Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Дагестанский государственный технический университет»

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

Нейронные сети

Методические указания

Махачкала 2020

Оглавление

ВВЕДЕНИЕ

В рамках данного проекта было создано комплексное программное решение для обучения нейронным сетям с названием NeuroLearn, которое состоит из двух, во многом независимых модулей. Первый модуль, отвечает за практическую работу с нейронными сетями – NeuroLearn Studio, второй же отвечает за контроль над учебным процессом – NeuroLearn Class. Первый это настольное приложение под Windows, второе же Web-приложение. Связь происходит посредствам Интернет службы NeuroNet Service. Актуальность и интерес данной задачи объясняется тем, что если пакетов для работы с обыкновенными нейронными сетями довольно много (Statistica Neural Networks, NeuroShell, Matlab Neural Network Toolbox, NeuroSolutions, BrainMaker), то для работы с нечеткими нейронными сетями их гораздо меньше. Некоммерческого программного обеспечения в данной области вообще найдено не было. Так же интересна и актуальна интеграция с модулем обучения, не встречающаяся в других пакетах.

I РЕАЛИЗАЦИЯ NEUROLEARN STUDIO

I. 1. Общее описание

В рамках данного проекта была разработана NeuroLearn Studio (визуальная среда, позволяющая моделировать нейронные сети, производить их обучение и тестирование), а также набор дополнительных модулей, обеспечивающих функционирование системы (NeuronNetInterfaces.dll, NetLogger.dll, NeuronNet.dll, Neuron.dll, NeuronNetSerialiser.dll). Обо всех этих модулях ниже будет рассказано более подробно.

Модуль NeuronNetInterfaces.dll статически подключен к NeuronNet.dll, Neuron.dll, NeuronNetSerializer.dll, NeuroLearn Studio.exe и содержит декларацию интерфейсов, используемых в системе. В частности, в нем присутствует иерархия интерфейсов для нейронов, нейросетей и фабрик позволяющих конструировать эти объекты. Также в этом файле представлены интерфейсы для классов сериализации и классов определяющих стратегии поведения нейронов. Полная иерархия интерфейсов приведена на Рис. 1. Разберем ее более подробно.

Иерархия интерфейсов унаследованных от INeuron является иерархией интерфейсов нейронов и реализована в Neuron.dll. В иерархии присутствует три интерфейса INeuron, ISimpleNeuron и IFuzzyNeuron. IFuzzyNeuron и ISimpleNeuron унаследованы соответственно от INeuron и предоставляют расширение базовой функциональности соответственно для нейронов, представляющих нечеткие множества и нейронов с весами, сумматором и активационной функцией. В качестве стратегий поведения для классов с подобными интерфейсами должны существовать классы реализующие IActivationFunctor (для ISimpleNeuron) и IFuzzyFunctor (для ISimpleNeuron). Это классы, где содержатся активационные функции для нейронов с весами, и функции, реализующие нечеткие множества для нечетких нейронов. Также

в них содержатся производные необходимых порядков и параметры, определяющие вид функций.

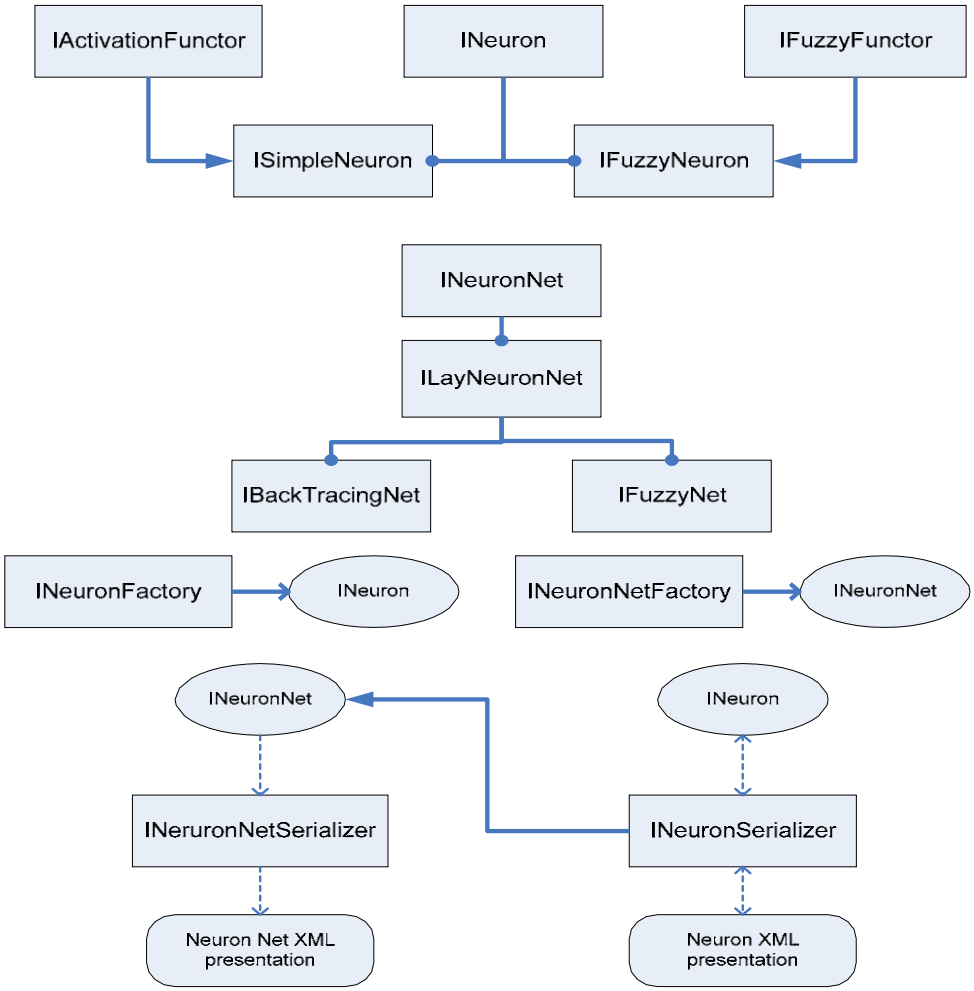

Рис. 1

Иерархия интерфейсов

Для каждой иерархии нейронов должна быть реализована фабрика, с помощью которой можно будет инстанцировать эти объекты (реализующая интерфейс INeuronNetFactory). Говоря языком шаблонов проектирования, в данном случае INeuronNetFactory представляет собой абстрактную фабрику. Конкретная фабрика инстанцируется в NeuroLearn Studio с помощью механизма рефлексии. Таким образом, отсутствует привязка к конкретным реализациям фабрик в коде (названия сборок и типов прописываются в конфигурационном файле). Опять же, если обратиться к терминологии, принятой в

шаблонах проектирования, это является реализацией принципа Injection of Control.

Иерархия интерфейсов, унаследованных от INeuronNet, по аналогии с нейронами образует набор интерфейсов нейронных сетей. ILayNeuronNet является базовым интерфейсом для слоистых нейронных сетей. От него унаследованы IBackTracingNet и IFuzzyNet (интерфейсы для сетей обучающихся по алгоритму обратного распространения ошибки и сетей с нечеткой логикой). Для каждой иерархии нейронных сетей также должны быть реализованы соответствующие фабрики объектов, реализующие интерфейс INeuronNetFactory (абстрактная фабрика для нейронных сетей). В NeuroLearn Studio подобные фабрики инстанцируются тем же способом, что и фабрики нейронов. Надо заметить, что каждая нейронная сеть в отдельности должна содержать фабрику нейронов. Иерархия классов, реализующих эти интерфейсы находится в сборке NeuronNet.dll

Была также предусмотрена функциональность, позволяющая сериализовывать сети (сохраненять их в XML виде и восстановлять из него). Классы сериализаторов должны реализовывать интерфейсы INeuronSerializer и INeuronNetSerializer. INeuronNetSerializer предполагает только сериализацию в XML без восстановления (это осуществляется в соответствующих фабриках или даже конструкторах классов). INeuronSerializer предполагает как сериализацию так и восстановление нейронов из XML. Сериализаторы реализованы в NeuronNetSerializer.dll. Соотвествующие спецификации приведены в П. I. 2.

NetLogger.dll содержит в себе реализацию класса Log, который обеспечивает журналирование работы системы. В частности, при подключенном отладчике этот класс записывает сообщения об ошибках и другую информацию именно в окно отладчика. Кроме того, Log способен делать записи в системный журнал событий, а также в специальный файл на диске. Log способен идентифицировать текущий поток и определять свою точку вывода

информации для каждого потока (использовался механизм на основе делегатов).

NeuroLearn Studio представляет собой среду с пользовательским интерфейсом позволяющую моделировать, обучать и исследовать нейронные сети и является Windows приложением на основе Windows.Forms. NeuroLearn Studio позволяет пользователю загружать проекты с сервера (проект состоит из текста задачи и обучающей выборки, также он может включать структуру сети). Проект может быть также загружен с локального жесткого диска или сохранен на жесткий диск или другой носитель. Моделирование, редактирование и обучение сетей происходит в интерактивном режиме. Среда позволяет пользователю изменять параметры нейронов, добавлять их и удалять (в случае нечетких сетей реализовано редактирование нечетких множеств и нечетких правил). Обучение сети происходит под контролем пользователя (обучение может быть как автоматическим, так и итеративным или пошаговым). Сеть способна вернуться в предыдущее состояние с помощью механизма отката (сериализация в XML).

I. 2. Нейроны.

В файле Neuron.dll реализовано две иерархии нейронов: их поведение различается в основном в механизме привязки друг к другу. В одном случае этот механизм построен на модели событий .NET, во втором случае, нейроны явно содержат на другие нейроны, которым они передают сигналы. Рассмотрим базовый класс, в случае реализации на основе делегатов (Листинг 1).

```
public abstract class NeuronD:INeuron
{
                /// <summary>
                /// Casual value generator
                 /// <remarks>
        /// Realization of Neuron uses delegates and events to connect Neurons
        /// </remarks>
        /// </summary>
                protected System.Random rand;
                /// <summary>
                /// Enter list
```

```
/// </summary>
protected ArrayList enter;
/// <summary>
/// Connection List of neurons
/// </summary>
public IList ConnList
{
        get
        {
                ArrayList cNeuList = new ArrayList();
                if(SendSignal != null)
                {
                        System.Delegate[] cInvList = SendSignal.GetInvocationList();
                                foreach(System.Delegate objDel in cInvList)
                        {
                                cNeuList.Add(objDel.Target);
                        }
                }
                return cNeuList;
        }
}
/// <summary>
/// Enter number
/// </summary>
public virtual int V
{
        get
        {
                return v;
        }
        set
        {
                if(value > 0)
                        v = value;
        }
}
/// <summary>
/// Enter number
/// </summary>
protected int v;
/// <summary>
/// Current val
/// </summary>
protected double ans = 0.0;
/// <summary>
/// Current val
/// </summary>
public double RetVal{get{return ans;}}
/// <summary>
/// Constructor
/// </summary>
/// <param name="n">Enter</param>
```

```
protected NeuronD(int n)
{
	if(n <= 0) throw new Exception("Wrong param in NeuronD constructor!");
	enter = new ArrayList();
rand = new Random();
}
/// <summary>
/// Neuron functionality
/// </summary>
protected abstract void go();
/// <summary>
/// Adds new signal
/// </summary>
/// <param name="sender">Sender object</param>
/// <param name="e">Event arguments</param>
public virtual void AddSignal(object sender,NeuronEventArgs e)
{
	if(v > 0)
	{
		enter.Add(e.Signal);
		if(enter.Count == v)
		{
			this.go();
			NeuronEventArgs objE = new NeuronEventArgs(ans);
			SendSignal(this,objE);
			enter.Clear();
		}
	}
}
/// <summary>
/// Connects different Neurons
/// </summary>
protected event SenderDelegate SendSignal;
#region INeuron Members
/// <summary>
/// Connnect this Neuron with others
/// </summary>
/// <param name="neu">Neuron ref</param>
public virtual void ConnectTo(INeuron neu)
{
	if(SendSignal != null)
	{
		if(!Connected(neu))SendSignal += new SenderDelegate(neu.AddSignal);

	else
	{
		SendSignal = new SenderDelegate(neu.AddSignal);
	}
}
/// <summary>
/// Disconnects two Neurons
/// </summary>
```

```
        /// <param name="neu">Ref to Neuron</param>
        public virtual void Disconnect(INeuron neu)
        {
                if(SendSignal != null)
                {
                        if(Connected(neu))SendSignal-=new SenderDelegate(neu.AddSignal);
                }
        }
        /// <summary>
        /// Disconnects Neuron with list of Neurons
        /// </summary>
        /// <param name="neulist">List of Neurons</param>
        public virtual void Disconnect(IList neulist)
        {
                if(SendSignal != null)
                {
                        foreach(INeuron neu in neulist)
                        {
                                if(Connected(neu))Disconnect(neu);
                        }
                }
        }
        /// <summary>
        /// Exams the connection
        /// </summary>
        /// <param name="neu">Ref to Neuron</param>
        /// <returns>true: connected false: disconnected</returns>

public virtual bool Connected(INeuron neu)
        {
                if(SendSignal != null)
                {
                        Delegate[] dellist = SendSignal.GetInvocationList();
                        foreach(Delegate del in dellist)
                        {
                                if(del.Target == (object)neu)return true;
                        }
                }
                return false;

        }
        /// <summary>
        /// Connects list of Neurons
        /// </summary>
        /// <param name="neulist">Neuron list</param>
        public virtual void ConnectTo(IList neulist)
        {
                if(SendSignal != null)
                {
                        foreach(INeuron neu in neulist)
                        {
                                if(!Connected(neu))ConnectTo(neu);
```

```
                        }
                }
        }
        /// <summary>
        /// Returns the name of current Neuron
        /// </summary>
        /// <returns>Neuron name</returns>
        public virtual string Name
        {
                get
                {
                        return null;
                }
        }

        #endregion
}
```

Как видно этот класс реализует интерфейс INeuron. Рассмотрим функции ConnectTo и Disconect. В этих функциях реализован способ привязки нейрона к другим нейронам (возможно даже порожденным другой фабрикой). Как видно он строится на основе добавления к событию SendSignal делегатов инкапсулирующих вызов функций AddSignal других нейронов. С помощью свойства ConnList можно получить все нейроны, сигнал к которым передается от данного нейрона. Сам механизм функционирования нейронов можно описать следующим образом: сначала вызывается функция AddSignal данного нейрона (она вызывается ровно столько раз, сколько в нейроне определено входов (значение свойства V)). Если количество сигналов стало равно количеству входов, то функция AddSignal вызывает функцию go для каждого нейрона реализованную соответствующим образом, которая преобразует входной сигнал в выходной, после чего выходной сигнал упаковывается в оболочку (объект NeuronEventArgs) и посылается другим нейронам через событие SendSignal, входной вектор очищается — нейрон готов к новой фазе функционирования.

Теперь мы обсудим функционирование нейронов реализующих интерфейсы ISimpleNeuron и IFuzzyNeuron. Функциональность остальных нейронов тривиальна.

FuzzyNeuronD реализует вышеуказанные интерфейсы, в нем основную функциональность несет объект FuzzyFunction типа IFuzzyFunctor. IFuzzyFunctor реализован в классе FuzzyFunctor. В этом классе реализована собственно функция, которая определяет нечеткое множество, параметры влияющие на ее вид, а также производные по соответствующим параметрам.

Осталось отметить, что кроме перечисленных также были реализованы нейроны, на выходе выдающие минимальный из сигналов, максимальный, сумму, произведение, значение любой функции, а также пустой нейрон являющийся точкой входа для сети.

I. 3. Нейронные сети.

Классы нейронных сетей реализованы в сборке NeuronNet.dll. Базовым классом является абстрактный BaseNeuronNet, реализующий INeuronNet, в этом классе определены несколько базовых функций и членов для всех сетей. Ниже по иерархии идет класс BaceLayNeuronNet, который является базовым для слоистых нейронных сетей. В нем определены разнообразные операции, такие как добавление и удаление отдельных нейронов, а также слоев и связей. Кроме того, реализована базовая функция Start, отвечающая за запуск сети и получение выходного вектора. Далее следуют два класса реализующие сети с алгоритмом обучения: FuzzyNeuronNet и SimpleNeuronNet.

Рассмотрим класс FuzzyNeuronNet. Этот класс, как говорилось выше, является наследником BaseLayNeuronNet, кроме того, он также реализует интерфейс IFuzzyNeuronNet. Данная сеть представляет собой модуль нечеткого управления и обучается по описанному выше алгоритму обратного распространения ошибки. Знания, составляющие основу корректного функционирования сети, записываются в виде нечетких правил вида:

$$R^{(k)} : IF(x_1 \ \ это \ \ A_1^k \ \ AND \ \ x_2 \ \ это \ \ A_2^k \ \ ...AND \ \ x_n \ \ это \ \ A_n^k) \ \ THEN \ \ (y_1 \ \ это \ \ B_1^k)$$

Рассмотрим структуру этой сети по слоям. В нулевом слое сети находятся входные нейроны. Этот слой реализует собой функцию принадлежности

нечеткого множества A_i^k, $i=1,..,n;\ k=1,...,N.$ На его вход подаются входные сигналы $\bar{x_i}$, а на выходе формируется значение функции принадлежности этих сигналов, т.е. во введенных в предыдущей главе обозначениях $\mu_{A_i^k}(\bar{x}_i)$ Далее следует слой нечетких нейронов, реализующих интерфейс IFuzzyNeuron и определяющих вид нечетких множеств. Количество нечетких множеств для каждой координаты неограниченно, с помощью методов AddFuzzySet и RemoveFuzzySet можно его регулировать. Конфигурация связей этого слоя соответствует базе правил. Далее следует слой нейронов, которые считают произведение (каждый нейрон соответствует некоторому нечеткому правилу в данном случае). Данный слой состоит из так называемых мультиплекаторов, которые перемножают все входящие параметры. Мультиплекаторы реализуют блок вывода . Слой этих нейронов завершает блок фуззификации, после чего следует блок деффузификации, состоящий из двух нейронов класса ISimpleNeuron. Веса одного из нейронов образуют в данном случае центры нечетких множеств B_k .Здесь был применен метод деффузификации по среднему центру.

В сети вида IFuzzyNet существует три функции обучения (собственно как и в любой другой слоистой сети в данной реализации): Inc, Epoch и Teach. Метод Inc является инкрементом обучения, атомарной обучающей операцией на одном векторе (именно в этой функции реализован алгоритм обучения). Метод Epoch выполняет эпоху обучения на обучающей выборке: на каждом обучающем векторе происходит инкремент обучения. Метод Teach определяет полностью автоматическое обучение. В нем можно задавать число эпох, точность и другие параметры, влияющие на процесс обучения и его сходимость.

В сети можно добавлять и удалять нечеткие правила. Делается это с помощью методов AddRule и RemoveRule. Эти функции позволяют указывать нечеткие множества участвующие в правилах и определять центр результирующего нечеткого множества B_k. Код сети приведен в приложении.

Другим примером реализации нейронной сети является сеть BackTracingNet обучающаяся по алгоритму обратного распространения ошибки. Эта сеть не является модулем нечеткого управления, но может быть использована в качестве компонента в подобном модуле (как это сделано в модуле Такаги-Сугено в [1]). Кроме того, подобные сети представляют и самостоятельный интерес, поскольку являются одними из наиболее изученных и помогают понять основной принцип функционирования нейронных сетей. Для этой сети реализованы операции добавления и удаления нейронов и слоев, соответствующие подобной архитектуре (в отличие от нечеткого модуля, где операции определены не в терминах нейронов а в терминах нечетких множеств и правил). Также как и в нечетком модуле для обучения существуют три метода: Inc, Epoch и Teach, функциональность которых аналогична методам из нечеткого модуля. Код сети также приведен в приложении.

II РЕАЛИЗАЦИЯ NEUROLEAN CLASS

Эта часть посвящена описанию той части, которая занимается администрированием процесса обучения.

II. 1. Общее описание.

Neuro Learn Class представляет собой ASP.NET приложение для удаленного администрирования учебного процесса, и выполняет следующие основные функции:

- Работа с учебными материалами (УМК).
- Работа с пользователями.
- Предоставление сведений о состоянии учебного процесса.

Как следует из предыдущей главы это необходимые элементы любого программного обеспечения нацеленного на обучение. Остановимся на конкретной реализации общих принципов в конкретной работе.

Выбор технологии ASP.NET объясняется тем, что доступ ко всему инструментарию осуществляется через web интерфейс. Благодаря современным технологиям и применению таких средств как AJAX, обеспечивающих удобство интерфейса близкое к настольному, программы, не требующие высокой производительности, стало предпочтительнее делать средствами WEB из-за их, следующей отсюда, доступностью ото всюду где есть Интернет. Кроме того весь проект писался под Windows с использованием .NET Framework, так как эта технология сейчас передовая и точно не сдаст свои позиции в ближайшие годы.

NeuroLearn Class писался по возможности не учитывая индивидуальные особенности NeuroLearn Studio, чтобы при необходимости предоставления сервисов другой программе не приходилось менять архитектуру всего NeuroLearn Class. Связь с инструментом, считывающим данные происходит через web-service с методами общей функциональности.

II. 2. Работа с учебными материалами

В вершине иерархии (Рис. 2) учебного материала лежит предмет. Данная сущность была введена именно для поддержки возможности широкопрофильного использования. Предмет может называться, например, “Биология” и для работы с ним нам понадобится BIO Studio, однако в конкретном нашем случае предмет называется “ Нейронные сети”.

Предметы можно создавать, удалять редактировать.

Изменяемые поля - имя, описание и используемый для работы инструмент, такой как NeuroLearn Studio.

Далее по иерархии идет курс лекций.

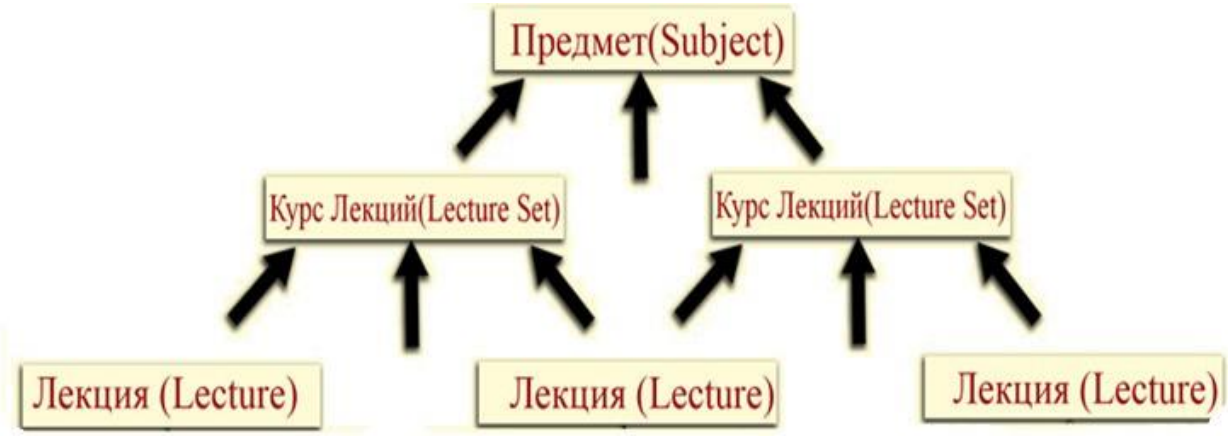

Рис. 2

Верархия учебного матерьяла.

Курс лекций группирует лекции для обучения какой-то группы. Например, предмет Нейронные сети можно преподавать первому и третьему курсу, и сложность и содержание лекций будут разные. Курсы лекций так же можно создавать удалять и редактировать.

Изменяемые поля – название, описание, принадлежность к группе студентов, принадлежность к предмету, принадлежность лекций.

Внизу иерархии лежит основная сущность, используемая в обучении – лекция.

Лекция в NeuroLearn Studio состоит из двух частей.

Первая часть – теоретическая.

Как описано в предыдущей главе, электронные учебные курсы целесообразно создавать в гипертекстовом виде. В таком виде лекция просто пересылается посредствам Интернет, просматривается любым браузером. Лекции в NeuroLearn именно гипертекстовые. Так же в лекцию можно вставлять файлы для скачивания, формат и количество которых не ограничено, что в принципе вообще ограничения на формат теоретической части лекции снимает. Редактируется текст лекции через удобный Wysiwyg редактор. Файлы хранятся в папке лекции на диске сервера. Доступ к ним имеет только авторизованный пользователь с правами на данную лекцию.

Вторая часть – практическая.

Как было описано во второй главе, NeuroLearn Studio сериализует нейронные сети и может сохранять или восстанавливать таким образом свое состояние. Практическая часть лекции состоит из описания практической задачи и опционально может включать в себя файл нейронной сети для работы с ней.

Рис 4.2

Практических заданий для одной лекции может быть сколько угодно. Каждое из них имеет весовой коэффициент оценки преподавателем для подсчета итоговой оценки за всю лекцию.

Таким образом, схематично лекцию можно представить как на рис. 4.2. Помимо этого лекция имеет порядковый номер, чтобы студент не справившийся с заданиями предыдущей лекции не брался за следующую.

За этим следит трэкер, который описывается ниже.

II. 3. Работа с пользователями

Пользователи в NeuroLearn хранятся на сервере, и у них обязательно должен быть тип, определяющий их степень доступа к системе. Всего существует 4 типа.

1. Администратор – полный доступ ко всему.
2. Наполнитель – может просматривать списки пользователей и их успеваемость, а так же полный доступ на создание, редактирование, удаление всех объектов из категории работы с учебными материалами (лекциями, курсами лекций, предметами).
3. Преподаватель – доступ к студентам своих групп и всем курсам лекций. Возможность назначть курсы лекций своим группам.
4. Студент – просмотр лекций и списка одногрупников.

Студенту и Преподавателю можно назначить группу, или несколько групп. Группам назначаются курсы лекций. Так пользователь связывается с лекциями.

Пользователи для авторизации NeuroLearn Studio через web-service – те же самые.

II. 4. Контроль над учебным процессом

Контроль за учебным процессом осуществляется посредством трэкера. При запросе лекции пользователем и начале работы с ним через web-service или же сайт, информация об этом событии заносится в базу данных. При успешном выполнении практических заданий студент получает доступ к следующей лекции.

Успешность проверяется преподавателем. Все работы сохраняются в общей базе данных. Из этой базы преподаватель может выбрать работу студента и проверить правильность построения сети. После чего преподаватель ставит оценку. Если практических заданий в результате много, то итоговая оценка за лекцию выставляется с учетом весов. В последующих версиях про-

граммы планируется добавить автоматические тесты сетей, а так же инструмент создания произвольных компьютерных тестов.

II. 5. Програмная реализация

В основном данная часть проекта это рутинное прикладное программирование. Однако было применено несколько интересных решений.

В фундаменте лежат классы связи с базой данных. Все они наследуются от класса DAO (Data Access Object).

При помощи рефлексии и программирования с использованием атрибутов было существенно упрощено обращение к базе данных. Например при определении полей класса пользователь

```
[PK]
[FieldNameAttr("Id")]
private int id;
[FieldNameAttr("Name")]
private String name;
[FieldNameAttr("MiddleName")]
private String middleName;
[FieldNameAttr("FamilyName")]
private String familyName;
[FieldNameAttr("Email")]
private String email;
[FieldNameAttr("CellPhone")]
private String phoneNumber;
[FieldNameAttr("Login")]
private String login;
[FieldNameAttr("PasswordHash")]
private String passwordHash;
[FieldNameAttr("Type")]
private int type;
```

благодаря атрибутам можно автоматически создавать пользователя по его Id в базе данных, изменять его поля, удалять.

Однако при выборке больших объемов для представления всех пользователей, были написаны отдельные хранимые процедуры.

Каждая сущность в проекте имеет свой класс для упрощения работы с ним. Если функционала, представляемого DAO не хватает (удалении, перезапись, удаление), то методы реализуются в классе с названием НазванииСущности Manager.

Для реализации отношений многие ко многим существуют специальные классы, которые наследуют интерфейс ILinkClass и класса DAO. И был напи-

сан контрол, AssigningCntrl.ascx, который при передаче в качестве параметра объект, реализующий ILinkClass, автоматически дает функционал по связи, отвязке и выборке связанных данных.

III Инструкция по развертыванию системы

Системные требования.

1. Сервер для web приложения должен иметь операционную систему Windows (xp/server2000/server2003) с установленным IIS 5.0 или IIS 6.0. Также на машине должен стоять .Net Framework 2.0
2. На сервере базы данных должен стоять Microsoft SQL Server 2005, и он должен быть доступен для сервера Web приложения. Сервер базы данных и сервер приложения могут находиться на одном компьютере.
3. Компьютер студента должен работать на баз Windows (xp/server2000/server2003), и у него должен быть доступ по http к серверу web приложения.

III. 2. Развертывание базы данных.

1. Развертывание необходимо начинать с развертывания базы данных. Для этого пошагово:
2. Открыть SQL Management Studio.
3. Подключиться к серверу который будете использовать.
4. На папке DataBase правой клавишей выбрать Restore DataBase...
5. Выбрать Source for restore: From Device.
6. Выбрать файл DataBase без расширения в корне компакт диска.
7. Ниже в окне “Select backup sets to restore” выберите появившуюся базу данных.
8. В выпалающем списке To Database: выберите базу NeuroLearn.
9. Нажмите OK.
10. Создайте пользователя для доступа к БД или используйте существующего с соответствующими параметрами. Например, пользователя sa.

III. 3. Развертывание сайта.

1. Сохраните файлы сайта в папке на жестком диске.
2. Создайте сайт или виртуальную директория на файл/файлы.
3. Дайте право на запись.
4. Откройте вкладку настройки виртуальной директории/сайта в закладке ASP.NET выберите версию 2.0

5. Откройте Web.config в корне сайта. Там поправьте

```
<appSettings>
            <add key="LogFilePath"
value="C:\Work\NL2\Class\NeuroLearnClass\bin"/>
            <add key="VAULT_PATH"
value="C:\Work\NL2\Class\NeuroLearnClass\LectureFiles"/>
      </appSettings>
      <connectionStrings>
            <add name="Local"
connectionString="Server=SUPERVASIA\SQL2005;database=NeuroLearn;uid=NL;pwd=1;
" />
            <add name="NeuroLearnConnectionString1" connectionString="Data
Source=SUPERVASIA\SQL2005;Initial Catalog=NeuroLearn;User ID=NL;Password=1"
                  providerName="System.Data.SqlClient" />
            </connectionStrings>
```

То, что выделено жирным надо изменить. LogFilePath — путь куда будет писаться log-файл.

VAULT_PATH – путь к папке с файлами лекций.

connectionStrings Local – изминте, указав параметры вашего сервера базы данных.

Настройка вэб сервиса абсолютно точно такая же.

III. 4. Развертка приложения.

Оно просто должно быть скопировано на компьютер пользоватнля вместе с файлом APP.config. В нем необходимо изменить путь к asmx файлу web сервиса.

Система готова к работе.