# Замечание

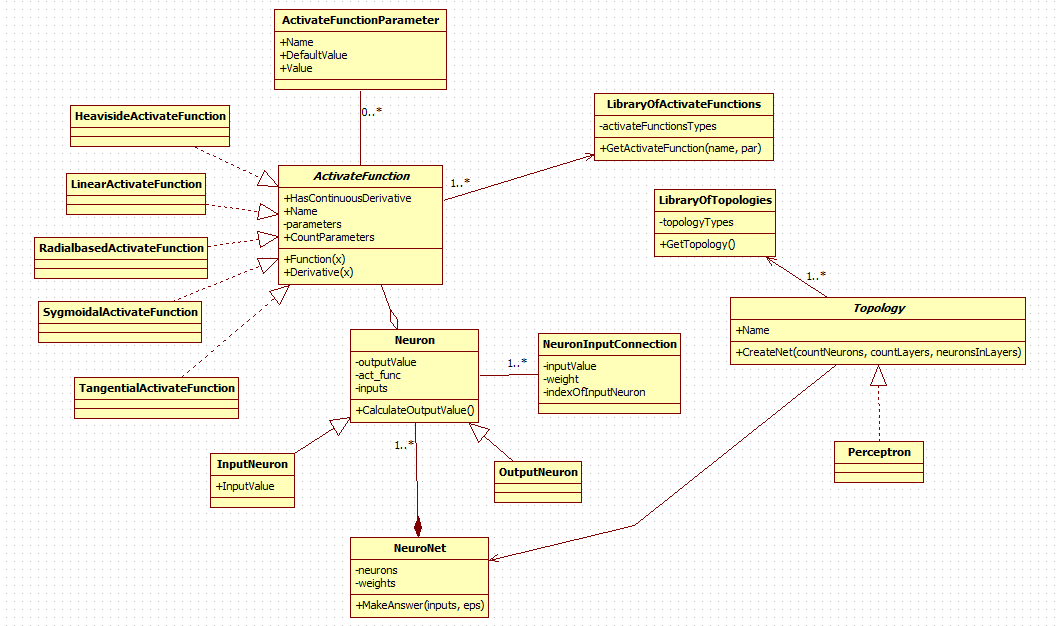
Данный документ детализирует архитектуру *проекта*, являющегося частью ИСИАД. Для получения информации об общей архитектуре и принятых соглашениях, обозначениях см. документ «DMTS-architecture».

# Архитектура *проекта* NeuroWnd

NeuroWnd является библиотекой классов, обеспечивающих работу искусственных нейронных сетей внутри ИСИАД. Данный *проект* состоит из следующих частей:

* Классы *форм* приложения, обеспечивающие интерфейс между пользователем и функциональными модулями *проекта*.
* Библиотека классов ИНС.
* Интерфейс работы с классами *проекта* **LearningAlgorithms**.
* Классы работы с БД ИСИАД.
* Вспомогательные классы, обеспечивающие взаимодействие между различными модулями в рамках данного *проекта*.

## Диаграмма библиотеки классов ИНС



## Классы *форм* приложения

*Проект* содержит следующие *формы*:

* NeuroNetsMainWindow – главное окно, открывающееся при запуске приложения. Позволяет просмотреть доступные ИНС, перейти в режим добавления новой ИНС, удалить существующую ИНС, перейти в окна, обеспечивающие обучение ИНС или решение задачи.
* NeuroNetSolvingWindow – окно, позволяющее решить задачу обученной ИНС. Позволяет ввести значение входных параметров и считать значение полученного выходного.
* AddChangeNeuroNetDialog – окно, позволяющее создать новую ИНС или изменить существующую.
* LoadingWindow – окно, выводящее состояние длительной операции.

## Библиотека классов ИНС

Данная библиотека состоит из следующих компонентов:

1. Пространство имен **NeuroWnd.Activate\_functions**, содержащее классы активационных функций.

1.1. Абстрактный класс **ActivateFunction**.

* public abstract bool HasContinuousDerivative – абстрактное свойство, имеет ли АФ непрерывную производную. Доступно только для чтения.
* public abstract string Name – название АФ, отображаемое в GUI приложения. Доступно только для чтения.
* public int CountParameters – свойство, возвращающее число параметров АФ. Доступно только для чтения.
* protected List<ActivateFunctionParameter> parameters – список параметров АФ.
* public string GetNameOfParameter(int index) – возвращает название параметра по его индексу в списке parameters.
* public double GetDefaultValueOfParameter(string name || int index) – возвращает значение параметра по умолчанию. На вход получает либо имя параметра, либо индекс в массиве parameters. (Это 2 различных метода, для краткости здесь представлены одним).
* public double GetValueOfParameter(string name || int index) – возвращает текущее значение параметра АФ.
* public void SetValueOfParameter(string name || int index, double value) – устанавливает значение параметра АФ равным value.
* public ActivateFunction()
* public ActivateFunction(ActivateFunction af)
* public abstract double Function(double x) – реализация самой АФ.
* public abstract double Derivative(double x) – реализация производной АФ.

1.2. Класс параметра АФ **ActivateFunctionParameter**.

* public string Name – название параметра АФ. Свойство, доступное только для чтения. Это название используется в GUI приложения.
* public double DefaultValue – Значение параметра по умолчанию. Данное свойство доступно только для чтения.
* public double Value – текущее значение параметра
* public ActivateFunctionParameter(string \_name, double \_defVal)
* public ActivateFunctionParameter(ActivateFunctionParameter par)

1.3. Класс **LibraryOfActivateFunctions**, через который остальное приложение получает доступ к нужной АФ и необходимой информации о ней в режиме выполнения.

* static public int GetCountActivateFunctions() – возвращает количество доступных реализаций класса ActivateFunction (наследников).
* static public string[] GetAllActivateFunctionNames() – возвращает значения поля Name всех реализаций АФ.
* static public string GetActivateFunctionName(string typeName) – возвращает значение поля Name класса АФ с именем typeName.
* static public string GetActivateFunctionTypeName(string activateFunctionName) – операция, обратная предыдущей.
* public enum GetterParameter { ActivateFunctionName, TypeOfActivateFunctionName } – перечисление, необходимое для следующих методов. Содержит пояснение, чем является входной параметр метода – значением поля Name или названием класса АФ.
* static public int GetCountParametersOfAF(string name, GetterParameter par) – возвращает число параметров АФ по имени класса АФ/ названию АФ.
* static public double GetDefaultValueOfParameterAF(string name, int indexPar || string namePar, GetterParameter par) – возвращает значение по умолчанию параметра АФ.
* static public ActivateFunction GetActivateFunction(string name, GetterParameter par) – возвращает класс активационной функции.

1.4. Реализации АФ: **HeavisideActivateFunction, LinearActivateFunction, RadialbasedActivateFunction, SygmpidalActivateFunction, TangentialActivateFunction.**

2. Пространство имен **NeuroWnd.Neuro\_Nets.**

2.1. Пространство имен **NeuroWnd.Neuron\_definition**.

2.1.1. Класс входа нейрона **NeuronInputConnection**

* public double inputValue – входное значение нейрона
* public double weigth – вес входной связи
* public int indexOfInputNeuron – индекс нейрона, от которого идет данная связь.
* public NeuronInputConnection(double \_weight, int \_indexOfInputNeuron)
* public NeuronInputConnection(double \_weight, double \_inputValue, int \_indexOfInputNeuron)
* public NeuronInputConnection(NeuronInputConnection nic)
  + 1. Класс **Neuron**
* public double OutputValue– выходное значение нейрона. Свойство только для чтения.
* public int InputsCount– количество входов нейрона.
* protected NeuronInputConnection[] inputs – входы нейрона. Свойство только для чтения.
* public ActivateFunction ActivateFunctionOfNeuron – АФ нейрона. Свойство только для чтения.
* public Neuron(ActivateFunction af)
* public Neuron(ActivateFunction af, NeuronInputConnection[] \_inputs)
* public Neuron(Neuron neu)
* public void SetInputConnections(NeuronInputConnection[] \_inputs) – устанавливает входные связи нейрона.
* public void SetInputValue(double inputValue, int indexOfInputConnection) – устанавливает значение связи с индексов indexOfInputConnection.
* public void SetWeightValue(double weightValue, int indexOfInputConnection) – устанавливает вес связи.
* public virtual void CalculateOutputValue() – вычисляет выходное значение, посчитав взвешенную сумму входов и применив АФ. Результат запишется в OutputValue.
  + 1. Класс входного нейрона ИНС **InputNeuron**
* public double InputValue – входное значение ИНС, поступающее на данный нейрон. Поле для чтения и записи.
* public InputNeuron(ActivateFunction af)
* public InputNeuron(ActivateFunction af, double inputValue)
* public InputNeuron(ActivateFunction af, double inputValue, NeuronInputConnection[] \_inputs)
* public InputNeuron(InputNeuron neu)
* public override void CalculateOutputValue()
  + 1. Класс выходного нейрона **OutputNeuron**
* public OutputNeuron(ActivateFunction af)
* public OutputNeuron(ActivateFunction af, NeuronInputConnection[] \_inputs)
* public OutputNeuron(OutputNeuron neu)
* public override void CalculateOutputValue()
  1. Класс **NeuroNet** – нейросеть любой топологии. Топология представляет собой отдельный класс. Для инициализации конкретной нейросети необходимо сформировать объект данного класса, передав ему в качестве параметра конструктора класс топологии.
* public int NeuronsCount – свойство только для чтения. Возвращает общее число нейронов в сети.
* public int InputNeuronsCount – число входных нейронов сети. Только для чтения.
* public int OutputNeuronsCount – число выходных нейронов сети. Только для чтения.
* public bool IsIterationsFinished – данное свойство равно true, если были подсчитаны значения всех выходных нейронов после последней подачи входных параметров сети. Только для чтения.
* public bool IsWaveCameToOutputNeuron – данное свойство равно true, если было хотя-бы один выходной нейрон посчитал свое выходное значение. Только для чтения.
* public int[] NeuronsInLayers – Только для чтения. Возвращает массив, в котором i-й элемент есть число нейронов в i-м слое. (Нумерация слоев с 0, от входного слоя до выходного).
* public bool[,] ConnectionsOfNeurons – Только для чтения. Возвращает матрицу инцидентности сети.
* public double[,] WeightsOfConnections – Только для чтения. Возвращает матрицу весовых коэффициентов сети.
* public NeuroNet(int numb\_input\_neurons, int numb\_output\_neurons, int[] numbNeuronsInLayers, bool[,] \_connections, double[,] \_weights, ActivateFunction af) – Нейроны ИНС могут иметь только один тип АФ. Причем каждому нейрону сопоставляется один и тот же объект АФ, т.о. параметры АФ имеют одинаковые значения для любого нейрона сети.
* public NeuroNet(NeuroNet net)
* public void ResetNeuroNet() – Обнулить выходные значения всех нейронов, перевести ИНС в начальное состояние.
* public Neuron GetNeuron(int index) – получить нейрон по внутреннему индексу в сети.
* public int GetIndexNeuron(Neuron neuron) – обратная операция.
* public void SetNewConnection(int indexInput, int indexOutput, double weight) – изменить вес связи между двумя нейронами.
* public void DeleteConnection(int indexInput, int indexOutput) – удалить связь между двумя нейронами, обнулив соответствующее значение матрицы весов и матрицы инцидентности.
* public double[] MakeStep(double[] inputs) – возвращает массив выходных значений нейронов выходного слоя после того, как было подсчитано хотя-бы одно данное выходное значение. Размер массива входных значений inputs должен быть равен числу входных нейронов ИНС.
* public double[] MakeIteration(double[] inputs) - возвращает массив выходных значений нейронов выходного слоя после того, как были подсчитаны все данные выходные значения.
* public double[] MakeAnswer(double[] inputs, double eps = 1E-16) – для рекурсивных нейронных сетей. Возвращает значения выходных нейронов после того, как эти значения перестанут отличаться от своего значения на предыдущей итерации более чем на eps.
  1. Абстрактный класс **Topology.**
* public Topology()
* public abstract bool[,] CreateNet(int countNeurons, int countLayers, int[] neuronsInLayers) – метод, возвращающий матрицу инцидентности данной топологии для сети с заданным количеством нейронов, слоев и распределением нейронов по слоям.
* public abstract string Name – свойство только для чтения. Представляет собой название топологии, отображаемое в приложении.
  1. Реализация топологии **Perceptron**
  2. Класс **LibraryOfTopologies –** через него остальное приложение получает доступ ко всем реализациям класса **Topology**, по аналогии с классом **LibraryOfActivateFunctions.**
* public enum GetterParameter { TopologyName, TypeOfTopologyName } – определяет семантику переменной name в методах, описанных ниже. Либо это название топологии, указанное в свойстве Name, либо это название класса, наследованного от **Topology**.
* public static Topology GetTopology(string name, GetterParameter par)
* public static List<string> GetAllTopologyTypeNames()
* public static string GetTopologyName(string typeName)
* public static string GetTopologyTypeName(string topologyName)

### Интерфейс работы с классами *проекта* **LearningAlgorithms**

В *проекте* **LearningAlgorithms** определен интерфейс **ILearning** для обучения решателей. От него наследован интерфейс **INeuroNetLearning**, определяющий взаимодействие алгоритма обучения и нейросети. Этот интерфейс объявляет следующие методы:

* bool[,] get\_bool\_links() – получить матрицу инцидентности.
* double[,] get\_links() – получить матрицу весов.
* void set\_links(double[,] links) – установить новую матрицу весов.
* double get\_res(double[] X) – получить результат работы сети по входному вектору X. Здесь подразумевается, что выходной параметр один.
* INeuroNetLearning copy() – получение копии реализации интерфейса (фактически, копии обучаемой ИНС).
* void write\_result(string algorithm) – запись текущих весовых коэффициентов в БД.

В пространстве имен **NeuroWnd.Neuro\_Nets** определены следующие классы:

**NeuronLocation,** определяющий позицию нейрона в ИНС:

* public int Layer – поле для чтения/записи.
* public int Number – поле для чтения/записи.
* public NeuronLocation(int \_layer, int \_number)

**NeuroNetLearningInterface**, являющийся реализацией **INeuroNetLearning**. Внутри себя содержит (агрегирует) объект класса **NeuroNet**, обучаемую ИНС. Помимо реализаций методов данного интерфейса, в нем объявлены:

* public int CountLayers – только для чтения. Число слоев в ИНС.
* public int CountNeurons – только для чтения. Число нейронов в ИНС.
* public int CountInputNeurons – только для чтения. Число входных нейронов.
* public int CountOutputNeurons – только для чтения. Число выходных нейронов.
* public bool IsIterationsFinished – аналог соответствующего свойства в **NeuroNet**
* public bool IsWaveCameToOutputNeuron – аналог соответствующего свойства в **NeuroNet**
* public NeuroNetLearningInterface(NeuroNet net, string \_neuroNetName, string \_selectionName, DataBaseHandler \_dbh)
* public NeuroNetLearningInterface(NeuroNetLearningInterface inn)
* public int GetCountNeuronsInLayer(int layerIndex)
* public NeuronLocation[] GetInputsOfNeuron(NeuronLocation neuronLocation) – получить массив входов данного нейрона.
* public NeuronLocation[] GetOutputsOfNeuron(NeuronLocation neuronLocation) – получить позиции нейронов следующего слоя, связанных с данным.
* public bool IsConnection(NeuronLocation input, NeuronLocation output) – возвращает true, если между данными нейронами есть связь.
* public double GetConnectionWeight(NeuronLocation input, NeuronLocation output)
* public Tuple<double, NeuronLocation>[] GetInputsOfNeuronWithWeights(NeuronLocation location) – получает пары вида «Позиция нейрона, являющегося входом данному», «Вес связи между этими нейронами».
* public void ChangeConnectionWeight(NeuronLocation input, NeuronLocation output, double weight)
* public void SetNewConnection(NeuronLocation input, NeuronLocation output, double weight)
* public void DeleteConnection(NeuronLocation input, NeuronLocation output)
* public double GetOutputValueOfNeuron(NeuronLocation neuron)
* public string GetNameOfAF(NeuronLocation neuron)
* public bool HasAFContinuousDerivative(NeuronLocation neuron)
* public int GetCountParametersAF(NeuronLocation neuron)
* public string[] GetNamesOfParametersAF(NeuronLocation neuron)
* public double GetValueOfParameterAF(NeuronLocation neuron, string parameterName)
* public void SetValueOfParameterAF(NeuronLocation neuron, string parameterName, double value)
* public double GetAFValue(NeuronLocation neuron, double x)
* public double GetDerivativeAFValue(NeuronLocation neuron, double x)
* public void ResetNeuroNet() – данный метод и все последующие являются аналогами соответствующих методов из **NeuroNet.**
* public double[] MakeStep(double[] inputs)
* public double[] MakeIteration(double[] inputs)
* public double[] MakeAnswer(double[] inputs, double eps = 1E-16)

### Классы работы с БД ИСИАД.

Для поддержания связи с БД реализован класс **SQLiteConnector.** Он содержит:

* public SQLiteConnection connection – класс связи из **System.Data.SQLite**
* public string dbPath – путь до БД.
* public void ConnectToDB()
* public void DisconnectFromDB()

Все запросы к БД, осуществляемые классами приложения, содержатся в классе **DataBaseHandler.**

### Вспомогательные классы

* **DataContainer<T>**, представляющий собой словарь для типа **Т**. Используется для отображения справочной информации о задачах и выборках в *форме* **NeuroNetsMainWindow**.
* **DataNode<T>** - узел данных для **DataContainer<T>**.
* **NeuroNetDefinition** – класс, содержащий справочную информацию о ИНС. Используется для пересылки данных между *формами*.