=gawdxcs&gclid=CNGlk8_8sqECFQceZwodsiHz_g

Проектирование Нейронной Сети.

Проектирование искусственной нейроннной сети для конкретного приложения включает выбор правильного типа сети, определение количества скрытых слоёв сети, подходящего метода инициализации весов связей, подходящего обучающего алгоритма, количества эпох обучения и размера обучающей выборки. Большинство из этих параметров зависит от приложения для которого нейронная сеть разрабатывается.

Вот основные принципы проектирования нейронной сети.

Количество скрытых слоев в Сетях Backpropagation.

Сеть Backpropagation без скрытых слоев не может выполнить нелинейную классификацию. (Не может быть использована, как функциональный апроксиматор XOR). Так, один скрытый слой - обязателен для сети backpropagation. Математически доказано что сеть backpropagation с единственным скрытым слоем может быть использована для аппроксимации любой функции. Такой единственный скрытый слой является наилучшим выбором в большинстве случаев.

Дополнительные скрытые слои ускоряют изучение процесса и обучаемая сеть точно соответствует обучающим образцам, но появляется эффект overtraining (переобучение), при котором подготовленная сеть стремится запоминать обучающие образцы вместо изучения представляемых ими образов.

Инициализация нейронных сетей.

Соответствующая инициализация связей между нейронами (синапсов) существенно облегчает обучение сети. Обычно, веса связей инициализируются случайными величинами между -0.5 и +0.5 (более высокие начальные величины стремятся к насыщению после активизации).

Простая модификация этой инициализации, способствующая значительно более быстрому обучению сети, предлагалась Nguyen и Widrow.

NeuronDotNet ОСУЩЕСТВЛЯЕТ процесс инициализации как сменный модуль. Другие алгоритмы инициализации могут быть реализованы при выполнении интерфейса IInitializer.

Количество обучающих примеров

Определяет, как хорошо обучающая выборка представляет исследуемую функцию. Большее количество обучающих примеров увеличивает эффективность обучения сети. Обычно «средняя квадратическая ошибка» слегка возрастает с ростом размера обучающей выборки, в то же самое время она может уменьшаться, увеличивая эффективность обучаемости сети.

Установка среды разработки

Обзор класса NeuronDotNet.

NeuronDotNet предназначен для отображения компонентов нейронной сети в библиотечные классы и интерфейсы. Подробная документация может быть загружена вместе с источником NeuronDotNet.

(Слои, разъемы, сети и принадлежность TrainingSet к интерфейсу ISerializable).

Пространство имён NeuronDotNet.Core.

- **INeuron** : интерфейс для представления нейрона
- **ISynapse** : интерфейс, представляющий связи (synapse) нейронной сети
- **ILayer** : интерфейс, представляющий слои нейронной сети
- **IConnector** : интерфейс, представляющий connector (который соединяет два слоя) нейронной сети

- **INetwork** : интерфейс для представления нейронной сети

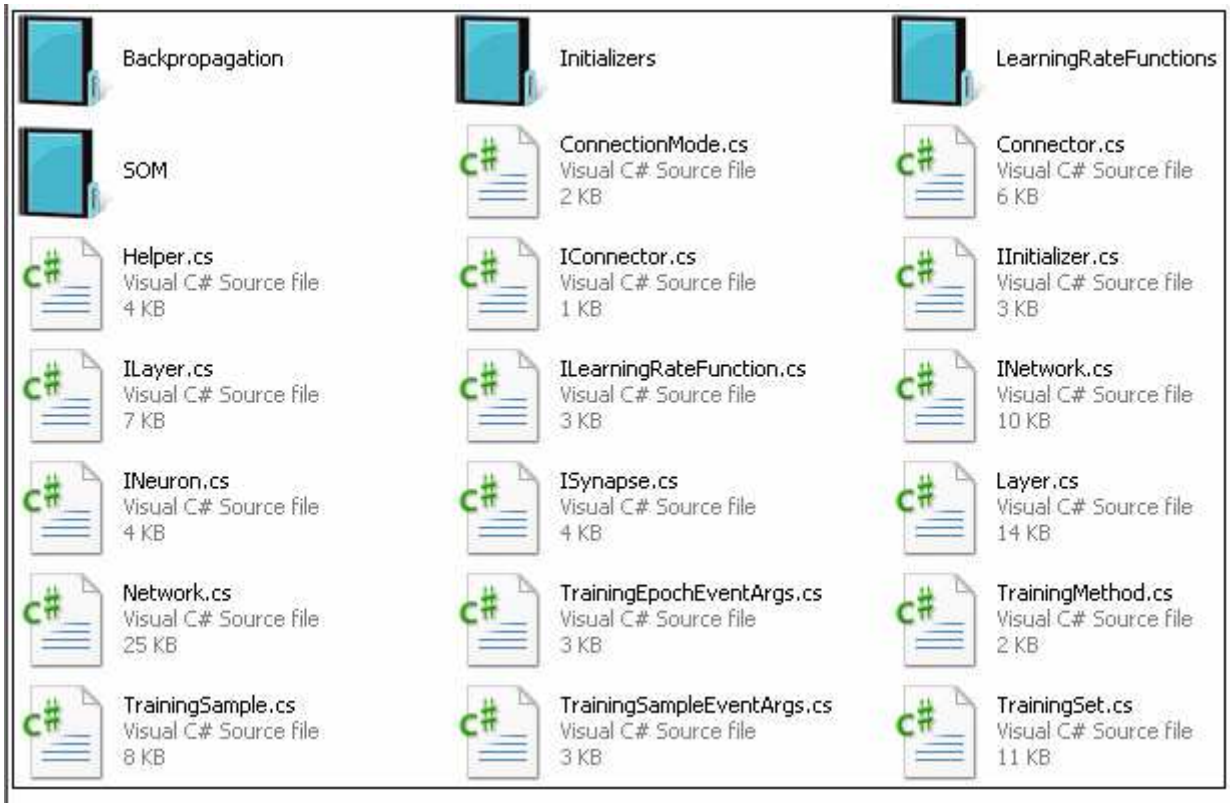
- **IInitializer** : интерфейс для представления инициализатора, который определяет методы инициализации для конкретных слоёв и коннекторов.
- **ILearningRateFunction** : Интерфейс показателя функции обучаемости (Learning Rate Function interface).

- **Layer** : Абстрактная реализация 'ILayer, как генерация коллекции нейронов ('INeuron's)
- **Connector** : Абстрактная реализация 'IConnector', соединяющая два слоя
- **Network** : Абстрактный базовый класс основной нейронной сети. Он реализует 'INetwork'.
- **TrainingSample** : Обучающая выборка, представляющая пару векторов: вход – и ожидаемый выход.
- **TrainingSet** :Тренировочная выборка, как коллекция пренировочных примеров..

- **TrainingMethod** : Указание, какой метод обучения используется: с учителем, или без учителя (supervised or unsupervised).

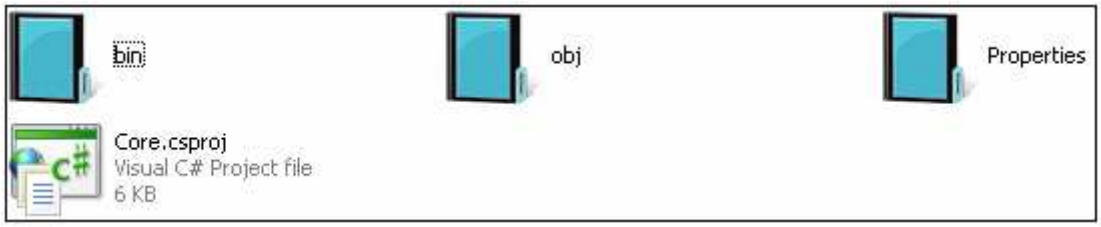
- **ConnectionMode** : Указание на то, какой коннектор используется: один-один, или полный.

Весь пакет NeuronDotNet в виде исходных текстов собран в папке Core, содержащей:



Здесь видно, что пространство имён Core содержит 17 cs-файлов и 4 пространства имён в виде папок: Backpropagation, Initializers, LearningRateFunctions, SOM, которые представляют собой ещё 4 пространства имён (их имена совпадают с именами папок).

Кроме того, в дистрибутиве пространства имён Core содержится программный проект:



В его состав входит файл NeuronDotNet.Core.dll – откомпилированная динамическая библиотека пакета NeuronDotNet .

Некоторые полезные кодовые фрагменты.

Данный ниже код создает сеть backpropagation с линейным входным слоем, содержащем десять нейронов, сигмоидный промежуточный слой, содержащий пять нейронов и сигмоидный выходной слой, содержащий семь нейронов.

```
LinearLayer inputLayer = new LinearLayer(10);
SigmoidLayer hiddenLayer = new SigmoidLayer(5);
SigmoidLayer outputLayer = new SigmoidLayer(7);
new BackpropagationConnector(inputLayer, hiddenLayer);
new BackpropagationConnector(hiddenLayer, outputLayer);
BackpropagationNetwork network = new BackpropagationNetwork(inputLayer, outputLayer);
```

Для того, чтобы создать сеть, сначала создаются слои (layers), затем соединяются слои, создавая разъемы (connectors), затем создают сеть, определяя желаемый входной и выходной слои. Этот порядок, если нет других указаний, может провести к полному malfunctioning сети. Пользователь НЕ должен модифицировать структуру сети после её создания.

Нужно помнить, что входной слой сети обратного распространения всегда линейный, чтобы не модифицировать пользовательский вход при добавлении bias или activation функций.

Коды для создания сети Kohonen SOMмогут быть следующими:

```
KohonenLayer inputLayer = new KohonenLayer(2);
KohonenLayer outputLayer = new KohonenLayer(new Size(neuronCount, 1));
KohonenConnector connector = new KohonenConnector(inputLayer, outputLayer);
KohonenNetwork network = new KohonenNetwork(inputLayer, outputLayer);
```

При необходимости свойства некоторых слоёв или connector могут быть изменены.

```
outputLayer.Initializer = new ZeroFunction();
outputLayer.NeighborhoodFunction = new GaussianFunction(10);
outputLayer.IsRowCircular = true;

connector.Initializer = new RandomFunction(-1, 1);
```

Обучающая функция связана с каждым слоем в сети. Мы можем модифицировать свойства индивидуально для каждого слоя. Если мы хотим использовать единственную функцию для всех слоев, мы можем использовать метод SetLearningRate сети.

```
network.SetLearningRate(0.3);
```

Для обучения сети необходимы обучающие примеры. Коллекция обучающих примеров объединяется в обучающую выборку (файл), который и используется для обучения сети.

Сеть обучается на всех обучающих примерах один-за-одним, повторяя их много раз, пока сеть не обучится полностью.

```
TrainingSet trainingSet = new TrainingSet(2, 1);
trainingSet.Add(new TrainingSample(new double[2] { 0d, 0d }, new double[1] { 0d }));
trainingSet.Add(new TrainingSample(new double[2] { 0d, 1d }, new double[1] { 1d }));
trainingSet.Add(new TrainingSample(new double[2] { 1d, 0d }, new double[1] { 1d }));
trainingSet.Add(new TrainingSample(new double[2] { 1d, 1d }, new double[1] { 0d }));
network.Learn(trainingSet, 1000);
```

Метод Initialize ре-инициализирует (unlearns) связи и нейроны, подготавливая их к обновлению обучения. BeginEpochEvent, EndEpochEvent, BeginSampleEvent и EndSampleEvent - некоторые события, которые помогают потребителю, чтобы изучать поведение сети во время обучения.

Обучающая выборка и все сети используют интерфейс 'ISerializable', благодаря чему могут быть сохранены в файле для последующего использования

Структура папок

```
Серийный номер тома: 0006EFC4 3009:ADA7
C:.
|   1111.doc
|   ConnectionMode.cs
|   Connector.cs
|   Core.csproj
|   Helper.cs
|   IConnector.cs
|   IInitializer.cs
|   ILayer.cs
|   ILearningRateFunction.cs
|   INetwork.cs
|   INeuron.cs
|   ISynapse.cs
|   Layer.cs
|   Network.cs
|   TrainingEpochEventArgs.cs
|   TrainingMethod.cs
|   TrainingSample.cs
|   TrainingSampleEventArgs.cs
|   TrainingSet.cs
|
+---Backpropagation
|       ActivationLayer.cs
|       ActivationNeuron.cs
|       BackpropagationConnector.cs
|       BackpropagationNetwork.cs
|       BackpropagationSynapse.cs
|       LinearLayer.cs
|       LogarithmLayer.cs
|       SigmoidLayer.cs
|       SineLayer.cs
|       TanhLayer.cs
|
+---bin
|   \---Release
|           NeuronDotNet.Core.dll
|           NeuronDotNet.Core.xml
|
+---Initializers
|       ConstantFunction.cs
|       NguyenWidrowFunction.cs
|       NormalizedRandomFunction.cs
|       RandomFunction.cs
|       ZeroFunction.cs
|
+---LearningRateFunctions
|       AbstractFunction.cs
|       ExponentialFunction.cs
|       HyperbolicFunction.cs
|       LinearFunction.cs
|
+---Properties
|       AssemblyInfo.cs
|
\---SOM
    |   INeighborhoodFunction.cs
    |   KohonenConnector.cs
    |   KohonenLayer.cs
    |   KohonenNetwork.cs
```

```
|   KohonenSynapse.cs
|   LatticeTopology.cs
|   PositionNeuron.cs
|
└──NeighborhoodFunctions
       GaussianFunction.cs
       MexicanHatFunction.cs
```

TSP Solver Source

SOM Demo Source

Свёрточные сети, Deep learning и Caffe

Для работы со свёрточными сетями и с глубоким обучением нейросетей существует много разных библиотек и фреймворков. Какими из них можно пользоваться, обычно зависит от:

1. наличия и доступности туториалов,
2. легкости освоения,
3. легкости разворачивания,
4. наличия активного сообщества.

По всем этим параметрам отлично подходит пакет Caffe:

1. Хорошие туториалы есть на их сайте. Отдельно рекомендуются лекции из Caffe Summer Bootcamp. Для быстрого старта можно почитать про основания нейронных сетей и потом про Caffe.
2. Для начала работы с Caffe даже не требуется язык программирования. Конфигурируется Caffe с помощью конфигурационных файлов, а запускается из командной строки.
3. Для разворачивания есть chef-кукбук и docker-образы.
4. На гитхабе ведется активная разработка, а в гугл-группе можно задать вопрос по использованию фреймворка.

К тому же Caffe очень быстрый, т.к. использует GPU (хотя можно обойтись и CPU).

Разработка Caffe ведется с сентября 2013 г. Начало разработки положил Yangqing Jia во время его обучения в калифорнийском университете в Беркли. С указанного момента Caffe активно поддерживается Центром Зрения и Обучения Беркли (The Berkeley Vision and Learning Center, BVLC) и сообществом разработчиков на GitHub. Библиотека распространяется под лицензией BSD 2-Clause.

Caffe реализована с использованием языка программирования C++, имеются обертки на Python и MATLAB. Официально поддерживаемые операционные системы — Linux и OS X, также имеется неофициальный порт на Windows. Caffe использует библиотеку BLAS (ATLAS, Intel MKL, OpenBLAS) для векторных и матричных вычислений. Наряду с этим, в число внешних зависимостей входят glog, gflags, OpenCV, protoBuf, boost, leveldb, nappy, hdf5, lmdb. Для ускорения вычислений Caffe может быть запущена на GPU с использованием базовых возможностей технологии CUDA или библиотеки примитивов глубокого обучения cuDNN.

Разработчики Caffe поддерживают возможности создания, обучения и тестирования полностью связанных и сверточных нейросетей. Входные данные и

преобразования описываются понятием слоя. В зависимости от формата хранения могут использоваться следующие типы слоев исходных данных:

DATA — определяет слой данных в формате leveldb и lmdb.

HDF5_DATA — слой данных в формате hdf5.

IMAGE_DATA — простой формат, который предполагает, что в файле приведен список изображений с указанием метки класса.

и другие.

Преобразования могут быть заданы с помощью слоев:

INNER_PRODUCT — полностью связанный слой.

CONVOLUTION — сверточный слой.

POOLING — слой пространственного объединения.

Local Response Normalization (LRN) — слой локальной нормализации.

Наряду с этим, при формировании преобразований могут использоваться различные функции активации.

Положительная часть (Rectified-Linear Unit, ReLU).

Сигмоидальная функция (SIGMOID).

Гиперболический тангенс (TANH).

Абсолютное значение (ABSVAL).

Возведение в степень (POWER).

Функция биноминального нормального логарифмического правдоподобия (binomial normal log likelihood, BNLL).

Последний слой нейросетевой модели должен содержать функцию ошибки. В библиотеке имеются следующие функции:

Среднеквадратичная ошибка (Mean-Square Error, MSE).

Краевая ошибка (Hinge loss).

Логистическая функция ошибки (Logistic loss).

Функция прироста информации (Info gain loss).

Сигмоидальная кросс-энтропия (Sigmoid cross entropy loss).

Softmax-функция. Обобщает сигмоидальную кросс-энтропию на случай количества классов больше двух.

В процессе обучения моделей применяются различные методы оптимизации. Разработчики Caffe предоставляют реализацию ряда методов:

Стохастический градиентный спуск (Stochastic Gradient Descent, SGD) [6].

Алгоритм с адаптивной скоростью обучения (adaptive gradient learning rate algorithm, AdaGrad) [7].

Ускоренный градиентный спуск Нестерова (Nesterov’s Accelerated Gradient Descent, NAG) [8].

В библиотеке Caffe топология нейросетей, исходные данные и способ обучения задаются с помощью конфигурационных файлов в формате prototxt. Файл содержит описание входных данных (тренировочных и тестовых) и слоев нейронной сети. Рассмотрим этапы построения таких файлов на примере сети “логистическая регрессия” (рис. 2). Далее будем считать, что файл называется linear_regression.prototxt, и он размещается в директории examples/mnist.

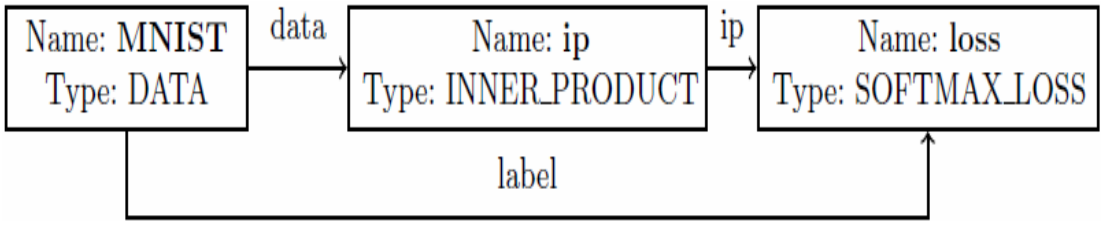


Рис. 2. Структура нейронной сети

Работа с пакетом.

1. Зададим имя сети.

```
name: "LinearRegression"
```

2. В качестве обучающего множества используется база данных MNIST, хранящаяся в формате lmdb. Для работы с форматами lmdb или leveldb используется слой типа “DATA”, в котором необходимо указать некоторые параметры, описывающие входные данные (data_param): путь до данных на жестком диске (source), тип данных (backend), размер выборки (batch_size). Также с данными можно производить различные преобразования (transform_param). Например, можно произвести нормировку изображения, умножив все значения на 0.00390625 (число, обратное к 255). В параметре top указывается одно или несколько имен, которые будут использованы для идентификации выхода слоя. В данном примере это обработанные изображения (data) и метки классов, которым принадлежат изображения (label).

```
layers {
  name: "mnist"
  type: DATA
  top: "data"
  top: "label"
  data_param {
    source: "examples/mnist/mnist_train_lmdb"
    backend: LMDB
    batch_size: 64
  }
  transform_param {
    scale: 0.00390625
  }
}
```

3. Определим полносвязный слой (выход каждого нейрона предыдущего слоя связан с входом каждого нейрона последующего слоя). Полносвязный слой в библиотеке Caffe задается с помощью слоя типа INNER_PRODUCT. Имя входных данных указывается с помощью параметра bottom. В данном слое входными данными являются обработанные изображения (data). Количество нейронов в слое определяется автоматичестки (по количеству выходов в предыдушем слое), а количество выходных нейронов указывается с помощью параметра num_output. Результат работы слоя положим по тому же имени, что и имя слоя (ip).

```
layers {
  name: "ip"
```

```
  type: INNER_PRODUCT
  bottom: "data"
  top: "ip"
  inner_product_param {
    num_output: 10
  }
}
```

4. В конце добавим слой, вычисляющий функцию ошибки. Он принимает на вход результат предыдущего полносвязного слоя (ip) и номера классов для каждого изображения (label). После вычислений к результатам работы данного слоя можно обратиться по имени loss.

```
layers {
  name: "loss"
  type: SOFTMAX_LOSS
  bottom: "ip"
  bottom: "label"
  top: "loss"
}
```

Конфигурация сети готова. Далее необходимо определить параметры процедуры обучения в файле формата prototxt (назовем его solver.prototxt). К числу параметров обучения относятся путь к файлу с конфигурацией сети (net), периодичность тестирования во время обучения (test_interval), параметры стохастического градиентного спуска (base_lr, weight_decay и другие), максимальное количество итераций (max_iter), архитектура, на которой будут проводиться вычисления (solver_mode), путь для сохранения обученной сети (snapshot_prefix).

```
net: "examples/mnist/linear_regression.prototxt"
test_iter: 100
test_interval: 500
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
lr_policy: "inv"
gamma: 0.0001
power: 0.75
display: 100
max_iter: 10000
snapshot: 5000
snapshot_prefix: "examples/mnist/linear_regression"
solver_mode: GPU
```

Обучение выполняется с использованием основного приложения библиотеки. При этом передается определенный набор ключей, в частности, название файла, содержащего описание параметров процедуры обучения.

```
caffe train --solver=solver.prototxt
```

После обучения полученную модель можно использовать для классификации изображений, например, с помощью оберток на Python:

1. Подключаем библиотеку Caffe. Устанавливаем режим тестирования и указываем архитектуру для выполнения вычислений (CPU или GPU).

```
import caffe
caffe.set_phase_test()
caffe.set_mode_cpu()
```

2. Создаем нейронную сеть, указывая следующие параметры: MODEL_FILE — конфигурация сети в формате prototxt, PRETRAINED — обученная сеть в формате caffemodel, IMAGE_MEAN — среднее изображение (вычисляется по набору входных изображений и используется для последующей нормализации интенсивности), channel_swap задает цветовую модель, raw_scale — максимальное значение интенсивности, image_dims — разрешение изображения. После чего, загружаем изображение для классификации (IMAGE_FILE).

```
net = caffe.Classifier(MODEL_FILE, PRETRAINED, IMAGE_MEAN,
channel_swap=(0,1,2), raw_scale=255, image_dims=(28, 28))
input_image = caffe.io.load_image(IMAGE_FILE)
```

Получаем ответ нейросети для выбранного изображения и выводим результаты на экран.

```
prediction = net.predict([input_image])
print 'prediction shape:', prediction[0].shape
print 'predicted class:', prediction[0].argmax()
```

Таким образом, путем несложных действий можно получить первые результаты экспериментов с глубокими нейросетевыми моделями.

Более сложные и подробные примеры можно увидеть на сайте разработчиков.

Для работы с Caffe могут использоваться дополнительные программы, такие, как библиотеки Pylearn2, Библиотека Torch, Библиотека Theano.

Сравнение библиотек

Литература.

Deep Learning Tutorial
LISA Deep Learning Tutorial
Understanding Neural Networks from a Programmer’s Perspective от Andrej Karpathy (Stanford)
Neural Networks and Deep Learning
Tutorial on Deep Learning for Vision
http://karpathy.github.io/neuralnets/
DIY Deep Learning for Vision with Caffe
Tutorial Documentation
Installation instructions
API Documentation
caffe.berkeleyvision.doc

Examples

ImageNet tutorial
Train and test "CaffeNet" on ImageNet data.
LeNet MNIST Tutorial
Train and test "LeNet" on the MNIST handwritten digit data.
CIFAR-10 tutorial
Train and test Caffe on CIFAR-10 data.
Fine-tuning for style recognition
Fine-tune the ImageNet-trained CaffeNet on the "Flickr Style" dataset.
CaffeNet C++ Classification example
A simple example performing image classification using the low-level C++ API.
Feature extraction with Caffe C++ code.
Extract CaffeNet / AlexNet features using the Caffe utility.
Web demo
Image classification demo running as a Flask web server.
Siamese Network Tutorial
Train and test a siamese network on MNIST data.
Image Classification and Filter Visualization
Instant recognition with a pre-trained model and a tour of the net interface for visualizing features and parameters layer-by-layer.
Learning LeNet
Define, train, and test the classic LeNet with the Python interface.
Off-the-shelf SGD for classification
Use Caffe as a generic SGD optimizer to train logistic regression on non-image HDF5 data.
Fine-tuning for Style Recognition
Fine-tune the ImageNet-trained CaffeNet on new data.
Editing model parameters
How to do net surgery and manually change model parameters for custom use.
R-CNN detection
Run a pretrained model as a detector in Python.
Siamese network embedding
Extracting features and plotting the Siamese network embedding.
http://caffe.berkeleyvision.org/tutorial/net_layer_blob.html
http://caffe.berkeleyvision.org/
http://courses.cs.tau.ac.il/Caffe_workshop/Bootcamp
npo Caffe
View On GitHub
https://github.com/BVLC/caffe/
http://nbviewer.jupyter.org/github/BVLC/caffe/blob/master/examples/00-classification.ipynb

Сетевое электронное издание учебного пособия.

Кириченко А.А.
профессор Департамента программной инженерии Факультета компьютерных наук Федерального государственного автономного образовательного учреждения высшего профессионального образования "Национальный исследовательский университет "Высшая школа экономики" при правительстве РФ”.

«Нейропакеты. Лекции»

ISBN 978-5-9904911-5-1