АЛЕКСАНДР КИРИЧЕНКО

НЕЙРОСЕТЕВОЕ ПРОГРАММИРОВАНИЕ

ИНСТРУМЕНТАРИЙ НЕЙРОКОМПЬЮТИНГА

Александр Кириченко

Нейросетевое программирование. Инструментарий нейрокомпьютинга

http://www.litres.ru/pages/biblio_book/?art=62874712
ISBN 9785005163271

Аннотация

Данная книга в основном посвящена четвёртому уровню моделирования мозга (создание комплексов, содержащих большое количество совместно работающих нейронных сетей различного назначения, которые оформляются в виде нейросетевых моделей, систем управления, нейроконструкций, гибридных нейронных сетей и т. д. вплоть до нейрокомпьютеров) и представляет интерес учащимся в магистратуре, аспирантами лицам, углублённо специализирующимся на нейросетевых технологиях.

Содержание

Нейросетевое программирование Инструментарий нейрокомпьютинга

Александр Кириченко

ISBN 978-5-0051-6327-1
Создано в интеллектуальной издательской системе Ridero

Нейроконструкции и модели управления

Структурный подход к моделированию мозга реализуется на нескольких уровнях (этапах). В основах теории искусственных нейронных сетей [3] рассматриваются 4 уровня нейросетевого моделирования:

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.

3. Третий уровень нейронного моделирования составляют искусственные нейронные сети (ИНС). Это нейросетевые конструкции, которые с помощью специальной процедуры обучения могут гибко изменять свои параметры.

4. На четвёртом уровне создаются комплексы, содержащие большое количество совместно работающих программ и нейронных сетей различного назначения, которые оформляются в виде нейросетевых моделей, систем управления, нейроконструкций, и т. д. вплоть до нейрокомпьютеров.

Данная книга в основном посвящена четвёртому уровню

моделирования мозга и представляет интерес магистрам, аспирантам и лицам, углублённо специализирующимся на нейросетевых технологиях.

В начале 2000 годов сформировался 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» 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.

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.

С 2012 года в России началось активное проведение IT-исследований в сфере разработки искусственных когнитивных систем, разработана стратегическая программа создания центра прорывных исследований в области информационных технологий «Искусственные когнитивные системы».

Повышение интереса к тематике искусственного интеллекта приводит к появлению большого количества публикаций о структуре и возможностях нейросистем, о типах искусственных нейросетей и открываемых ими возможностях автоматизации мыслительных процессов. Для удовлетворения возникающих потребностей необходимы с одной стороны – новые информационные материалы, и с другой стороны – программные средства, которые позволяют без особых затрат проверять новую информацию на практике, создавать свои нейросетевые системы разных типов, модели нейросетевых устройств и даже узлов нейрокомпьютеров.

Доступные программные средства можно получить из Интернет. Наиболее подготовлен к такой работе freeware нейроконструктор MemBrain [1—4].

Искусственные нейросети являются электронными моделями нейронной структуры мозга.

Продолжительный период эволюции придал мозгу человека много качеств, отсутствующих в современных компьютерах с архитектурой фон Неймана. К ним относятся:

· способность к обучению и обобщению
· ассоциативность и адаптивность

· толерантность к ошибкам
· параллельность работы

Множество проблем, не поддающиеся решению традиционными компьютерами, могут быть эффективно решены с помощью нейросетей.

Достижения в области нейрофизиологии дают начальное понимание механизма естественного мышления, в котором хранение информации происходит в виде сложных образов. Процессы хранения информации в виде образов, использования образов при решении поставленных проблем определяют новую область в обработке данных, которая, не используя традиционного программирования, обеспечивает создание нейронных сетей и их обучение.

В лексиконе разработчиков и пользователей нейросетей появились слова, отличные от традиционной обработки данных, в частности, «вести себя», «реагировать», «самоорганизовывать», «обучать», «обобщать» и «забывать». Такие слова характерны для интеллектуальных систем.

Наиболее устоявшимся является мнение, что интеллект тесно связан с представлением и использованием знаний, машинным творчеством, и затрагивает такие направления, как инженерия знаний, представление знаний, роботы, искусственные нейронные сети, машинное обучение, глубокое обучение, нейронный процессор.

Направление «инженерия знаний» объединяет задачи получения знаний из простой информации, их систематизации

и использования. Это направление исторически связано с созданием экспертных систем – программ, использующих специализированные базы знаний для получения достоверных заключений по какой-либо проблеме.

Производство знаний из данных – одна из базовых проблем интеллектуального анализа данных. Существуют различные подходы к решению этой проблемы, в том числе – на основе нейросетевой технологии, использующие процедуры вербализации нейронных сетей.

К области машинного обучения относится большой класс задач на распознавание образов. Например, это распознавание символов, рукописного текста, речи, анализ текстов.

Нейронные сети используются для решения нечётких и сложных проблем, таких как распознавание или кластеризация объектов.

Природа человеческого творчества ещё менее изучена, чем природа интеллекта. Тем не менее, эта область существует, и в ней поставлены проблемы написания компьютером музыки, литературных произведений (например, стихов или вариаций на темы сказок), художественное творчество. Создание реалистичных образов широко используется в кино и индустрии игр.

Отдельно выделяется изучение проблем технического творчества систем искусственного интеллекта. Теория решения изобретательских задач, предложенная в 1946 году российским изобретателем Г. С. Альтшуллером, положила на-

чало таким исследованиям [16].

В процессе работы над искусственным интеллектом (ИИ) появились новые виды информации, алгоритмы работы с ними, новые методы получения и обработки данных.

Информация может быть представлена в виде данных, знаний, правил и закономерностей, способов получения (добычи), способов хранения и использования. Обращено внимание на смысл, содержащийся в информации, на его поиск, хранение, получение, измерение, преобразование. Понимание смысла связано с выполнением умозаключений, с использованием интеллектуальных навыков, например, таких, как умение делать традуктивные, индуктивные, дедуктивные выводы.

По мере развития ИИ появились новые виды интеллектуальных изделий, в основном – это службы техподдержки различных компаний, экспертные системы по подбору товаров (подарков), по оказанию интеллектуальных услуг клиентам, автоматизированные онлайн-помощники, которые иногда реализованы как чат-боты на веб-страницах, в виде различных интеллектуальных изделий.

На четвёртом уровне моделирования нейроконструкций создаются комплексы, содержащие большое количество нейронных сетей различного назначения и оформляются в виде нейросетевых моделей, систем управления, вплоть до нейрокомпьютеров.

К ним относятся:

– Экспериментальная хаотическая нейросеть:

– Долгосрочная память «Long short term memory (LSTM)»

– Простые комплексы подсетей, такие, как автоэнкодер:

– Комплекс подсетей для образования Глубоких нейросетей и Глубокого обучения.

– Комплекс подсетей для реализации свёрточных сетей

В демонстрационных примерах нейропакета MemBrain рассматриваются характерные их особенности.

Есть разница между стандартным нейросетевым исследованием, для реализации которого необходима программная система типа нейропакет, и нейроконструированием, для реализации которого необходим особый инструмент – нейроконструктор.

Нейропакет должен уметь выполнять такие операции, как создать нейросеть, загрузить в неё набор данных, обучить обработке этого набора, вывести в файл результаты [4].

Нейроконструктор кроме этого должен иметь возможность разобраться в структуре созданной нейросети, прочитать весовые коэффициенты нейронов, изменить их, выгрузить по частям нейроконструкцию, выполнить внешнюю программу, и др.

При моделировании систем 4 уровня возникают проблемы, связанные с сохранением обрабатываемой информации, с использованием различных алгоритмов для её обработки, с управлением последовательностью обработки, с трактовкой информации. Для решения таких проблем необхо-

димо иметь возможность автоматизировать работу нейросетевых конструкций, хранить обрабатываемые образы, осуществлять их поиск, хранение и преобразование.

Чисто нейросетевые операции для этого неприменимы. Иногда их алгоритмы просто неизвестны (ассоциации, интуиция, смысловой поиск, ассоциативное мышление). Нейроконструктор, как инструмент, автоматизирующий нейросетевые исследования, должен расширять возможности проведения нейросетевых исследований, позволять активировать различные программы, регулировать время их исполнения, сохранять мгновенные значения элементов управления, сохранять, читать и запускать в работу необходимые программы.

Реализацию этих операций, очень характерных для моделирования нейроконструкций, отразим в разделах: «Инструментарий нейрокомпьютинга» и «Расширение инструментария нейрокомпьютинга».

Инструментарий нейрокомпьютинга

Используемые в MemBrain инструменты представим в виде перевода на русский язык соответствующих частей Help пакета MemBrain:

– Технология создания скрипта, скрипт таймера

– Программная коррекция свойств нейросетей (экспорт нейросети, структура экспортированного csv-файла, система команд нейропакета)

– Управление проведением нейросетевых иследований с помощью «зажигания нейронов»

– Организация интерфейса пользователя в нейросетевой конструкции

– Группирование нейросетей в нейроконструкции

– Свёрточные нейронные сети (Convolutional Neural Networks)

– Вывод обученной нейросети в виде C-кода

– DLL (Dynamic Link Library) нейроконструктора MemBrain

Технология создания скрипта

Большое значение для нейроконструктора имеет возможность автоматизировать процесс конструирования. В MemBrain для этого предусмотрена возможность использования скриптового языка. Скриптовый язык позволяет записывать в виде текста (стилизованных английских предложений) последовательность команд нейропакета. Эти же команды можно вводить вручную, используя главное меню пакета.

Например, при подготовке нейропакета к работе необходимо произвести его настройку, которая состоит из нескольких часто повторяющихся команд (рассмотрим содержимое файла «Decoder4To16_.as»):

```
// Отрегулируйте меню вид (view)
ViewSetting (BLACK_BG, true);
ViewSetting (SHOW_GRID, false);
ViewSetting (UPDATE_TEACH, true);
ViewSetting (UPDATE_THINK, true);
ViewSetting (SHOW_FIRE, false);
ViewSetting (SHOW_ACT_SPIKES, false);
ViewSetting (SHOW_LINKS, true);
```

/*Такая настройка предусматривает вывод на экран всех чертежей на чёрном фоне. Замена в первой строке true на false позволяет значительно повысить восприимчивость

графики.

Во второй строке запрещается использовать в чертежах сетку.

В третьй и четвёртой строках разрешается использовать коррекцию в процессе обучения и исполнения нейросетей.

Следующие две строки запрещают использование таких конструкций, как FIRE и SPIKES.

Последняя строка разрешает высвечивать связи на чертежах. Результат такой настройки можно посмотреть в меню View:

*/

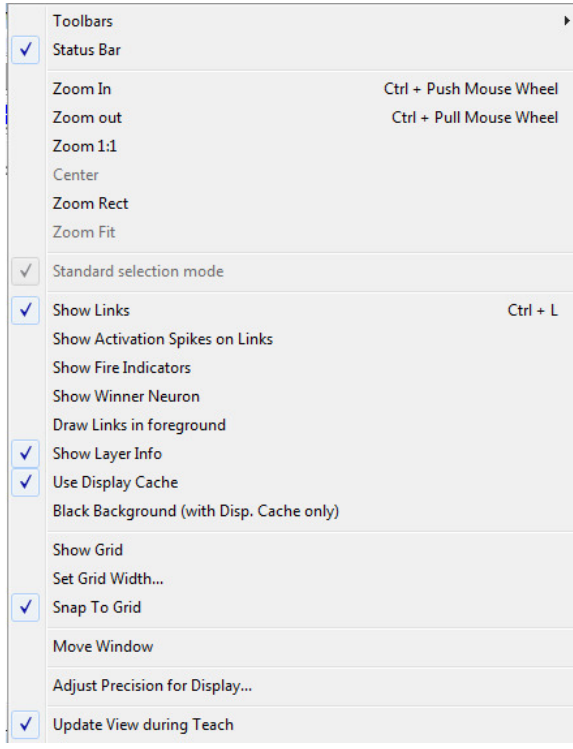

Рис.1 Настройки в меню View

//Открыть на экране нейросеть позволяет команда:

OpenNet («Decoder4To16.mbn»);

//в команде указывается название файла, в котором была сохранена нейросеть.

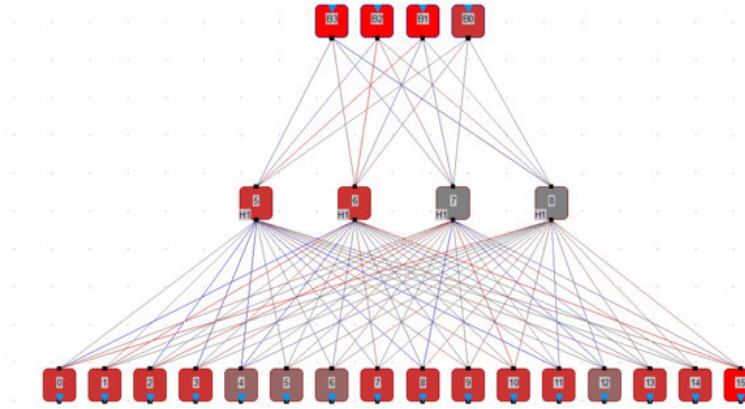

Рис.2 Нейросеть из файла «Decoder4To16.mbn»

//После того, как сеть будет прорисована на экране, необходимо произвести

//рандомизацию всех весов ссылок и порогов активации:

RandomizeNet (); // Или на графике:

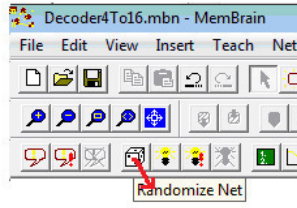

Рис.3 Клавиша рандомизации нейросети

//

//Следующие команды продолжают подготовку нейропакета:

ResetThinkSteps (); // Сброс счетчика шагов (уроков), или в окне:

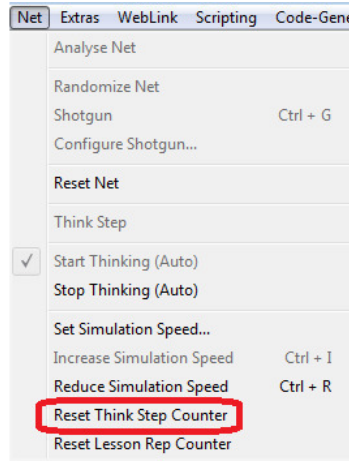

Рис.4 Сброс счетчика шагов (уроков)

//

SetLessonCount (1); // Установить количество уроков (1)

SetLessonCount (4); // в результате чего появится:

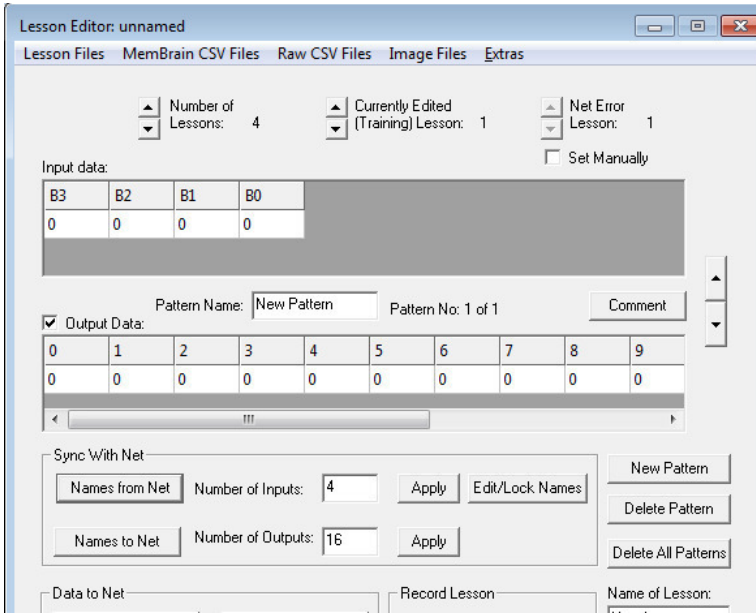

Рис.5 Установка «Number of Lessons»

// Здесь указано, что ожидается 4 урока
/*В данном примере предусмотрено 4 урока:

– training lesson (исходный обучающий урок) – исходные файлы mbn & training lesson. mbl

– validate lesson (исходный контролирующий урок) – исходный файл validate lesson. mbl

– активация выходных нейронов (реакция сети на тренировочные данные 1 к уроку №3) – вывод в файл TrainResult. csv

– результаты урока 2 на данные validate lesson (2 к уроку №3) – вывод в файл ValidateResult. csv

их адреса высвечиваются на экране после входа в Lesson Editor —> Lesson Files —> Урок №1. В скрипт – программе это записывается в виде крманды:

*/

SelectLesson (1); //начинается с меню Lesson Editor:

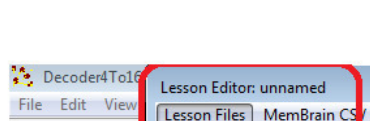

Рис.6 Загрузка урока 1

//

//Выполняется загрузка необходимых для работы данных, настройка учителя и обучение:

LoadLesson («Decoder4To16_Training. mbl»); // Загрузите обучающий урок

//Чтобы полностью просмотреть сеть (последней командой она была закрыта редактором уроков), уберите с экрана редактор урока, чтобы полностью стала видна сеть.

ShowLessonEditor (false);

//Установите учителя, который будет использоваться. Используйте имя учителя, который вы хотите установить в качестве активного.

SelectTeacher («RPROP»); // Используйте учителя с именем «RPROP»

//Настройте учителя (Adjust the teacher) – настройку можно произвести через окно //Teacher Manager:

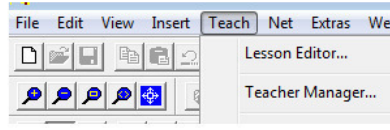

Рис.7 Настройка учителя

//

//Или с помощью команл:

TeacherSetting (LEARNRATE, 0.1); // Скорость обучения

TeacherSetting (TARGET_ERR, 0.3); // Установка допустимой целевой ошибки

TeacherSetting (LESSON_REPS, 1); // Повторение урока за шаг обучения

```
TeacherSetting (PATTERN_REPS, 1); // Повторение паттернов за шаг обучения
//Определите метод выбора шаблона:
TeacherSetting (PATTERN_SELECT, RAND_ORDER);
SetTeachSpeed (0); // Установка максимальной скорости обучения
//и так далее
//Запустите просмотрщик ошибок
ShowErrorViewer (true);
//и уберите его с экрана
ResetErrorViewer ();
//Начните обучение и дождитесь остановки после достижения целевой ошибки
StartTeaching ();
SleepExec (); // Ожидание окончания обучения
//Снова скрыть средство просмотра ошибок
ShowErrorViewer (false);
//Сбросить счетчик шагов (уроков)
ResetThinkSteps ();

//Теперь подготовьте урок №3, чтобы записать результаты работы сети при выполнении урока №1.

//Сохраните значения активации выходных нейронов
SetRecordingType (RT_ACT);
//Включите запись данных на занятие №3
```

```
StartRecording (3);
//Скрыть редактор урока
ShowLessonEditor (false);
//Установка времени для записи целого урока во внутреннюю память MemBrainSetThinkSpeed (500); //500 мс между этапами обдумывания для анимации

//Запись реакции сети на тренировочные данные к уроку №3
ThinkLesson (); // Выполните каждый шаблон урока №1 (и запишите на №3)
SleepExec (); // Ожидание окончания выполнения урока
//Отключение записи после окончания урока (500 мс)
StopRecording ();
SelectLesson (2); // Загрузка урока №2 с данными для проверки
LoadLesson («Decoder4To16_Validate. mbl»);
//Этот урок только для ввода. Не нужны никакие выходные данные,
EnableLessonOutData (false);
//Запишите значения активации выходных нейронов
SetRecordingType (RT_ACT);

//Включить запись данных на урок №4
StartRecording (4);
ShowLessonEditor (false); // Скрыть редактор урока
```

ThinkLesson (); // Выполните каждый шаблон урока №2 (и запишите на №4)

SleepExec (); // Ожидание окончания обучения («Think on Lesson)

StopRecording (); // Отключить запись

//Теперь экспортируйте уроки №3 и 4 в CSV

SelectLesson (3);

ExportLessonRaw («Decoder4To16_TrainResult. csv»);

SelectLesson (4);

ExportLessonRaw («Decoder4To16_ValidateResult. csv»);

/*Раскомментируйте следующую строку, если вы хотите, чтобы MemBrain завершал работу после выполнения скрипта*/

// ExitMemBrain ();

/*Описанная здесь последовательность команд может быть представлена в виде скриптового файла с расширением. as – файла ("MemBrainScript1.as».

Для запуска скрипта в основном меню пакета набрать Scripting -> Execute Script:

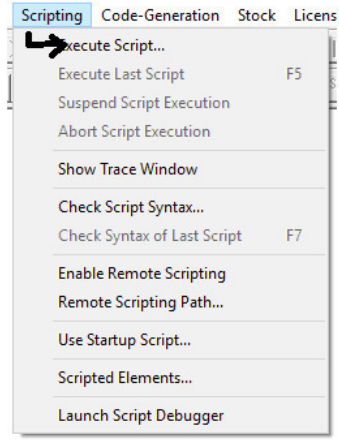

Рис.8 Запуск скрипта

*/

В открывшемся окне выделить название скрипта:

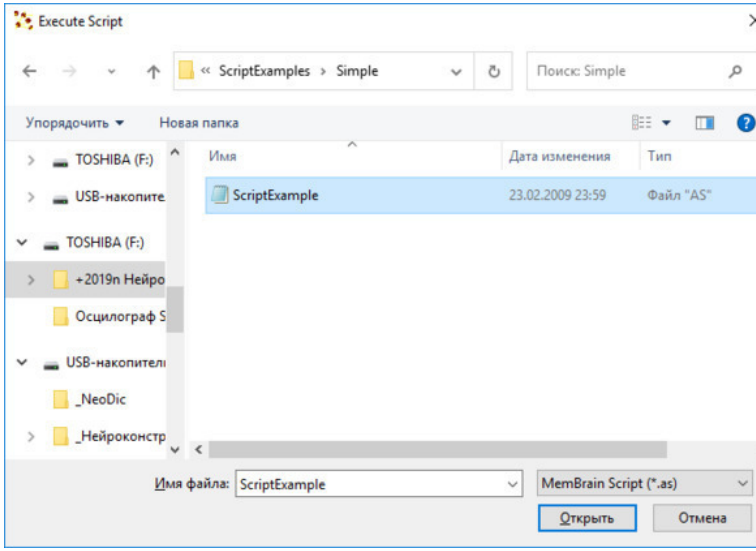

Рис.9 Имя скрипта

После выполнения скрипта полученные результаты выведены на экран в том порядке, как они были указаны в скрипте:

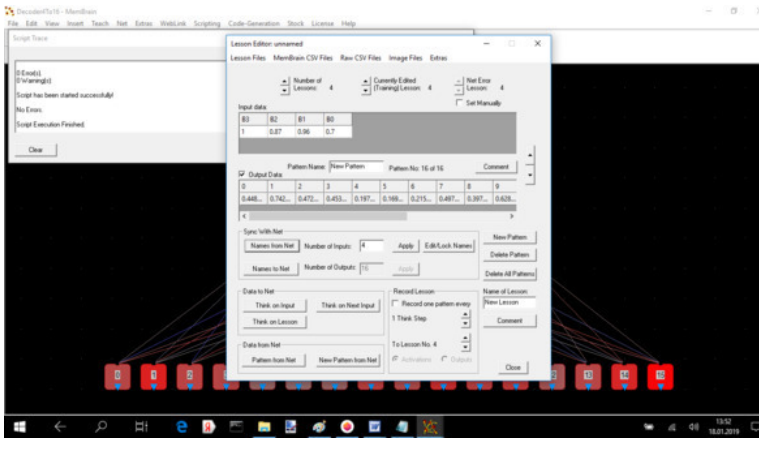

Рис.10 Результат выполнения скрипта

Результатов много. Для их анализа необходимо рассмотреть содержание полученных файлов.

После загрузки сети первым вводился файл Decoder4To16_Training. mbl. Содержимое этого файла можно вывести из MemBrain в файл с расширением csv для дальнейшего просмотра на экране или для вывода на принтер. Номер выводимого из MemBrain файла нужно набрать клавишами Currently Edited (Training) Lesson. После этого активировать Raw CSV Files -> Export Current Lesson (Raw CSV…). В окне Export Lesson As (Raw CSV) … набрать имя создаваемого csv файла и щёлкнуть по клавише «сохранить».

Необходимая для работы скрипта информация содержит-

ся в файлах:

Decoder4To16_Training. mbl

Decoder4To16_TrainResult. csv

Decoder4To16_Validate. mbl

Decoder4To16_ValidateResult. csv

Файлы с расширением mbl содержат двоичный код вводимых данных. Их ввод осуществляется через Lesson Editor:

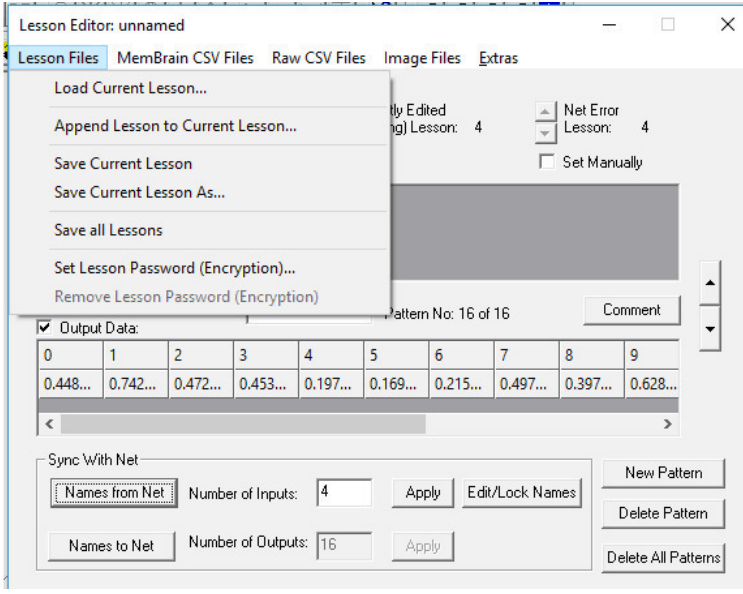

Рис.11 Загрузка mbl-файла

Lesson Editor -> Lesson Files -> Load Current Lesson.

Скриптовый файл может не содержать пояснений – самое важное – это команды. Редкие пояснения даются в виде комментариев.

Для примера: скрипт таймера SecondsTimer.as

/* – — – — – — – — – — – — – — – — – — – — – — – — – — –
— —

Этот пример демонстрирует, как функция сценария Now () может использоваться для поддержки произвольного числа таймеров с разрешением 1 с.

Чтобы использовать таймеры в вашем собственном скрипте, просто включите этот файл в ваш скрипт, используя директиву #include.

Затем создайте экземпляр класса «SecondsTimer' для каждого необходимого вам таймера. Обратите внимание, что функции таймера содержат отладочный код, который печатает сообщения в окне трассировки MemBrain.

Вы можете удалить соответствующий код, если не хотите, чтобы какие-либо сообщения печатались функциями таймера.

– — – — – — – — – — – — – — – — – — – — – — – — – — – —
– — – —

*/

```
/* Глобальная переменная для подсчета экземпляров класса. Используется только для генерации отладочных сообщений.
AngelScript пока не поддерживает статические члены класса. Вот почему это должна быть глобальная переменная.
*/
uint gSecondsTimerInstanceCount = 0;

class SecondsTimer
{
// Constructor without arguments
SecondsTimer ()
{
mId = gSecondsTimerInstanceCount++;
mReferenceTime = Now ();
}

// Constructor with initial elapse time [s] (начальное время истечения)
SecondsTimer (uint seconds)
{
mId = gSecondsTimerInstanceCount++;
// Получить текущее время, добавить прошедшие секунды и сохранить полученное время.
int now = Now ();
```

```
    mReferenceTime = now + seconds;

    // Только для целей отладки:
    /*
    string timeStr;
    TimeToString (now, timeStr);
    Trace (timeStr);
    TimeToString (mReferenceTime, timeStr);
    Trace (» Timer " + mId + " started. Will elapse at:" + timeStr
+ "\n»);
    строка timeStr;
    TimeToString (now (сейчас), timeStr);
    Trace (Трассировка) (timeStr);
    TimeToString (mReferenceTime, timeStr);
    Trace («Таймер» + mId + «запущен. Пройдет по адресу
(закончится):" + timeStr + "\ n»);
    */
    }

    ~SecondsTimer ()
    {
    gSecondsTimerInstanceCount – ;
    }

    // Проверяем, истек ли таймер
    bool IsElapsed ()
```

```
    {
    int now = Now ();
    bool elapsed = now> = mReferenceTime;
    return elapsed;
    }

    // Запускаем таймер по истечении заданного времени
    void Start (uint seconds)
    {
    // Получить текущее время, добавить прошедшие секунды
и сохранить полученное время.
    int now = Now ();
    mReferenceTime = now + seconds;
    }

    // Запускаем таймер для подсчета секунд
    void Start ()
    {
    // Получить текущее время, установить время истечения
на то же значение.
    int now = Now ();
    mReferenceTime = now;
    }

    // Return the elapsed seconds since the timer was started
    int SecondsSinceStart ()
```

```
    {
    // Получить текущее время
    return Now () – mReferenceTime;
    }

    // Контрольное время для этого таймера
    int mReferenceTime;
    // ID таймера для этого таймера (необходим только для
отладочных сообщений)
    uint mId;
    };
```

Экспорт нейросети

Что собой представляет нейросеть?

С внутренним устройством нейросети можно познакомиться, произведя экспорт созданной нейросети (Exporting a Net) в csv-файл.

В MemBrain есть возможность экспортировать сетевой список текущей нейронной сети через файл csv, содержащий значения, разделенные запятыми. Этот файл в дальнейщем можно использовать для импорта нейронной сети, созданной и, обученной с помощью MemBrain, в другое прикладное программное обеспечение.

С помощью следующего диалога можно выбрать уровень детализации, который вы хотите включить в файл экспорта.

Чтобы экспортировать сеть, выберите <File> <Export … >. Появится:

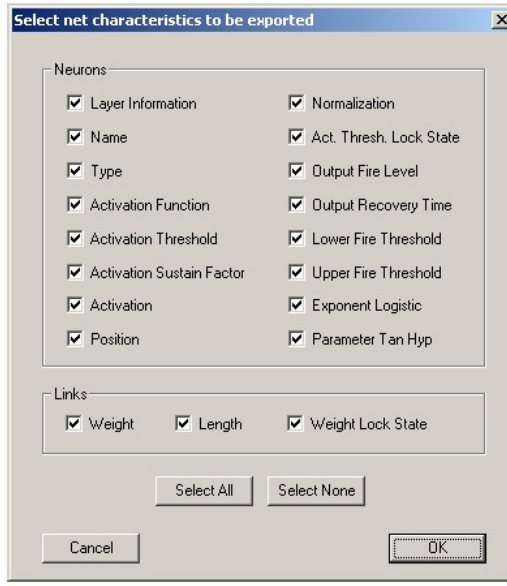

Рис.12 Настройка экспорта нейросети

Этот диалог позволяет выбрать свойства нейронов и ссылок, которые нужно экспортировать в файл. Когда вы сделали свой выбор, нажмите ОК. Вам будет предложено выбрать местоположение и имя файла для сохранения экспортированного файла.

Структура и состав экспортируемого CSV файла (Net CSV File)

Когда вы экспортируете сеть, MemBrain создает для этой цели секционированный CSV-файл. Этот файл состоит из разделов данных, которые идентифицируются несколькими ключевыми словами, каждое из которых включено в начальный и конечный теги '<' и '>' соответственно. Точный формат файла зависит от свойств, которые вы выбрали для экспорта и может выглядеть следующим образом:

/*
MemBrain, Version XX. XX
(<Month> <Day> <Year>)
Секционированный CSV-файл
Стартовые ключевые слова для сети
[<NET START>]
Дополнительная информация о содержимом файла
[<INFO HEADER>]
Этот файл представляет собой нейронную сеть MemBrain.

Информация о формате, используемом для экспорта нейронов – это своего рода «заголовок» для объяснения данных, содержащихся в разделе [<NEURONS <], который следует далее.

[<NEURON FORMAT INFO>]
ID; LAYER; NAME = Идентификатор; Слой; Название

Этот раздел содержит все нейроны в сети. Один нейрон всегда представлен одной строкой в CSV-файле.

[<NEURONS>]

1;I; In1

2;I; In2

3;O; Out

Информация о формате, используемом для экспорта ссылок – это своего рода «заголовок» для объяснения данных, содержащихся в раздел [<LINKS <], который будет следующим.

[<LINK FORMAT INFO>]

SOURCE_ID; TARGET_ID; WEIGHT

Этот раздел содержит все ссылки в сети.

Одна ссылка всегда представлена одной строкой в CSV-файле.

[<LINKS>]

1;3;0,263535

2;3;0,178995

Конец ключевых слов.

[<END>]

*/

Покажем на реальном примере:

Пример.

Создадим нейросеть для анализа принадлежности точки одному из двух множеств: синему или жёлтому.

Для обучения покажем нейросети смешанный файл с точками circle1.csv (374 записей):

csv-файл

In1	In2	Out
-0,97825	0,4355	1
0,4625	-0,70475	1
0,95125	-0,79225	1
-0,507	-0,023	2
0,22975	-0,82	1
-0,623	-0,839	1
0,41775	0,6025	2
-0,117	0,3185	2
-0,1285	-0,16225	2
-0,1855	0,588	2
0,6895	0,69975	1
...		

Файл circle1.csv

Для контроля хода обучения создадим файл с точками, которые не показываются нейросети при обучении:

In1	In2	Out6
0,4315	-0,887	1
0,6465	0,9965	1
0,2355	0,53825	2
0,14625	0,75875	2
0,99425	0,10825	1
0,18925	0,6705	2
0,71075	0,2335	2
0,82575	0,874	1
0,718	-0,1575	2
0,1975	0,623	2
-0,6595	0,41075	2
0,70175	-0,7325	1
0,6165	0,75575	1
0,74575	-0,328	1
0,51075	0,76825	1
0,4775	0,36025	2
0,38725	0,10725	2
0,15025	0,1705	2
0,09675	-0,192	2
0,12325	0,5235	2

Файл circle2.csv (21 запись)

Оба эти файла получены из полного исходного файла «circle. csv», содержащего 394 записи:

In1	In2	Out
0.978250	0.435500	1
0.462500	0.704750	1
0.951250	0.792250	1
0.507000	0.023000	2
0.431500	0.887000	1
0.229750	0.820000	1
0.623000	0.839000	1
0.417750	0.602500	2
0.646500	0.996500	1
...		

Файл «circle. csv»

Загружаем нейросеть из «Circle_1.mbn»

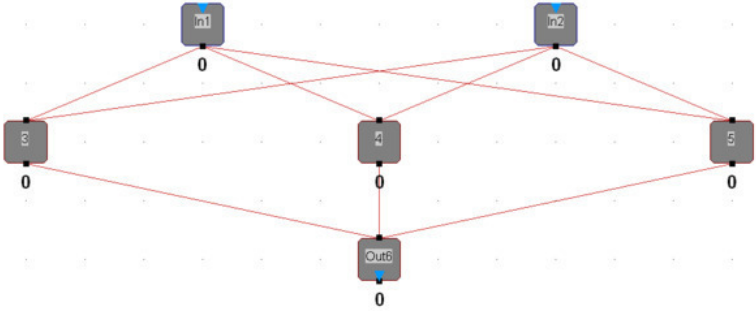

Рис.13 Нейросеть загружена

Загружаем «circle1.csv»

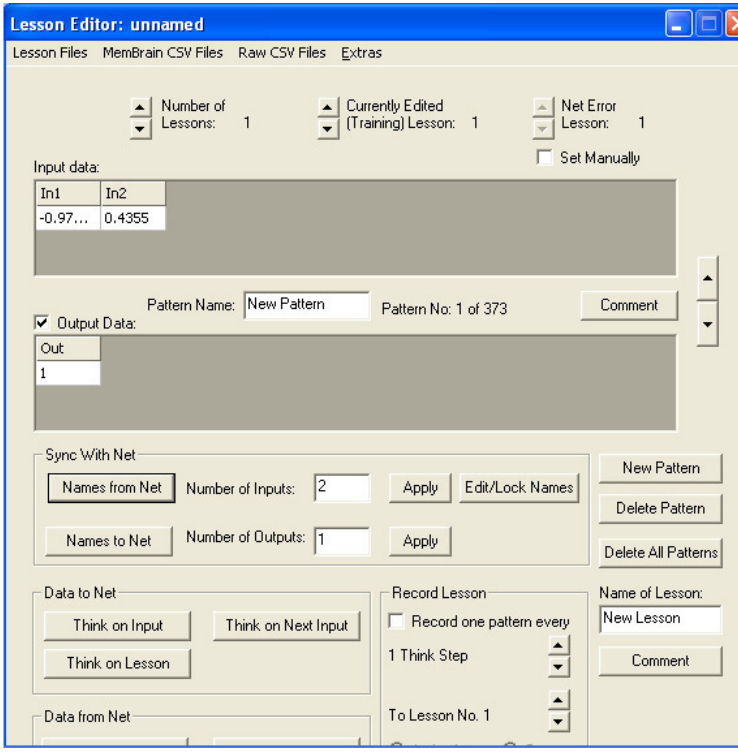

Рис.14 Загружен обучающий файл

Загружаем «circle2.csv»

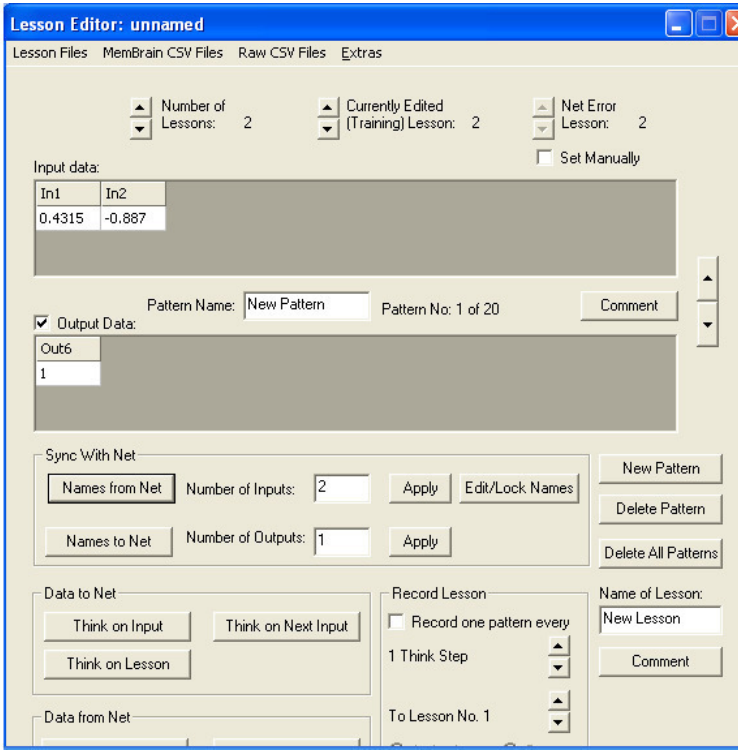

Рис.15 Загружен контролирующий файл

После настройки Lesson Editor для обучения:

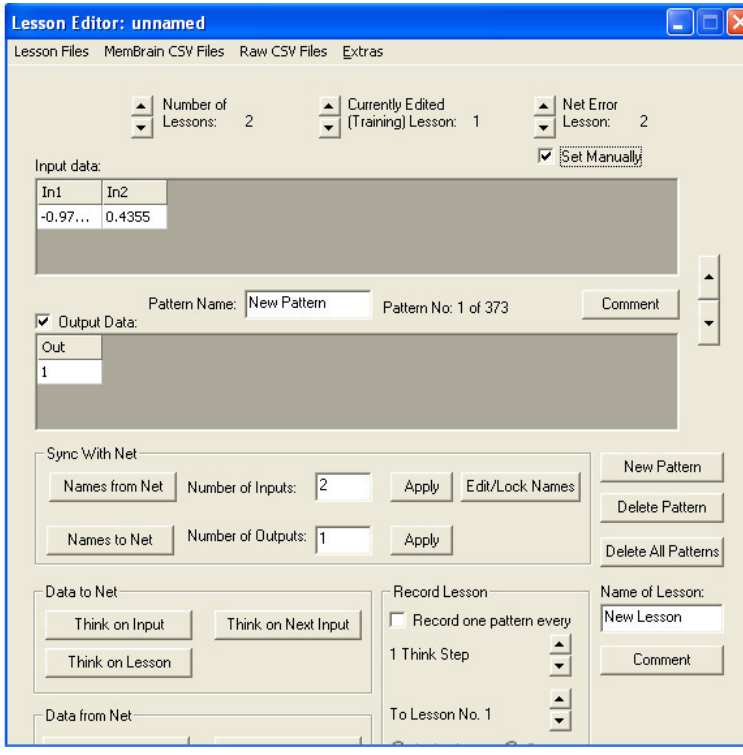

Рис.16 Подготовка обучения нейросети

Согласуем нейросеть с настройкой нейропакетаMemBrain.

После согласования нейросети с Lesson Editor (Names from Net) надо нормализовать Wizard через Extras. Теперь можно обучать нейросеть:

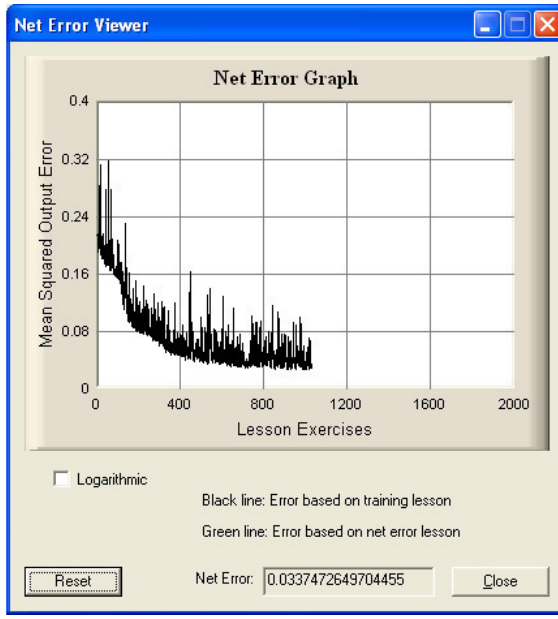

Рис.17 Ход обучения виден в окне Pattern Error Viewer:

Рассмотрим подробнее ход обучения.

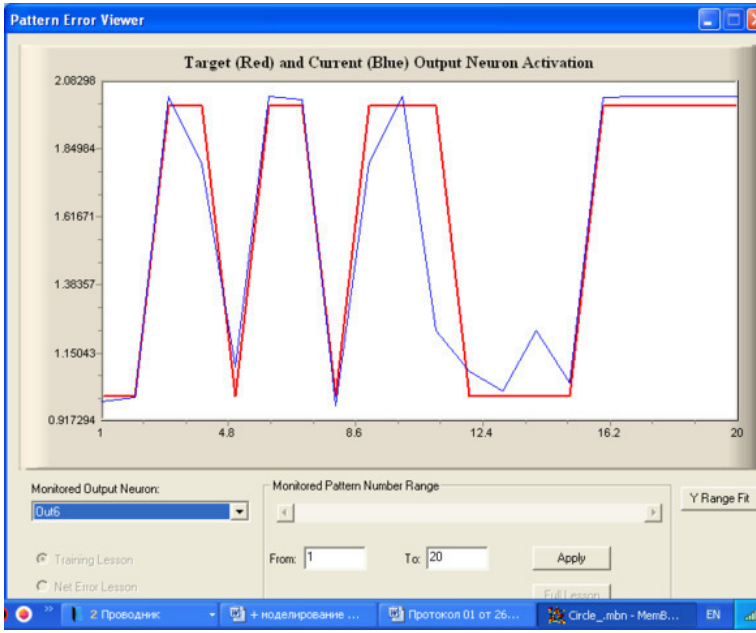

Рис.18 Ход обучения нейросети

Для ознакомления с полученной структурой и составом обученной нейросети экспортируем нейросеть в csv-файл «экспорт 1. csv». Для этого определим состав выводимой информации:

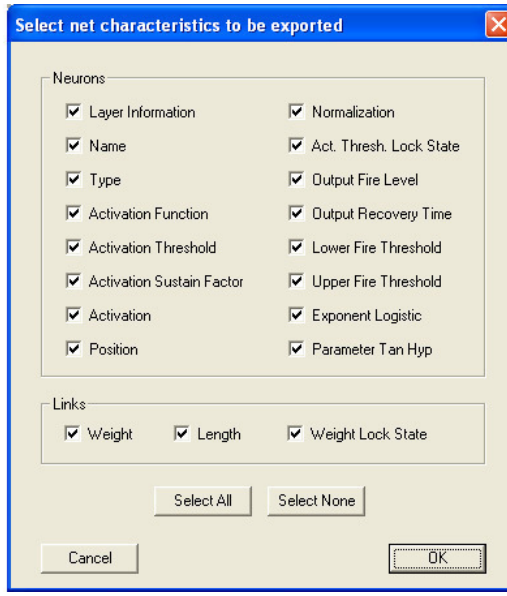

Рис.19 Подготовка экспорта обученной нейросети

Экспорт произведём нажатием кнопки «Export» в меню «File».

Обученная нейросеть будет выведена в файл «Export Circle. csv»:

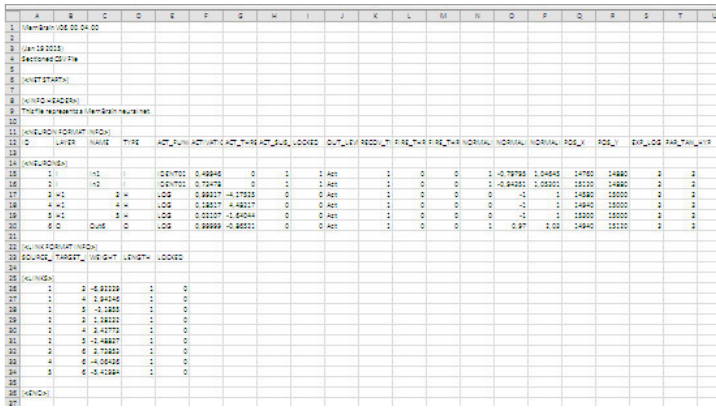

Рис.20 Характеристики нейросети после обучения

Таблица состоит из 3 частей.

1 часть – заголовок.

2 часть – [<NEURON FORMAT INFO>]

В этой части содержится информация о нейронах (№; Название слоя, в котором нейрон находится; Имя; Тип нейрона; Характеристики…):

[<NEURON FORMAT INFO>]							
ID	LAYER	NAME	TYPE	ACT_FUNC	ACTIVATION	ACT_THRES	ACT_S
[<NEURONS>]							
1	I	In1	I	IDENT01	0,499458	0	
2	I	In2	I	IDENT01	0,734785	0	
3	H1	3	H	LOG	0,993175	-4,17525	
4	H1	4	H	LOG	0,185174	4,482169	
5	H1	5	H	LOG	0,02107	-1,64044	
6	O	Out6	O	LOG	0,999985	-0,86521	

Рис.21 Информация о нейронах

3 часть – информация о связях нейронов:

[<LINK FORMAT INFO>]				
SOURCE_ID	TARGET_ID	WEIGHT	LENGTH	LOCKED
[<LINKS>]				
1	3	-6,92229	1	0
1	4	2,942461	1	0
1	5	-2,1855	1	0
2	3	1,282323	1	0
2	4	3,427726	1	0
2	5	-2,48837	1	0
3	6	3,738532	1	0
4	6	-4,06436	1	0
5	6	-5,41984	1	0
[<END>]				

Рис.22 Веса связей нейронов

Подробное знакомство с нейросетью

Подробное знакомство с тем, что представляют собой и что можно делать с нейронными сетями, можно получить по справочнику команд (Command Reference), в котором перечислены все доступные специфичные для MemBrain команды языка сценариев MemBrain в соответствии со следующими подкатегориями:

- MemBrain Application (MemBrain приложение)
- Handling Neural Nets (Обработка нейронных сетей)
- Handling Lesson Data (Обработка данных урока)
- Teaching (Обучение)
- Thinking (мышление)
- Adjusting the View (Регулировка вида)
- Controlling the Weblink (Управление веб-ссылкой)
- Communication with the user (общение с пользователем)
- Controlling External Applications (Управление внешними приложениями)
- Stock Management (Управление запасами)
- Arbitrary file Access (Произвольный доступ к файлам)
- Strings (Строки)
- String Utilities (Строка Утилиты)
- Maths (Математика)
- FFT Calculations (БПФ расчеты)
- Serial Ports (последовательные порты)

Команды языка сценариев MemBrain

В этих разделах можно познакомиться с такими командами, как:

//Открыть файл MemBrain net (*.mbn)

[void OpenNet (постоянная строка & в файле fileName)]

//Сохранить текущую сеть MemBrain в файл (*.mbn)
[void SaveNet (const string &in fileName)]
//Получить имя файла текущей загруженной сети
[void GetCurrentNetFileName (string& out fileName)]
//Экспорт текущей сети MemBrain в CSV-файл
[void ExportNet (const string &in fileName)]
//Получить количество входных нейронов в сети
[uint GetInputCount ()]
//Получить количество выходных нейронов в сети
[uint GetOutputCount ()]
//Получить имя входного нейрона
[bool GetInputName (uint inNeuronNum, string &out name)]
//Получить имя выходного нейрона
[bool GetOutputName (uint outNeuronNum, string &out name)]
//Получить текущую активацию выходного нейрона
[bool GetOutputAct (uint outNeuronNum, double &out activation)]
// Получить свойства выбранного в настоящее время нейрона
bool GetSelectedNeuronProp (SNeuronProp &out prop)
// Установите свойства всех выбранных в настоящее время нейронов
void SetSelectedNeuronProp (SNeuronProp &in prop)
// prop – Структура данных SLinkProp для считывания параметров ссылки:

SLinkProp

- Link Properties Data Structure and

[<LINK FORMAT INFO>]					
SOURCE_ID	TARGET_ID	WEIGHT	LENGTH	LOCKED	
[<LINKS>]					
1	3	-6,92229	1	0	
1	4	2,942461	1	0	
1	5	-2,1855	1	0	
2	3	1,282323	1	0	
2	4	3,427726	1	0	
2	5	-2,48837	1	0	
3	6	3,738532	1	0	
4	6	-4,06436	1	0	
5	6	-5,41984	1	0	
[<END>]					

Default Values -

Property (struct member) Name	Data Type	Valid Range	Default Value
weight	double	-	0.5

Рис.23 Веса нейронов

//По команде ViewSetting (EViewSetting setting, bool on) можно получить подробную информацию о конструктивных особенностях нейросети:

Command	void ViewSetting(EViewSetting setting, bool on)		
Purpose	Установите атрибут представления MemBrain.		
Parameters Must always be a multiple of two: A setting name followed by a corresponding value	Setting Name	Value	Function
	UPDATE_TEACH	true or false	Включает / отключает обновление вида во время обучения
	UPDATE_THINK	true or false	Включает / отключает обновление представления во время анализа
	SHOW_LINKS	true or false	Показывает / скрывает ссылки между нейронами
	SHOW_ACT_SPIKES	true or false	Показывает / скрывает шипы активации в ссылках
	SHOW_Fire	true or false	Показывает / скрывает индикаторы огня (fire) на выходах нейронов
	SHOW_GRID	true or false	Показывает / скрывает сетку
	BLACK_BG	true or false	Включает / отключает черный фон окна
	SHOW_WINNER	true or false	Показывает / скрывает текущий выходной нейрон победителя
Comments	none		

Рис.24 Конструктивные особенности нейросети

Разобравшись в технологии создания скрипта, используемой системе команд, получив необходимую информацию о составе, структуре, устройстве нейросети, можно модернизировать нейросетевую конструкцию, программно корректировать свойства нейросетей

Управление проведением нейросетевых исследований с помощью «Зажигания нейронов»

MemBrain разрешает использовать нейроны для управления выполнением различных действий с нейросетями и различными конструкциями компьютера.

Любой нейрон нейросети может быть объявлен «Управляющим». Управляющий нейрон может быт «зажжён» (Fire). Смысл термина «огонь» (Fire) заключается в том, что выход нейрона (его активация) принимает значение <> 0.

Запуск нейрона разрешается следующими параметрами, которые являются свойствами (Properties) каждого нейрона:

- Нижний порог огня (Threshold)
- Верхний порог огня (Upper Threshold)
- Выбор уровня выходного сигнала («1» или «Активация»)
- Время восстановления выхода (The Output Recovery Time)

Если активация нейрона меньше или равна нижнему порогу огня, то нейрон не будет срабатывать в любом случае.

Решение о том, зажигается ли нейрон (выход <> 0) или нет (выход = 0), принимается на основе вероятности запуска, которая возрастает от 0 до 1 с активацией в диапазоне

от нижнего порога огня к верхнему порогу огня.

Когда нейрон срабатывает, он может дополнительно запустить исполняемый или пакетный файл. Чтобы быть более точным, файл может быть указан Windows для открытия независимо от его типа. Это также может быть документ, который связан с определенным приложением в Windows.

Чтобы указать файл, который должен открываться при зажигании нейрона, выберите один или несколько нейронов, а затем выберите «Дополнительно»> «Исполняемый файл при запуске…» (<Extras> <Executable When Firing…>) в главном меню или в контекстном меню, которое появляется при щелчке правой кнопкой мыши на одном из выбранных нейронов.

Появится следующий диалог:

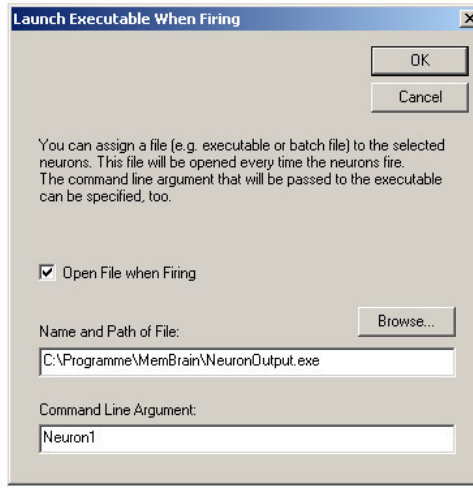

Рис.25 Настройка зажигаемого файла

Обратите внимание на флажок с именем <Открыть файл при стрельбе> (Open File when Firing). Если этот флажок установлен, информация ниже актуальна.

Помните, что нужно установить display neuron fire indicators (индикаторы огня нейрона) для того, чтобы визуализировать момент зажигания нейрона.

Путь и имя файла, который должен быть открыт, а также аргумент командной строки хранятся с каждым нейроном отдельно. Таким образом, вы можете указать разные файлы, которые будут открыты для каждого нейрона. Обратите внимание, что вы можете отображать на экране индикаторы ог-

ня нейронов, чтобы визуализировать их работу.

Если активированы настройки дисплея <Вид> <Показать индикаторы огня> (<View> <Show Links>), то нейроны показывают желтую точку на своем выходном разъеме каждый раз, когда они запускаются (выдают сигнал <> 0). На следующем рисунке показан нейрон, который в данный момент работает. На первом рисунке опция <Вид> <Показать ссылки> деактивирована, поэтому входной разъем не отображается. Второе изображение – это результат отображения, если опция активирована.

Рис.26 Индикация зажигаемого нейрона

Пример: Нейросетевой преобразователь последовательного кода в параллельный

В книге «Конструирование искусственных нейронных ансамблей (ИНА).pdf» [38] был представлен преобразователь

последовательного кода в параллельный, состоявший из трёх групп нейронов: входных, выходных и нейронов задержки. В состав преобразователя был включён нейрон In1, который практически не использовался. Назначение и принцип действия этиго нейрона было обещано рассмотреть при описании скриптов пакета MemBrain.

Основное назначение этого нейрона (In1): управление работой нейроконструкции.

Для демонстрации управляющих функций нейрона восстанавливаем в основном окне MemBrain преобразователь последовательного кода в параллельный с помощью программы «СдРег. mbn» [10]:

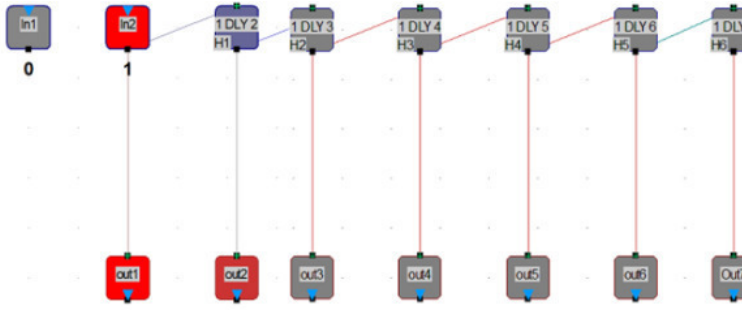

Рис.27 Схема преобразователя последовательного кода в параллельный

Анализ созданной нейросети показывает:

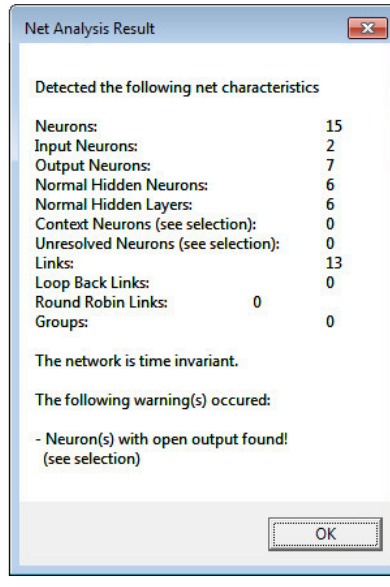

Рис.28 Характеристики нейросети

всего нейронов – 15. Из них: 2 входных, 7 выходных, 6 скрытых нейронов (1 DLY 2 – 1 DLY 7) в 6 скрытых слоях (H1 – H6).

Для демонстрации работы преобразователя кодов было предусмотрено два файла.

В первом файле (SdvReg0.csv) нейрон In1 всегда равен 0. По In2 поступают последовательные сигналы, накапливающиеся в параллельные группы по 7 сигналов.

In1	In2	Out1	Out2	Out3	Out4	Out5	Out6	Out7
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0
0	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1

Рис.29 Основной файл для проверки преобразователя кодов

Используем этот файл для обучения:

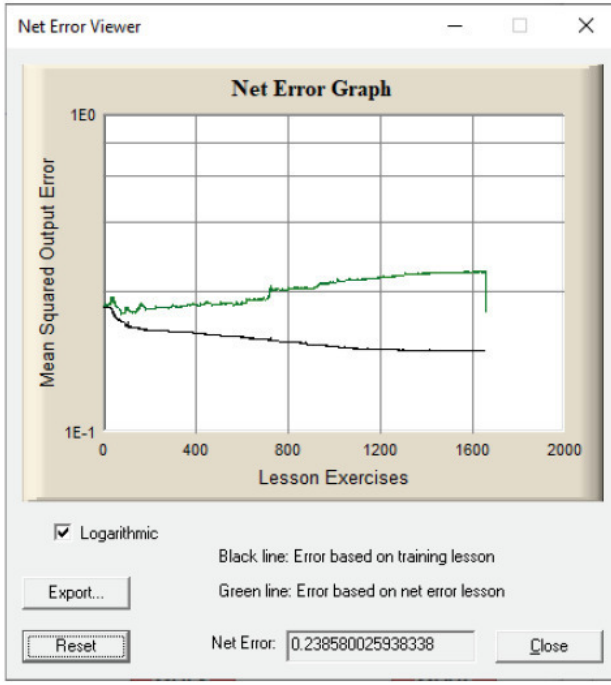

Рис.30 Ход обучения преобразователя кодов

Ошибка обучения имеет достаточно малую величину и показывает, что нейросеть постепенно приблизилась к правильному результату (это видно по голубой линии. на Pattern Error Viewer):

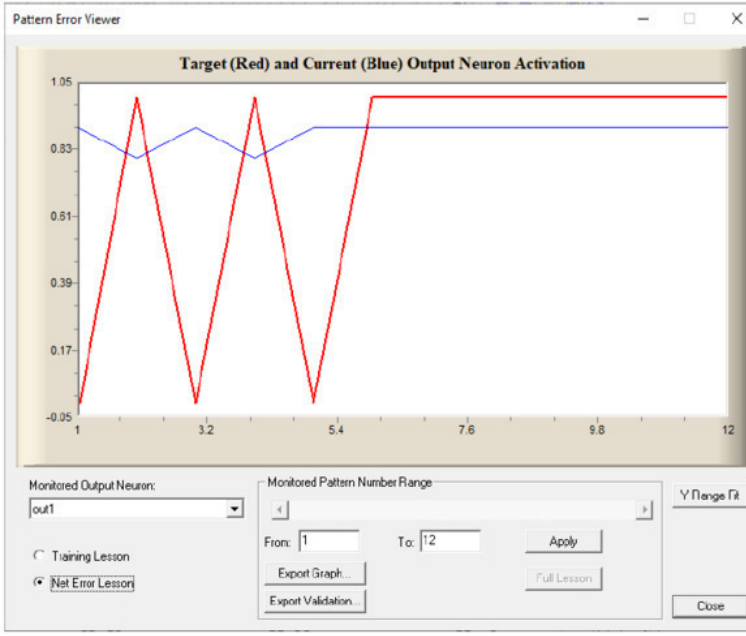

Рис.31 Процесс обучения нейросети

После 6 точки голубая линия совпадает с Target. По клавише Export Graph выводится исполненный пример после обучения:

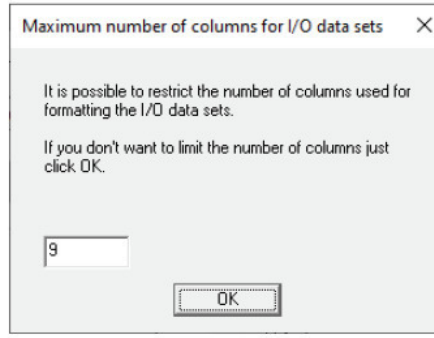

Рис.32 Максимальное количество колонок

Здесь содержится предупреждение о структуре выводимой информации.

Содержимое файла «SdvReg00.csv», полученного по клавише Export Graph:

In1	In2	out1	out2	out3	out4	out5	out6	Out7
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	1	0	1	0	0
0	1	1	1	0	1	0	1	0
0	1	1	1	1	0	1	0	1
0	1	1	1	1	1	0	1	0
0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1

Рис.33 Правильный результат после обучения.

Файл SdvReg2.csv имеет другое строение:

Нейрон In1 может исполнять разные функции: он может сигнализировать момент окончания превращения последовательного кода в параллельный и включать перепись сформированного параллельного кода с выходных нейронов в ячейку памяти для накопления. Если In1 отмечает момент, когда закончено формирование одной строки параллельного кода, то по сигналу In1=1 полученный код нужно считать с выходных нейронов нейросети. При исполнении такого действия нужно будет сформировать и вывести только одно число 1111000 (см. рисунок 34).

Но может быть In1=1 и сигналом, запускающим сохранение в памяти в виде csv-кода исполняемого (текущего) фрагмента обучающей выборки MemBrain. Тогда выводиться должно будет не одно число, а ряд чисел, составляющих текущую обучающую последовательность. Разница – в программах, которые будут работать по зажиганию нейрона In1.

In2 фиксирует символы поступающего последовательного кода.

Для демонстрации процесса зажигания управляющего нейрона подготовлен новый файл: 090720.csv:

In1	In2	out1	out2	out3	out4	out5	out6	Out7
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1

Рис.34 Файл, вызывающий зажигание нейрона In1

После загрузки этого файла выведем на экран одновременно часть нейросети с нейроном In1 и часть редактора уроков Lesson Editor, содержащую нейрон In1. До тех пор, пока In1 = 0, нейрон не зажигается (это видно по выходному порту этого нейрона. Одновременно в редакторе уроков видно, что нейрон In1=0):

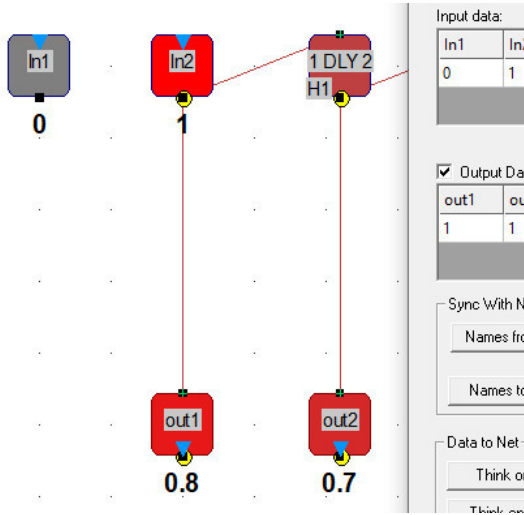

Рис.35 Нейрон In1 не зажжён

Как только In1 стал 1, нейрон зажёгся и начал работать…

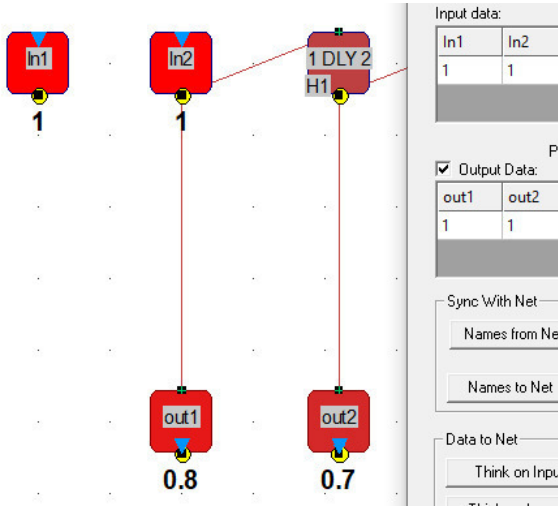

Рис.36 Нейрон In1 зажёгся

В нашем случае по команде из программы «230620.as» выводится весь последний пример, а не одна строка. Нейрон In1 = 1 будем использовать как команду считать в csv-файл содержимое выполненного примера. Считывание это должно производиться по команде ExportLessonRaw раздела LessonEditor, которая записана в скрипте «230620.as».

Программа «230620.as» :

```
void main ()
{
SelectLesson (3);
ExportLessonRaw(C:\ProgramData\MemBrain
```

\”230620-3.as»);
}

Последовательность запуска управляющего нейрона:

– Включить View -> «Show Fire Indicator».

– Средствами Windows записать в ячейку «SdvReg_111.csv» ноль.

Активизировать запуск программы при появлении огня (Fire):

– Пометить нейрон In1 на графической схеме нейросети свойством (Properties) «Executable When Firing» или: через Extras -> Executable when Firing.

– Заполнить диалог:

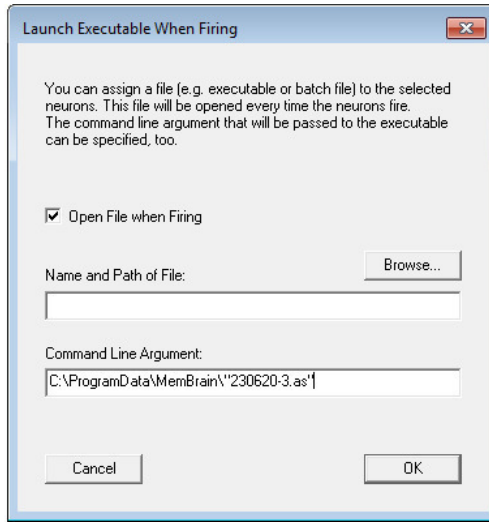

Рис.38 Выполняемая программа при зажигании нейрона

– Нажать «Think on Lesson» на Lesson Editor.

Lesson Editor настроен на запись по скриптовой команде из файла ”230620-3.as” в 3 урок:

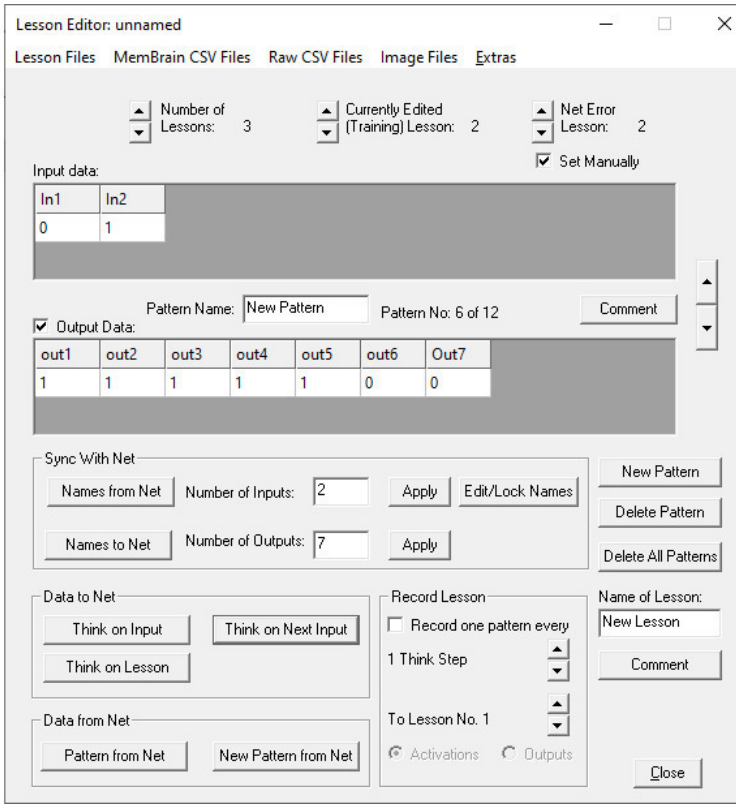

Рис.39 Вывод результата из 3 урока

Выводим из 3 урока информацию в файл SdvReg_111.csv. На экране появляется:

In1	In2	out1	out2	out3	out4	out5	out6	Out7
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1

Рис.40 Выводимая информация

В результате срабатывания управляющего нейрона In1, в файл SdvReg_111.csv выведена информация из 3 урока.

Таким образом:

В демонстрационной папке этого примера должно содержаться:

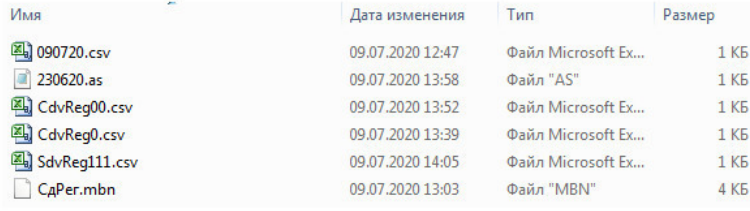

Рис.41 Содержимое папки, необходимое для зажигания нейрона

Интерфейс пользователя

Перевод раздела справочника команд нейропакета «Communication with the user».

Раздел содержит 7 команд:

1. Команда «void MessageBox (const string &in message)» выводит простое окно сообщения с кнопкой ОК. Сообщение содержит текст, который будет отображаться в выводимом окне. Выполнение сценария останавливается до тех пор, пока пользователь не нажмет кнопку ОК, которая автоматически закрывает окно сообщения.

2. Команда «EDlgRet MessageBox (const string &in message, EMsgBoxType type)» выводит окно сообщения с несколькими различными кнопками. Варианты используемых кнопок:

MB_OK – просто кнопка ОК
MB_OKCANCEL – кнопка ОК и ОТМЕНА
MB_YESNO – кнопки ДА и НЕТ
MB_YESNOCANCEL – ДА, НЕТ и кнопка ОТМЕНА
MB_RETRYCANCEL – кнопки RETRY и CANCEL
MB_ABORTRETRYIGNORE – кнопки ABORT, RETRY и IGNOR

Команда возвращает информацию о том, какую кнопку

в окне сообщения нажал пользователь. Выполнение сценария останавливается до тех пор, пока пользователь не нажмет одну из кнопок, которая автоматически закрывает окно сообщения.

3. Команда, содержащая просьбу ввести значение имеет три версии, которые поддерживают типы данных «double», «string» и «int». Все три версии ведут себя одинаково:

EDlgRet UserInput (const string &in explanation, double &in initValue, double &out userValue)

EDlgRet UserInput (const string &in explanation, int &in initValue, int &out userValue),

EDlgRet UserInput (const string &in explanation, const string &in initValue, string &out userValue),

Где InitValue указывает начальное значение в поле ввода данных,

userValue – переменная, которая получила фактический пользовательский ввод.

Команда содержит текст, объясняющий, тип данных, которые должны быть введены и каково их значение. Выполнение сценария останавливается до тех пор, пока пользователь не нажмет одну из кнопок, которая автоматически закрывает диалоговое окно. Главное – это то, что команда возвращает информацию о том, какую кнопку в диалоговом окне пользовательского ввода нажал пользователь: всегда одно из двух значений: IDOK или IDCANCEL

4. Диалоговое окно для выбора файла вызывается коман-

дой

«EDlgRet FileOpenDlg (const string &in title, const string &in extension, const string &in fileNameInit, string &out fileName)».

В title указывается заголовок для отображения в диалоге. Extension определяет расширение по умолчанию, которое будет использоваться для просмотра файлов в диалоговом окне. Например, строка расширения «txt» устанавливает фильтрацию файлов для всех файлов с расширением «txt».

Укажите «*» или просто «» (то есть пустую строку), если вы не хотите устанавливать фильтр файлов. Диалог в этом случае просматривает все типы файлов (*. *).

В fileNameInit можно указать начальное имя файла для отображения в диалоговом окне. Укажите «», если вы не хотите указывать начальное имя файла.

FileName содержит имя файла, полное имя файла (включая путь), которое выбрал пользователь.

Команда возвращает информацию о том, какую кнопку в диалоговом окне пользовательского ввода нажал пользователь: всегда одно из следующего, которое соответствует непосредственно нажатой кнопке: IDOK или IDCANCEL. Выполнение сценария останавливается до тех пор, пока пользователь не нажмет одну из кнопок, которая автоматически закрывает диалоговое окно.

5. Команда «Открыть диалоговое окно выбора файла для

выбора файла для сохранения:

«EDlgRet FileSaveDlg (const string &in title, const string &in extension, const string &in fileNameInit, string &out fileName)».

6. Команда «Показать / скрыть окно трассировки сценариев:

«void ShowTraceWin (bool show)».

При этом, show – «true» показывает окно трассировки, «false» скрывает его. Сценарий может помещать текстовые сообщения в окно трассировки с помощью команды сценария «Трассировка (…)».

7. Вывести текстовую строку в окно трассировки сценариев:

«void Trace (const string &in text)».

> Здесь text – текст для вывода в окно трассировки. Добавьте «\ r \ n» к тексту, если вы хотите, чтобы следующий текст был добавлен к новой строке в окне трассировки. Если окно трассировки никогда не отображалось с момента запуска MemBrain, оно будет показано автоматически при выполнении этой команды. Однако, когда окно трассировки закрыто пользователем, оно не будет отображаться автоматически!

Автоматизация создания нейросети с помощью группы скриптов NetEditor с автоматической прорисовкой схемы нейросети на экране

Для создания сети используются тексты следующих скриптов:

// UsingNetEditorExample.as

// NetEditor.as

// Position.as

Создание сети ведётся под управлением пользователя. После запуска скриптовой программы для связи с пользователем используется интерфейс – пользователю задаётся вопрос о характеристиках создаваемой нейросети:

Введите количество скрытых слоёв в нейросети (Pleas enter the number of hidden layers for the net) и предлагается вариант ответа: 2;

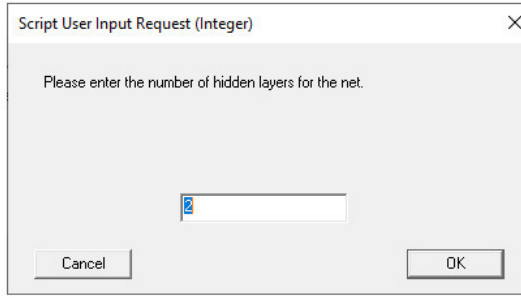

Рис.42 Количество скрытых слоёв

Соглашаемся на 2

Следующим задаётся вопрос: хотите использовать для создания нейросети файл, содержащий готовые уроки MemBrain? (Да, или Нет)

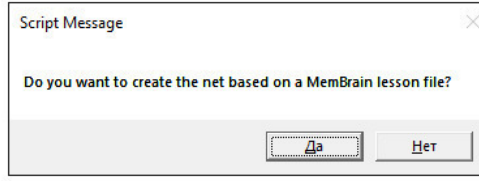

Рис.43 Будет ли использоваться mbl-файл

Да

Если есть готовый файл с уроками (файл в формате mbl), надо указать координаты этого файла. По содержимому файла определяется количество входных и выходных нейронов и начинается создание сети: появляется окно скрипта:

Рис.44 Окно скрипта с извещением о создании нейросети

На экране появляются фрагменты создаваемой нейросети:

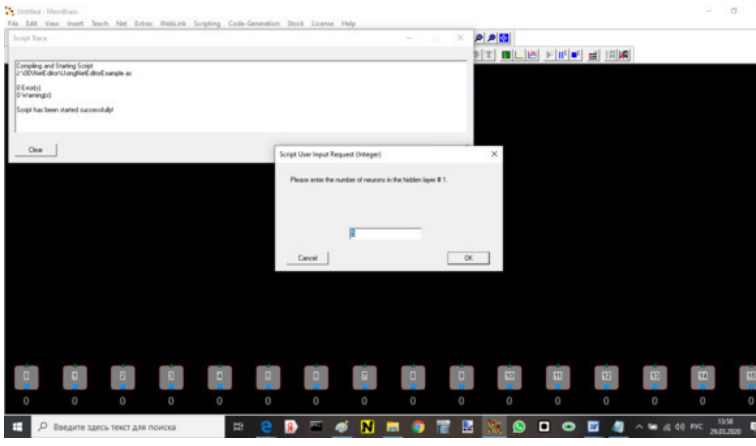

Рис.45 Выводимые на экран сообщения о создании нейросети

И задаётся вопрос, какое количество нейронов должен содержать скрытый слой №1?

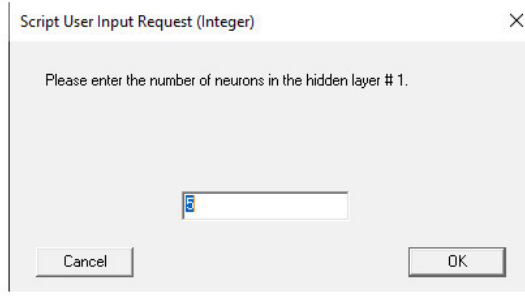

Рис.46 Количество нейронов в скрытом слое 1

Подсказывается вариант ответа (5). При ответе Да задаётся
вопрос о количестве нейронов в скрытом слое 2.

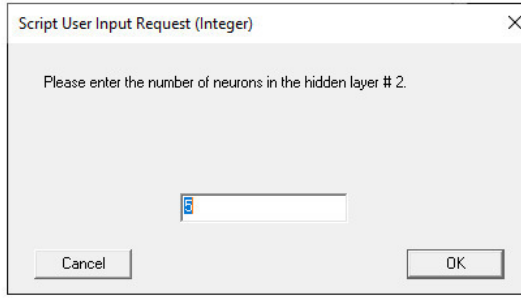

Рис.47 Количество нейронов в скрытом слое 2

При ответе Да завершается построение нейросети, о чём сообщается в трассе скрипта:

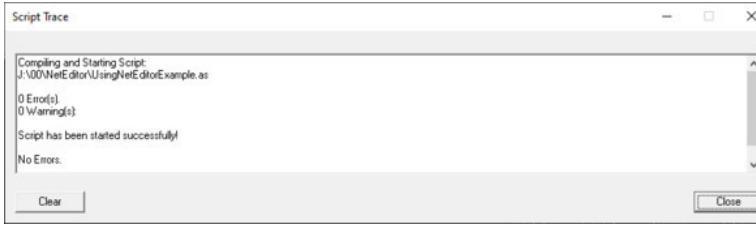

Рис.48 Часть трассы о завершении скрипта

Трасса в один кадр не умещается:

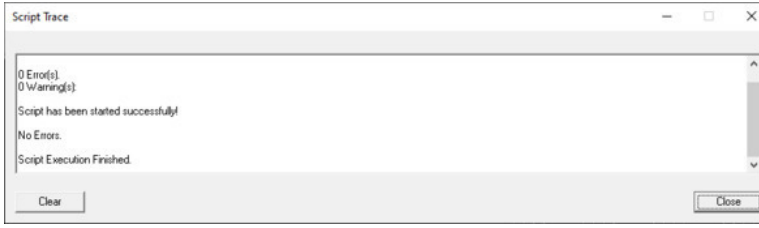

Рис.49 Часть трассы о завершении скрипта

На экран выводится готовая нейросеть.

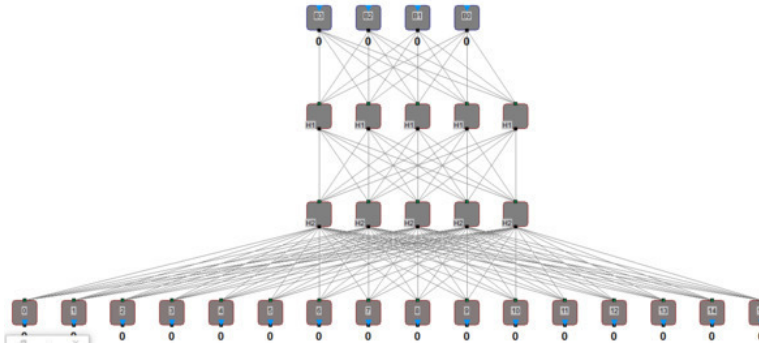

Рис.50 Нейросеть создана

В том случае, если не используем mbl-файл, запрашивается количество входных нейронов:

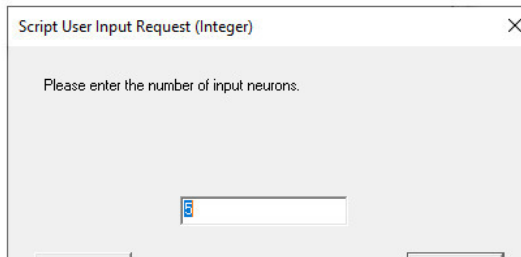

Рис.51 Количество входных нейронов

Да. Затем запрашивается количество выходных нейронов:

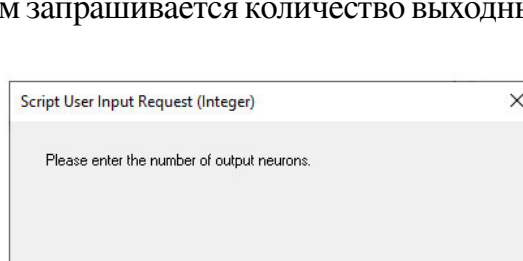

Рис.52 Количество выходных нейронов

Допустим, определяем это количество цифрой 3, затем запрашивается количество нейронов в каждом скрытом слое

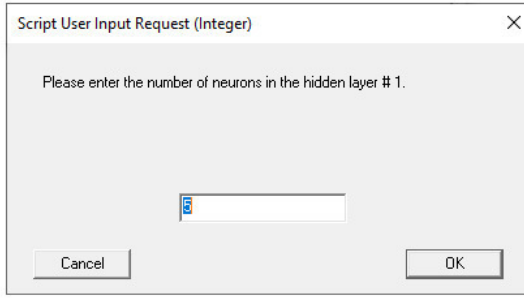

Рис.53 Запрос количества слоёв в скрытом слое 1

На этот вопрос ответ – 4. Появляется следующий вопрос о количестве нейронов в скрытом слое №2:

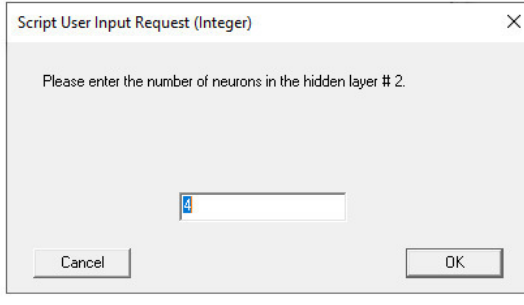

Рис.54 Запрос количества слоёв в скрытом слое 2

Ответ на этот вопрос – 7
Выводится трасса скрипта

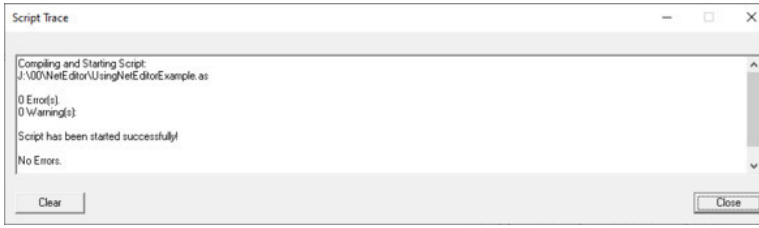

Рис.55 Трасса скрипта

Трасса в один кадр не умещается:

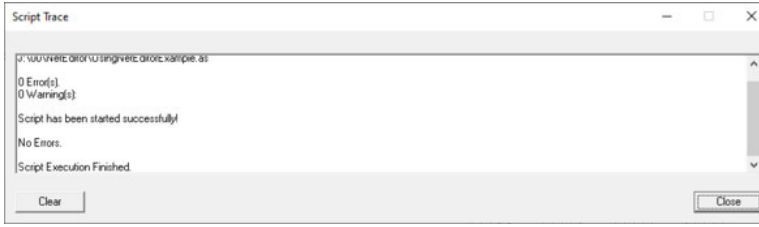

Рис.56 Второе сообщение о трассе скрипта

На экране остаётся схема созданной нейросети:

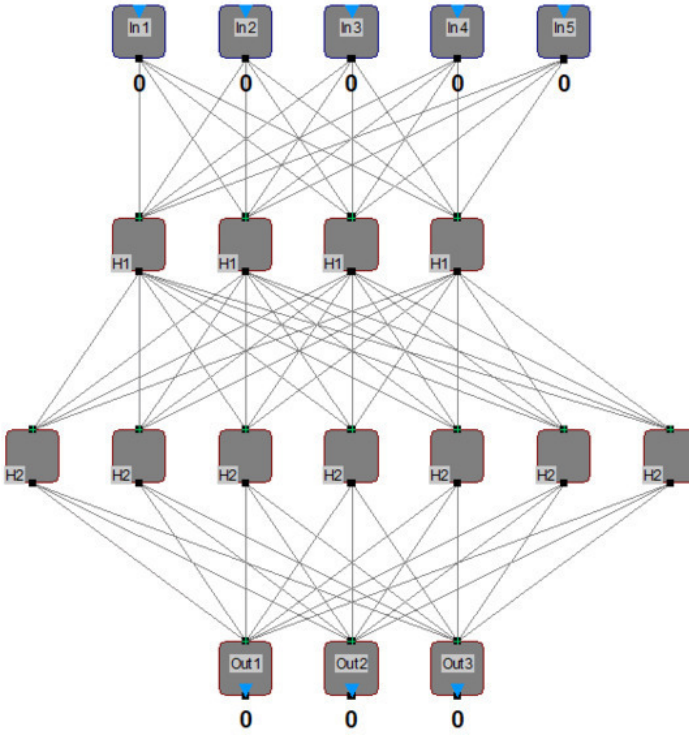

Рис.57 Схема созданной нейросети

Естественно, по умолчанию устанавливаются типовые свойства всех нейронов.

Начинается работа этой скриптовой группы с «Using Net Editor Example… as». Затем по мере необходимости к нему

подключаются ещё два скрипта.

В начале скрипта «Using Net Editor Example… as» содержится раздел «Константы», определяющий значения наиболее важных параметров создаваемой нейросети с помощью команды const. Затем создаётся список глобальных переменных, уточняются параметры создаваемой нейросети, после чего начинается сборка нейросети..

При необходимости выяснить количество скрытых слоёв в создаваемой нейросети проводится следующий диалог:

```
// Determine how many normal hidden layers there shall be
in the net
void DetermineHidLayerCount ()
{
int count;

if (UserInput («Please enter the number of hidden layers for
the net.»,
DEFAULT_HID_LAYER_COUNT, count) == IDOK)
{
if (count> gMaxHidLayerCount)
{
MessageBox («Too many hidden layers!»);
AbortScript ();
}

// Success. Adjust the size of the hidden layer array now.
gHidLayers.resize (count);
```

```
}
else
{
AbortScript ();
}
}
```

Создание входных нейронов по данным, содержащимся в mbl-файле

```
// Create the I/O neurons of the net using a MemBrain
lesson file
void CreateIOFromMbl ()
{
string lessonFileName;

if (FileOpenDlg («Select Lesson File», «mbl», «»,
lessonFileName) == IDOK)
{
// Load the lesson into #1 (delete all other lessons)
SetLessonCount (1);
LoadLesson (lessonFileName);
gInputCount = GetLessonInputCount ();
gOutputCount = GetLessonOutputCount ();
if (gInputCount> gMaxCountPerLayer)
{
MessageBox («Too many input columns in lesson!»);
AbortScript ();
```

```
}
else if (gOutputCount> gMaxCountPerLayer)
{
MessageBox («Too many output columns in lesson!»);
AbortScript ();
}

// Now we tell the editor to actually create the input and
output neurons.
// The number of hidden layers is required here to leave space
for them.
```

Это пример дискуссии с компьютером о способе создания базового компьютера с использованием mbl-файла

```
// Create net based on lesson or manual input?
if (MessageBox («Do you want to create the net based on
a MemBrain lesson file?», MB_YESNO) == IDYES)
{
gUseLesson = true;
CreateIOFromMbl ();
}
else
{
CreateIOFromUserInput ();

}
```

Здесь выводится сообщение с запросом ответа Да

или Нет.

При проверке очередного условия может проявиться превышение ожидаемого параметра:

```
{
if (count> gMaxHidLayerCount)
{
MessageBox («Too many hidden layers!»);
AbortScript ();

}
```

Тогда выводится сообщение о чрезмерном количестве скрытых файлов.

При удовлетворительном ответе:

```
// Success. Adjust the size of the hidden layer array now.
gHidLayers.resize (count);
}
```

Группирование нейросетей в нейроконструкции

(Перевод фрагментов архива «MemBrainExamples».)

До сих пор мы работали с нейропакетом каждый раз преимущественно – только с одной нейросетью. При работе с нейросетевыми конструкциями необходимо работать одновременно с несколькими нейросетями, одновременно находящимися на экране, переключаясь между ними. Такая возможность достигается за счёт группирования нейросетей.

Можно определить отношения разных типов между группами нейронов в сети. Это позволяет определять подсети внутри сети, которые затем могут обучаться отдельно с использованием различных алгоритмов и наборов данных.

Рассмотрим последовательно возникающие при этом проблемы :

– Что такое групповые отношения

– Как групповые отношения создаются и редактируются

– Доступные типы групповых отношений

– Использование групповых отношений для работы с подсетями

Конец ознакомительного фрагмента.