http://chipsystem.ru/

Нейронные сети и нейро нечеткие системы фирмы SIEMENS.

Для управления плохо изученными процессами фирма **SIEMENS**

предлагает программный пакет **NeuroSystem.**

Точность перевода гарантировать не могу, а излагаю то, как сам

понял этот пакет ПО, так что буду очень рад, если кто-нибудь

чего-нибудь добавит.

Преимущества и недостатки нейронных сетей:

☺ Большое преимущество нейронных сетей в решении проблемы, это использование данных из примера. Это данные могут быть в форме ряда измерений.

☺ Другое преимущество нейронных сетей – это их адаптируемость, то есть они могут приспосабливаться к новой ситуации, изменяя свое поведение.

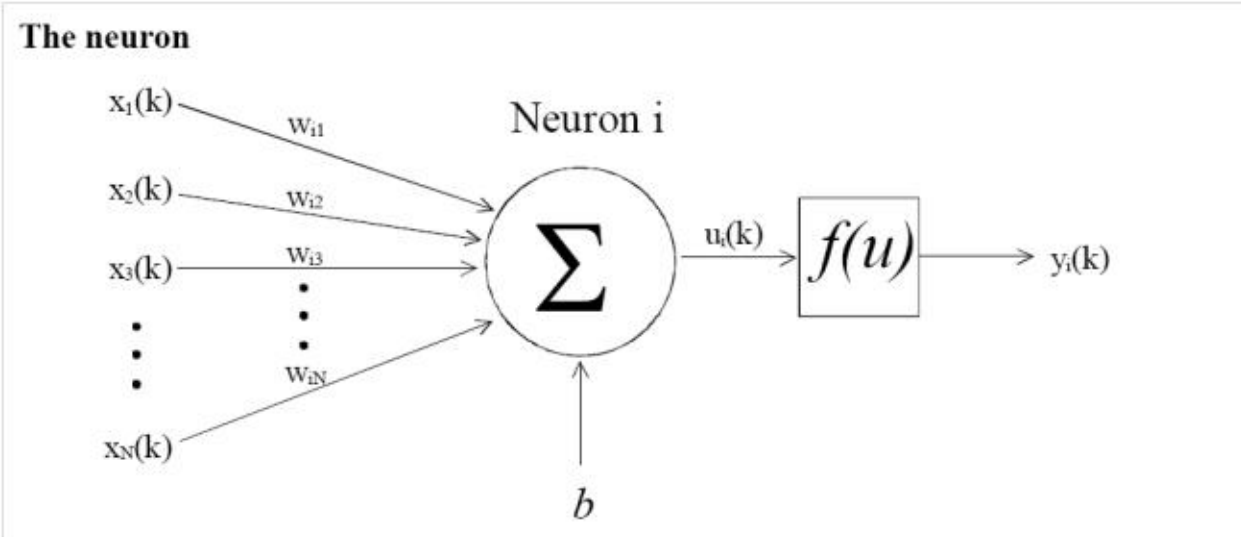
☺ Нейронная сеть особенно подходит для моделирования нелинейной зависимости.

– Один недостаток нейронных сетей в том, что вообще невозможно понять как они решили

проблему.

– Сеть - не умнее чем данные, которыми Вы "обучили", эту сеть, то есть в примере должно

быть адекватное представление проблемы.



На рисунке изображен MP нейрон разработанный У.Маккалоком и У.Питтсом, еще в 1943 году. Нейрон I имеет несколько входов **x (к)**, и один выход **Yi (к)**, где ***к*** время, от которого функция нейрона рассматривается. Из линий входов и выходов видно, что выходное значение зависит только от входных значений, а не наоборот.
(Выход может быть "подаваться обратно" на "дополнительный вход". Этот "нейрон с внутренней обратной связью " не обсуждается здесь.)
На поведение нейронов большое влияние имеет функция веса**(Wij)** и функция активации **f(U).**
Нейрон считается активным, если его выход имеет определенное значение, например, превышает пороговое значение.
Эта деятельность является результатом стимуляции входов и весов нейрона. Весовые коэффициенты могут способствовать стимуляции, если они больше нуля - или подавлять её, если они меньше нуля. Если весовой коэффициент равен нулю, то вход не влияет на деятельность нейрона.

Наиболее общие функции активации:

(a)Функция Выключателя: $$y = \begin{cases} 1, & \text{for } u > 0 \\ 0, & \text{else} \end{cases}$$

(b)Сигмоидальная функция: $$y = \frac{1}{1+e^{-u}}$$

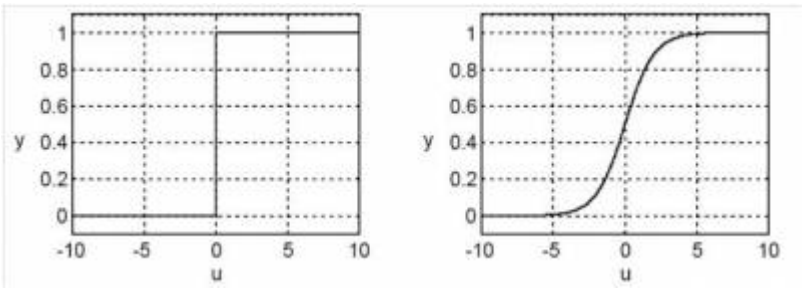
(c)Гиперболический тангенс: $$y = \tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}$$

(d)Линейная функция: $$y = u$$

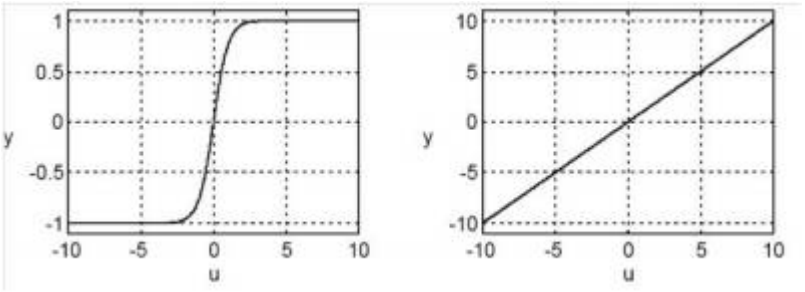
Следующие диаграммы показывают кривую для каждой функции.

(a) Функция выключателя

(b) Сигмойндная функция



(c) Гиперболический тангенс (d) Линейная функция



Статические сети и их свойства

Статические сети достаточно хорошо изучены и уже используются для широкого спектра применений.

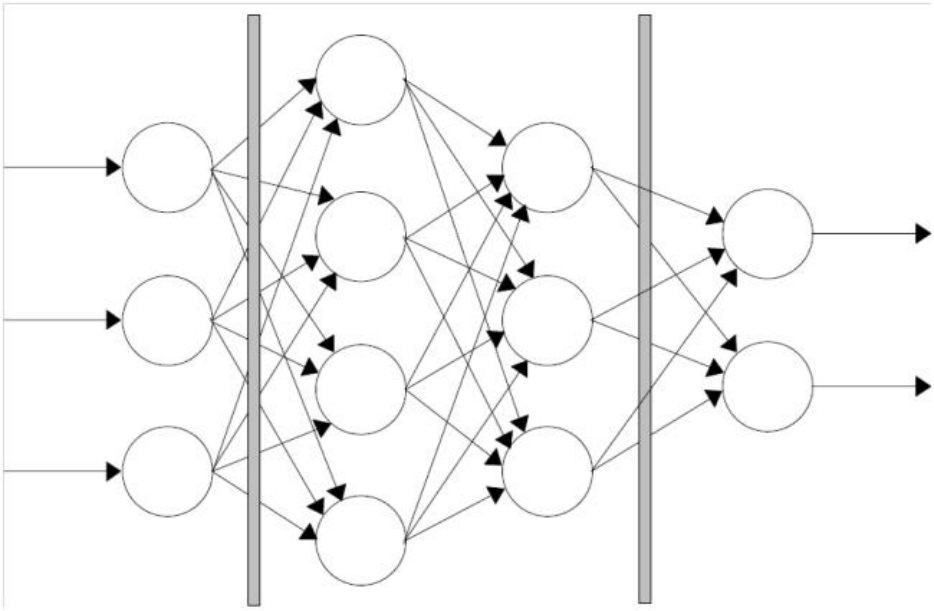
Динамические сети возникают потому, что внутренняя обратная связь была внедрена так, то выход нейрона в момент времени **к** зависит от его собственного выхода в момент **к-1** выход - дополнительный вход).

Динамические сети все еще находятся на стадии исследования и редко используются на практике, потому что большая трудность заключается в том чтобы убедиться в их устойчивости и комплексе обучения.

В этом ПО используются три типа нейронных сетей, это: **MLP, RBF** и **NeuroFuzzy**.

Многослойный персептрон (MLP Network)

Наверное, самый известный тип нейронной сети является многослойный персептрон. В этом случае, несколько нейронов соединены между собой прямыми связями. В сети **MLP**, нейроны расположены в несколько слоев. Два слоя, которые соединены с внешним миром называют входной и выходной слои. Слои между входным и выходным слоем, называют скрытыми слоями. Следующая диаграмма иллюстрирует эту структуру.



Входной слой Скрытые слои Выходной слой

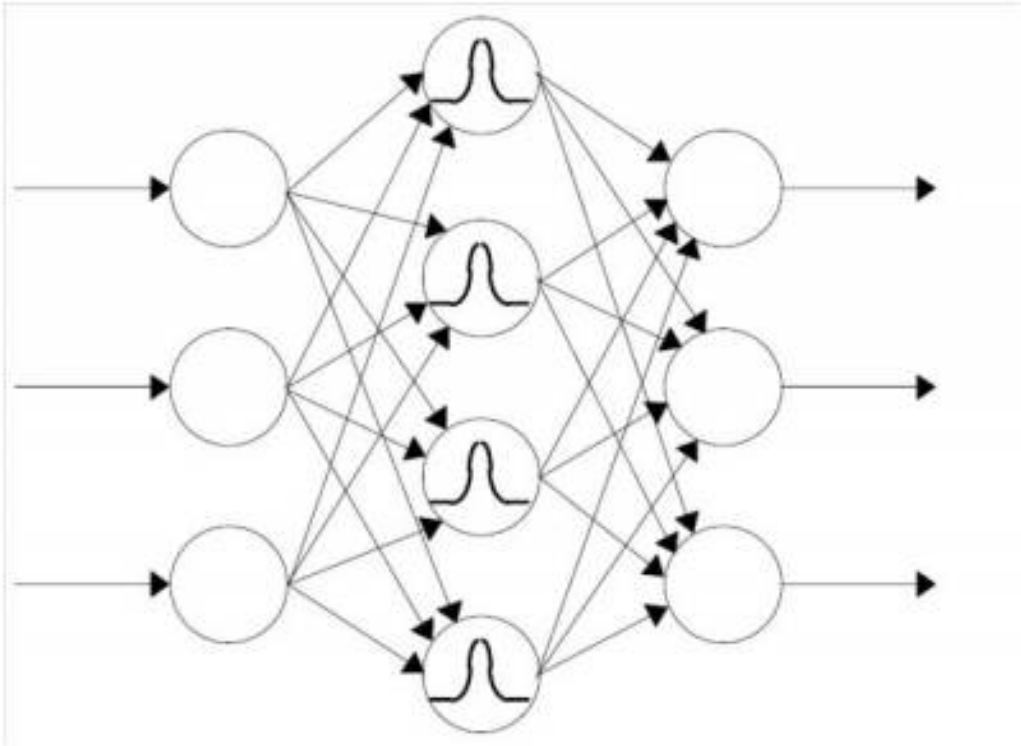
Нейроны входного слоя используются только для распределения входов к нейронам первого, скрытого слоя и поэтому каждый имеет только один вход. Вес входного соединения можно использовать для масштабирования входного сигнала. Входные нейроны являются частным случаем нейрона. Некоторые авторы, не включает входной слой в число слоев в сети. В **NEUROSYSTEMS**, входной слой учитывается.

Радиально базисные функции (RBF сети)

Метод, используемый в RBF сети, немного отличается от используемого для многослойного персептрона.

Они всегда состоят из трех слоев:

• Входной слой,
• Слой RBF нейронов, функция активации которых является функцией Гаусса (как показано на рисунке ниже),
• И выходного слоя.



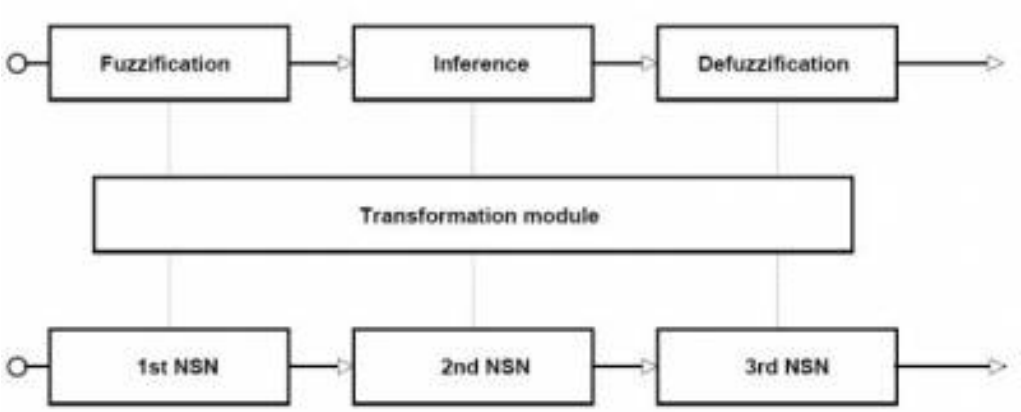
Входной слой Слой RBF Выходной слой

(Здесь: j=1 ... 3) (здесь: i=1 ... 4) (здесь: k=1 ... 3)

NeuroFuzzy сетей (NFN).

Для того чтобы воспользоваться преимуществами нейронных сетей и нечетких систем в одном проекте, вам потребуется система, которая обрабатывает нечеткие функции принадлежности и правила базы знаний нечеткой модели и может определить знания из выборки данных с использованием нейронных способов обучения. Вы можете решить эту проблему с помощью специальной нейронной сетью, называемой neurofuzzy сеть (NFN). Она состоит из трех нейронных подсетей (NSN), имитирующие три подзадачи - fuzzification, обработка правил и defuzzification.

НЕЧЕТКАЯ СИСТЕМА

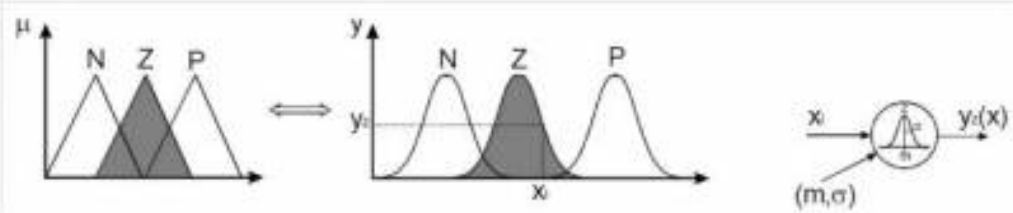


http://chipsystem.ru/

НЕЙРОННАЯ СЕТЬ

1) Fuzzification в 1-й слой NSN подсети.

Fuzzification – перевод входных переменных в нечеткий формат, реализует слой нейронов с функцией активации, аналогичной как в сети RBF. Один нейрон присваивается каждой функции принадлежности входных переменных.

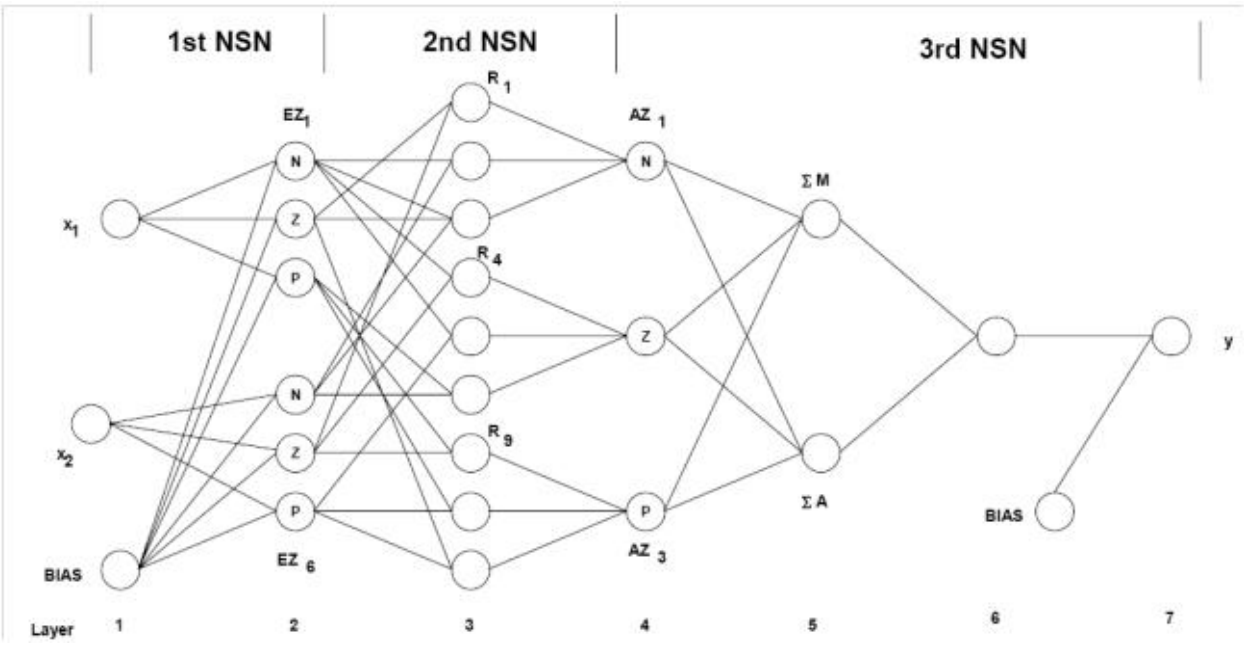


2) Вывод во 2-ом NSN слое подсети.

Основные правила базы знаний нечеткой системы находится в 2-ом NSN слое, и один нейрон назначен на каждое правило.

3) Defuzzification в 3-ем NSN слое подсети. Преобразование нечетких множеств в четкое число производиться одним из стандартных методов - центра тяжести.

Получается такая схема.



У системы иллюстрированной выше, есть входные сигналы **x1** и **x2**. Три функции: (Nnegative, Z-ноль, P-positive), распределенные на каждый входной сигнал. Три функции (N-negative, Z-ноль, P-positive) также приняты для выходного сигнала Y.

Демонстрация программного пакета NEUROSYSTEMS.

Создание сети.

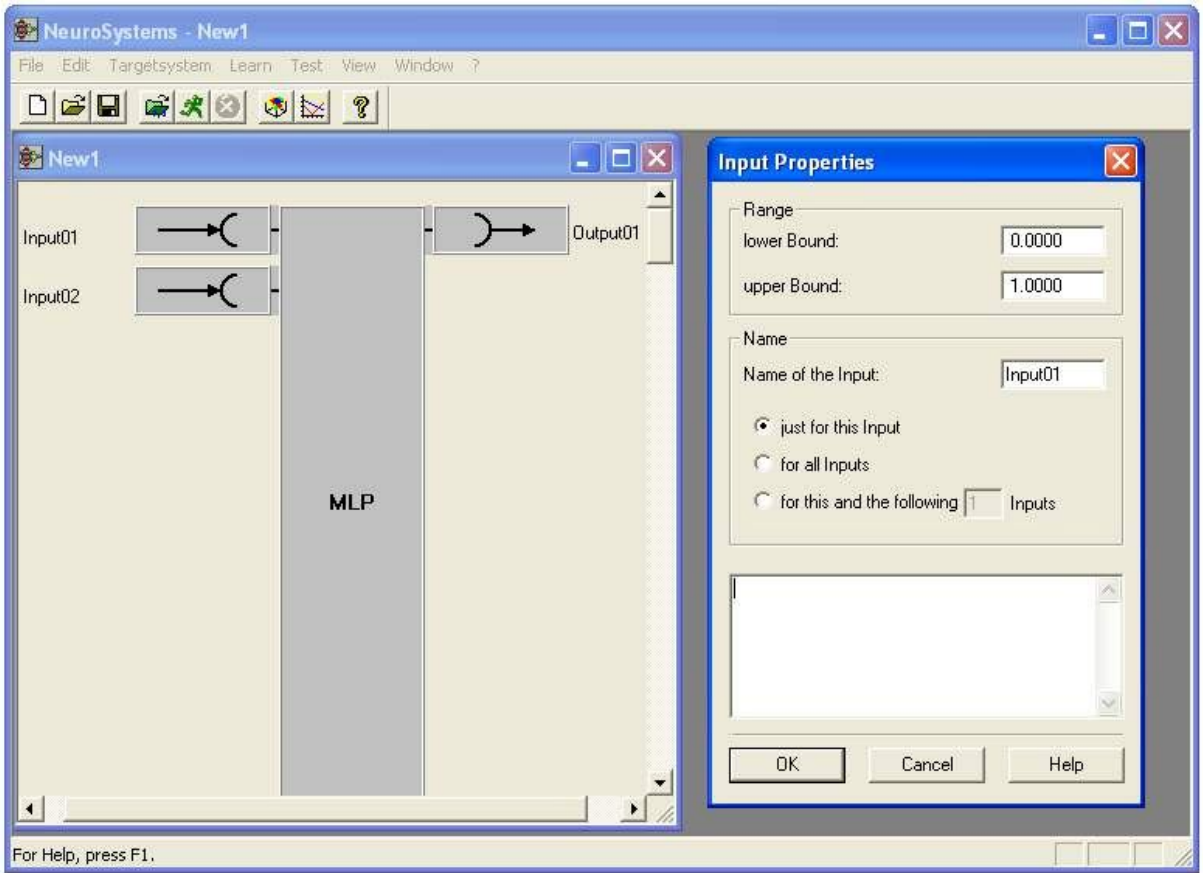
http://chipsystem.ru/

Для демонстрации работы с **NEUROSYSTEMS**, приводиться очень простой пример: Предположим, контроллер должен выполнить операцию, исключающую **ИЛИ**, т. е. функцию **XOR**, для двух входных сигналов. Данные для обучения берутся из таблицы истинности функции **XOR**:

Tags:	Inputs		Outputs
	I0	I1	O0
Values:	0	0	0
	0	1	1
	1	0	1
	1	1	0

После запуска программы, создаем новый проект, где указываем, сколько входов и выходов нам нужно, то есть два входа и один выход.

После начальных настроек, открывается блок схема нейронной сети, по умолчанию **MLP**. Заходим в настройку входа **Input01**. Нормализуем его, установив верхнюю - один, и нижнюю границу – ноль, для входного сигнала. Тоже проделаем и со вторым входом и выходом.



Еще нужно выбрать тип сети.

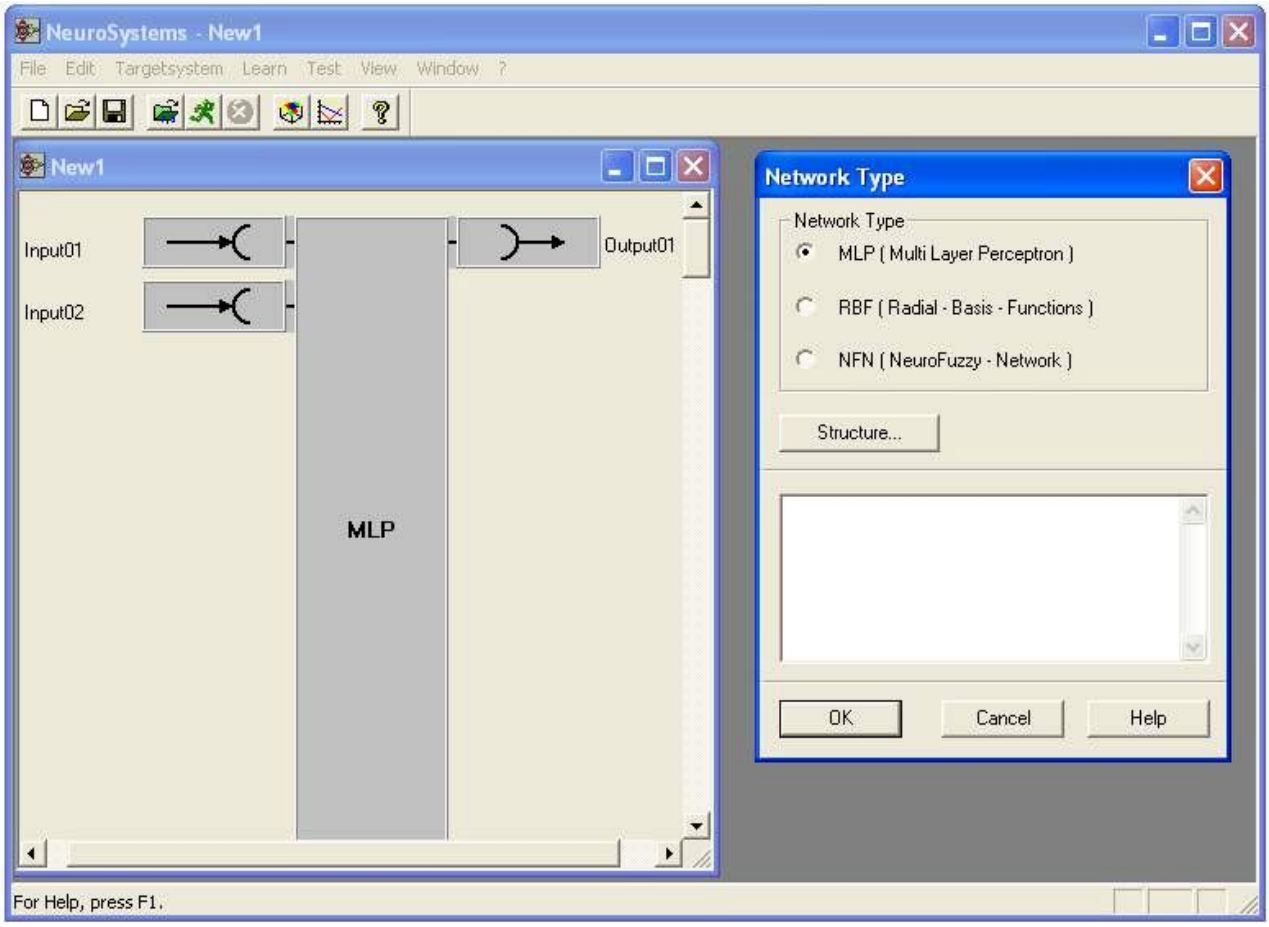
NEUROSYSTEMS предоставляет следующие типы сетей:

• **MLP** сеть (многослойный персептрон)
Этот тип сети подходит, если только относительно небольшой объем обучающих данных.

• **RBF** сеть (радиально базисные функции)
Этот тип сети должен использоваться, если много данных для обучения сети.

• **NFN** сеть (**neurofuzzy**)
Сети этого типа обрабатывают функции принадлежности и базу знаний нечеткой логики.

По умолчанию устанавливается нейронная сеть **MLP** (многослойный персептрон).

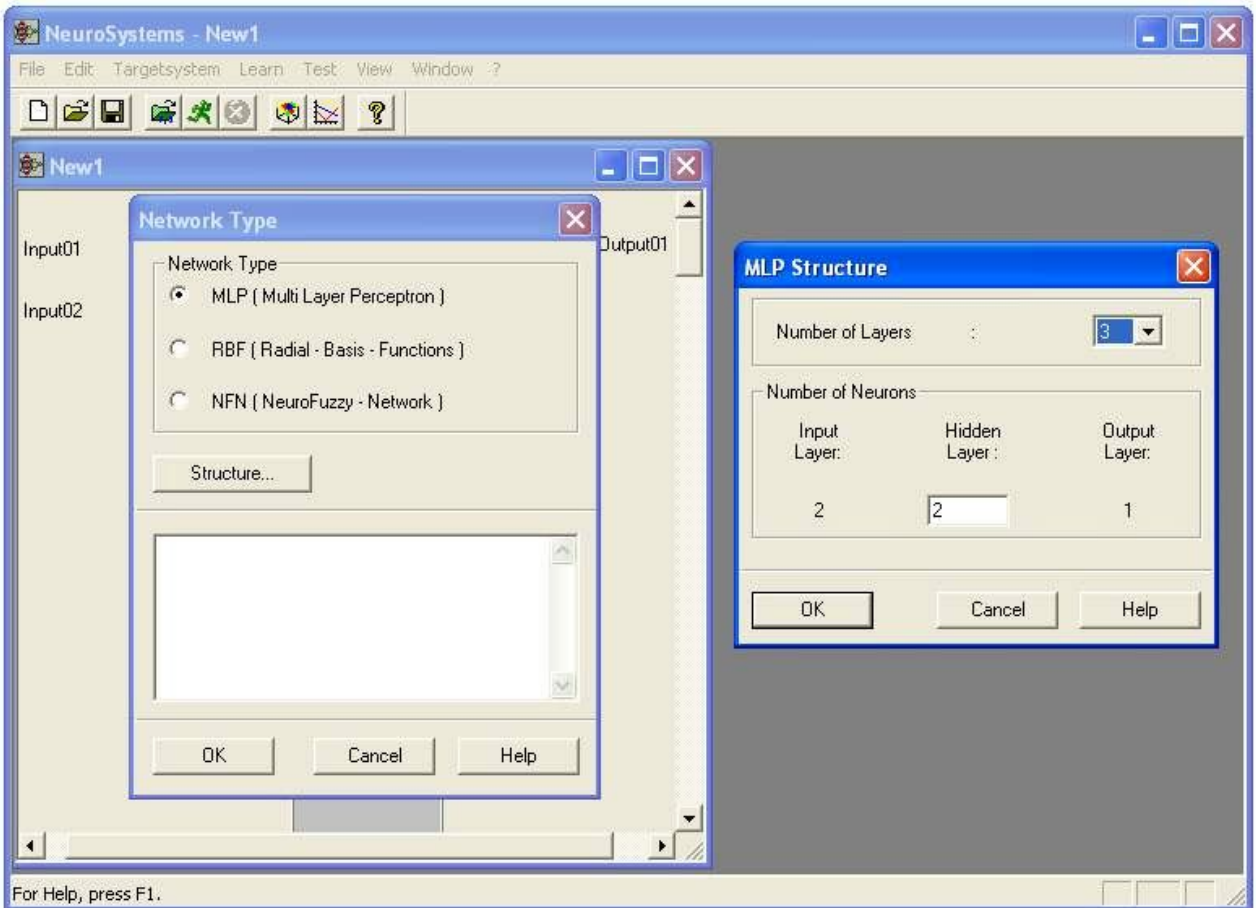


Далее настраиваем структуру нейронной сети. Структура сети определяется количеством слоев и число нейронов в слоях (для **MLP** и **RBF**), или числом функций принадлежности у входов и выходов (для **NFN**). Входной и выходной слои сети подключаются к внешним устройствам, поэтому число нейронов в этих слоях обязательно совпадает с числом входов и выходов, определенных в проекте. Количество скрытых слоев может быть выбрано в зависимости от типа сети. Для сети **MLP**, вы можете выбрать один или два слоя. Для каждого скрытого слоя, вы можете назначить от 1 до 50 нейронов.
Для выбора структуры сети полезна следующая информация:

• Чем меньше данных для обучения, тем меньше нейронов, вы должны выбрать.

• Начните с малого количества нейронов, и увеличивайте их количество, пока не получите удовлетворительный результат.

• По мере усложнения функции, которая будет моделировать нейронные сети, число слоев нейронов также должна увеличиваться.



Процесс обучения

Обучение сети означает выполнение шагов обучения до получения удовлетворяющего результата.

Каждый шаг обучения включает в себя:

• Расчет текущей, фактической на выходе от входного образца;

• Определение текущей разницы между фактической и заданной модели;

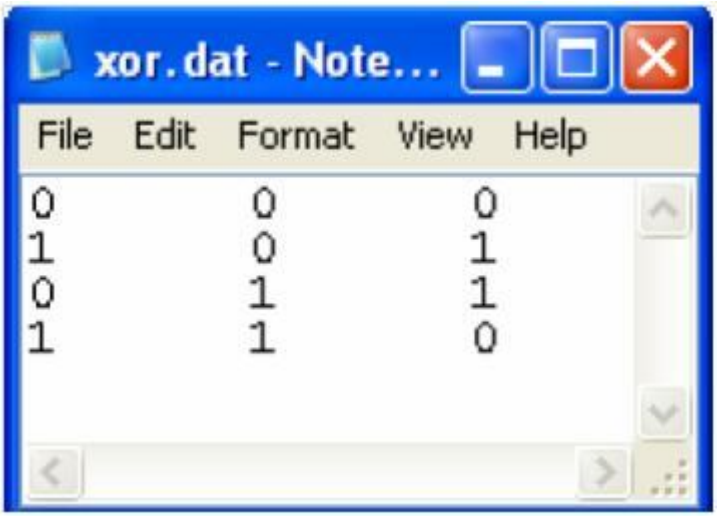
• Сравнение с отличием от предыдущего шага;

• Изменение подключение весов нейронов таким образом, что разница между двумя отличиями становилась меньше.

Набор связанных данных входной и выходной модели называют обучающим набором данных. Здесь, сеть должна научиться поведению логической функции **XOR** с помощью таблицы истинности.

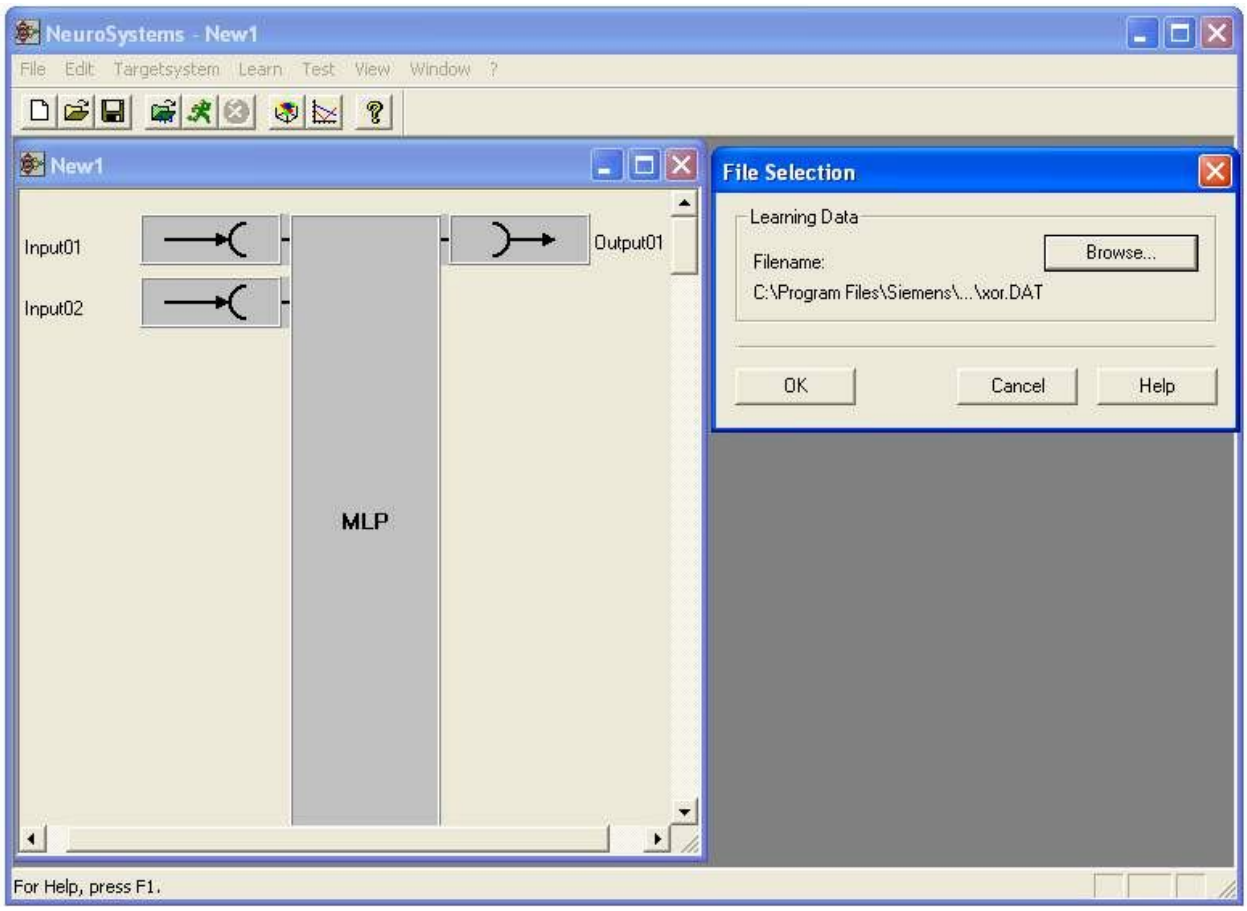
Time k ⇓	Input pattern Input vector		⇒	Output pattern Output vector	
0	**0**	**0**		**0**	1st learning pair
1	**1**	**0**	⇒	**1**	2nd learning pair
2	**0**	**1**		**1**	3rd learning pair
3	**1**	**1**	⇒	**0**	4th learning pair
Designation	Input signal 1 (*Input01*)	Input signal 2 (*Input02*)	⇒	Output signal (*Output01*)	Learning pair, learning data set

В блокноте составим таблицу истинности и сохраним с расширением .**DAT**.



Теперь подключим его к проекту.

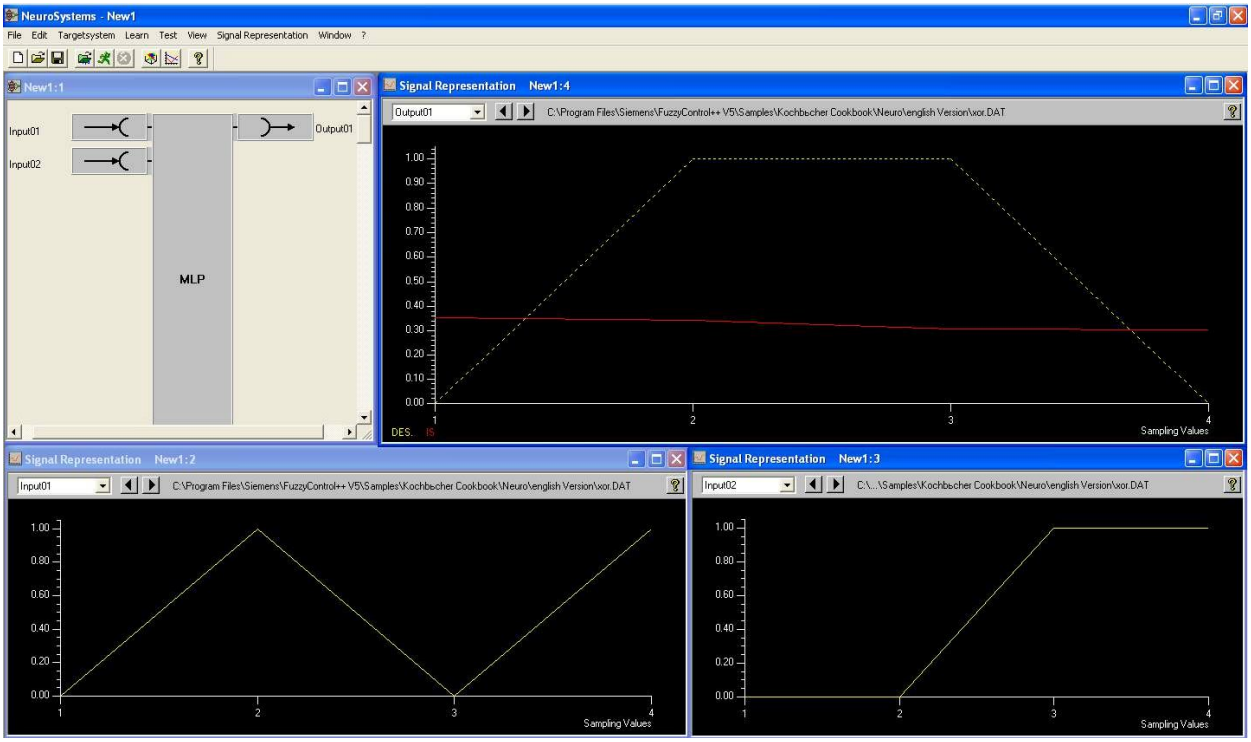
http://chipsystem.ru/



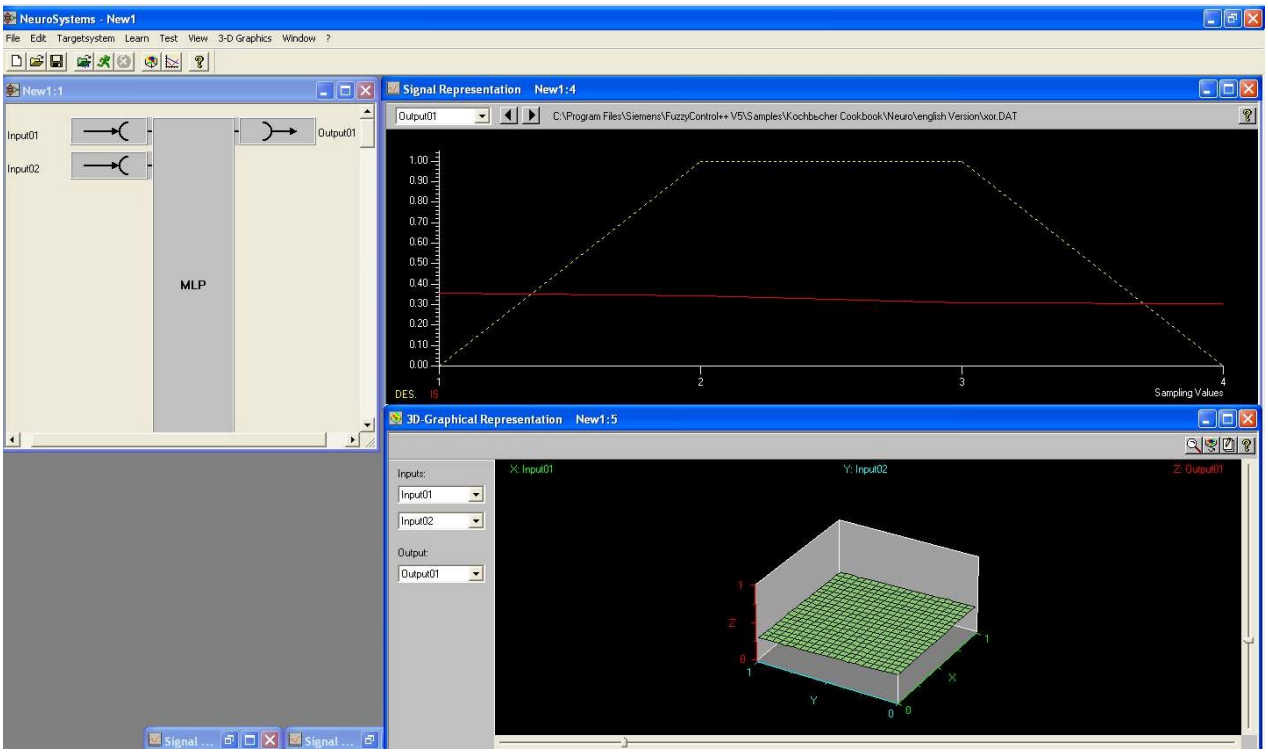
Посмотрим на учебный набор данных в графическом виде. На рисунке два нижних графика – входные данные, а верхний график – выходные данные.

Четыре значения каждой переменной (входа или выхода) могут быть соединены прямой линией, и отображается в виде кривой сигнала на оси времени. Это интерпретирует изучения данных таким образом, что четыре этапа обучения появится в сети один за другим ***к*** раз.

http://chipsystem.ru/

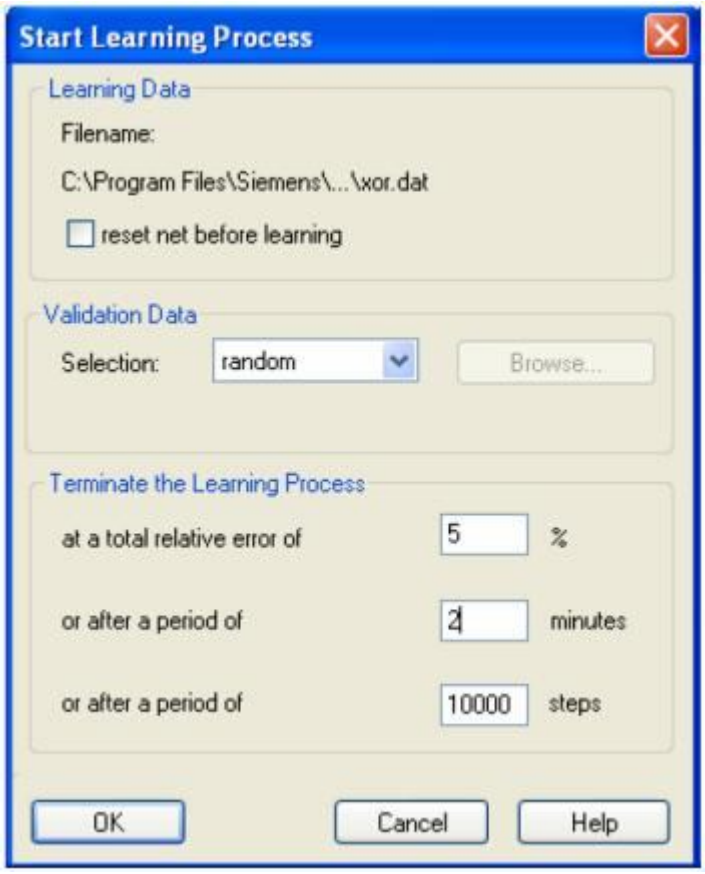


На графике выходного сигнала, трапециевидная кривая – желаемый результат, а красная кривая (почти горизонтальная линия) фактический сигнал на выходе не обученной сети.

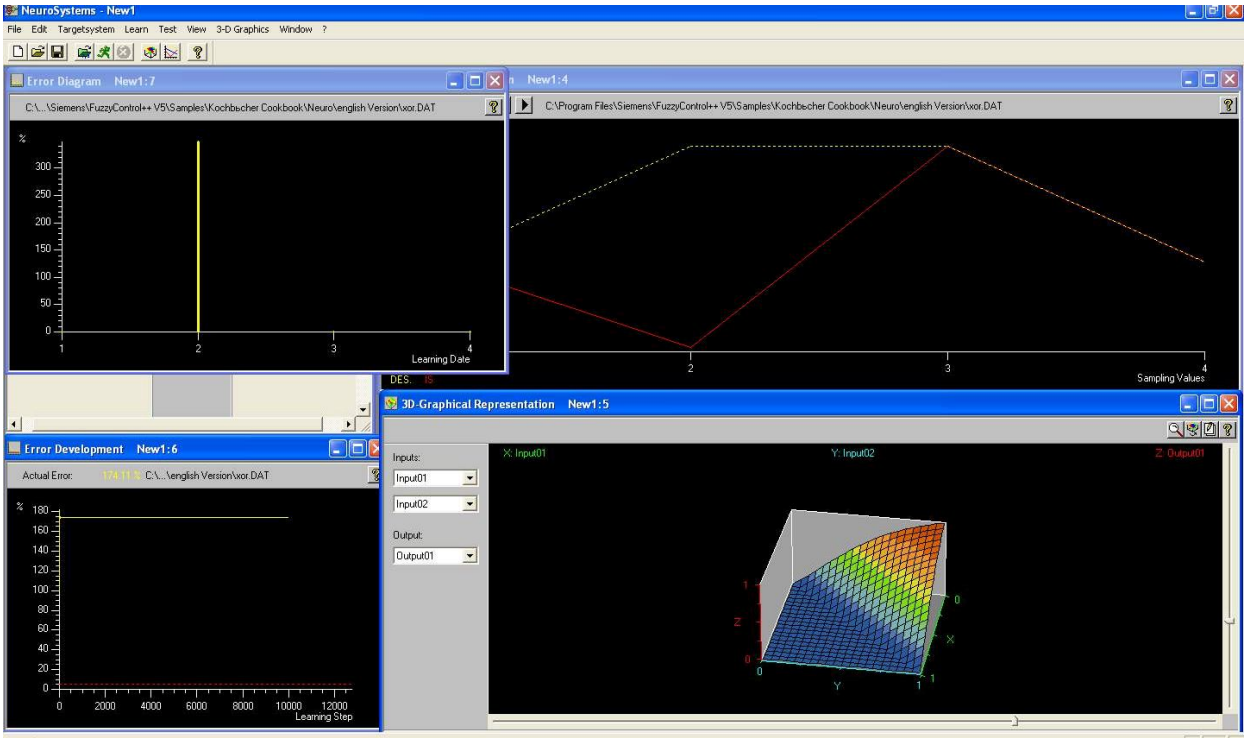


Откроем 3-D дисплей. Выходной сигнал – ось **Z**, входные сигналы – ось **X, Y**. На рисунке изображен график зависимости выхода **Z** от входов **X**,**Y** у не обученной сети.
Теперь обучим сеть с помощью таблицы в файле **xor.DAT**.
При запуске обучения нужно установить предел погрешности и время обучения.
Проверку данных оставим **RANDOM**(случайный выбор).

http://chipsystem.ru/

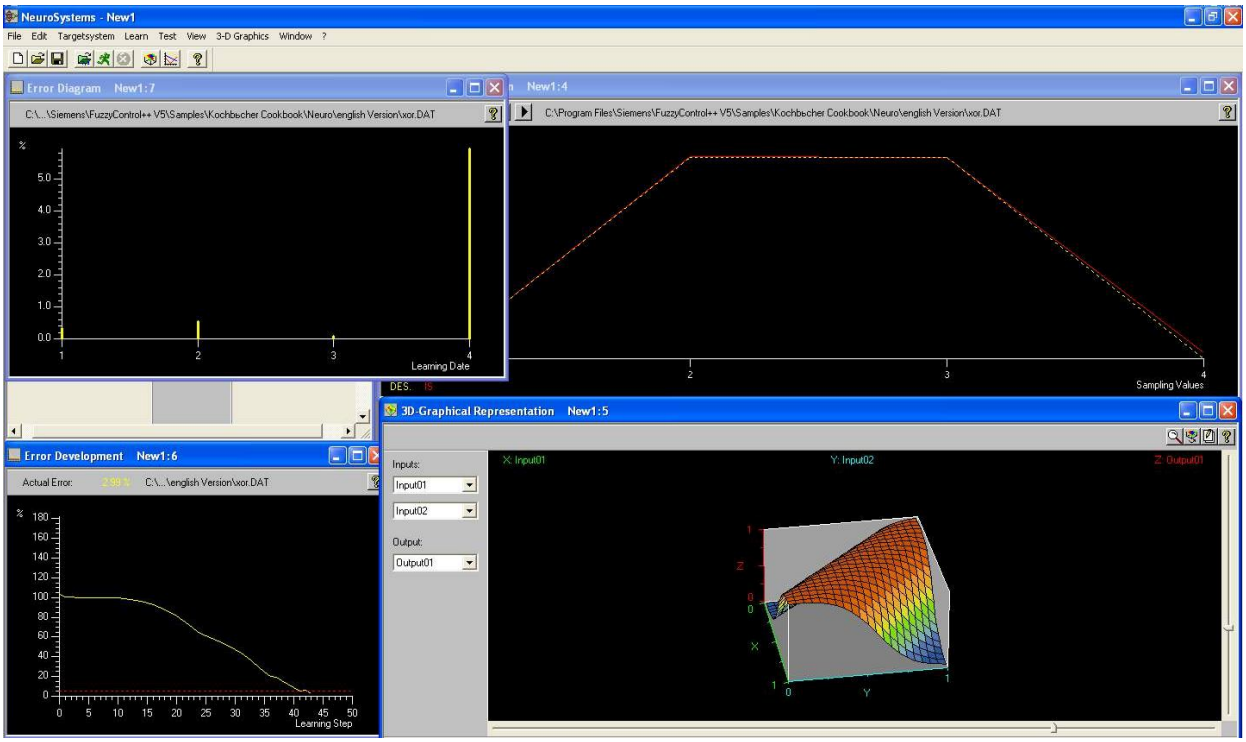


Начальный процесс обучения является неудачным и останавливается после прохождения изучения 10000 шагов.
В процессе обучения ошибка не достигает допустимой погрешности. (левый нижний график).

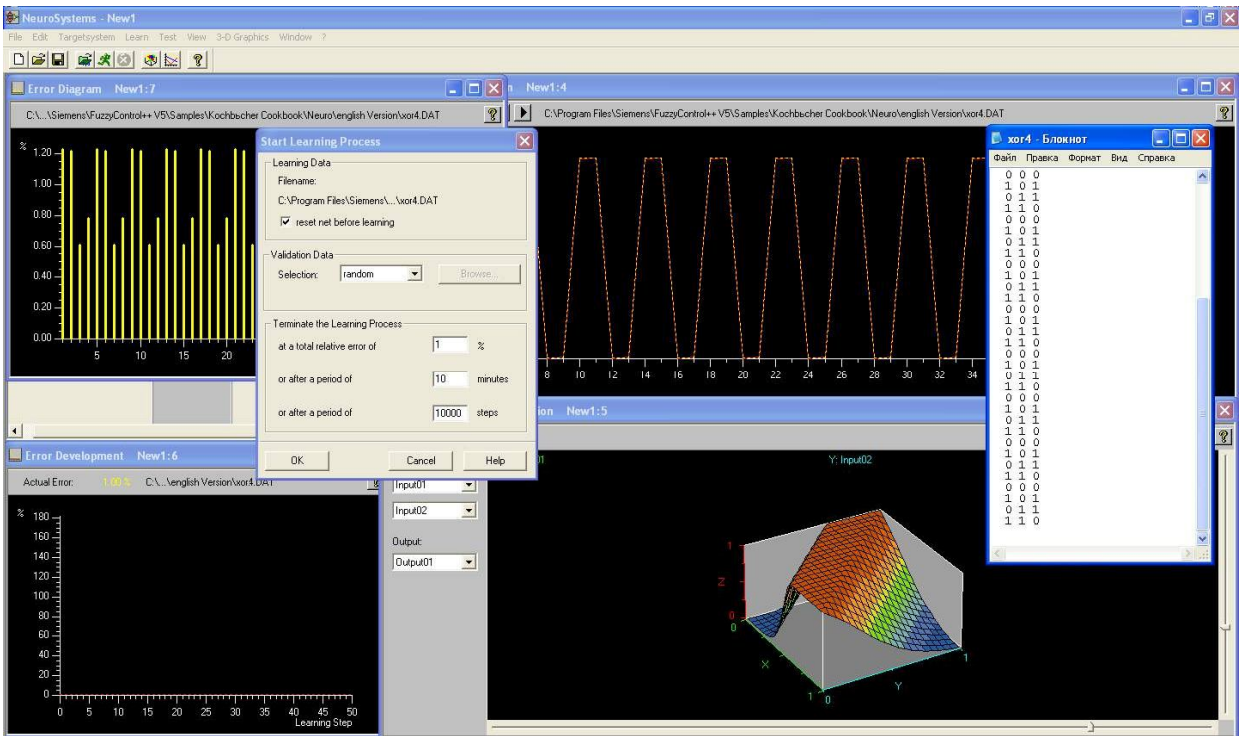


http://chipsystem.ru/

Обучение был неудачным, потому что случайная выборка проверки данных (**RANDOM**) является непригодным методом обучения для этой сети, так как крайне мал файл с обучающими данными. Во второй попытке, мы будем использовать **none** в проверке данных.

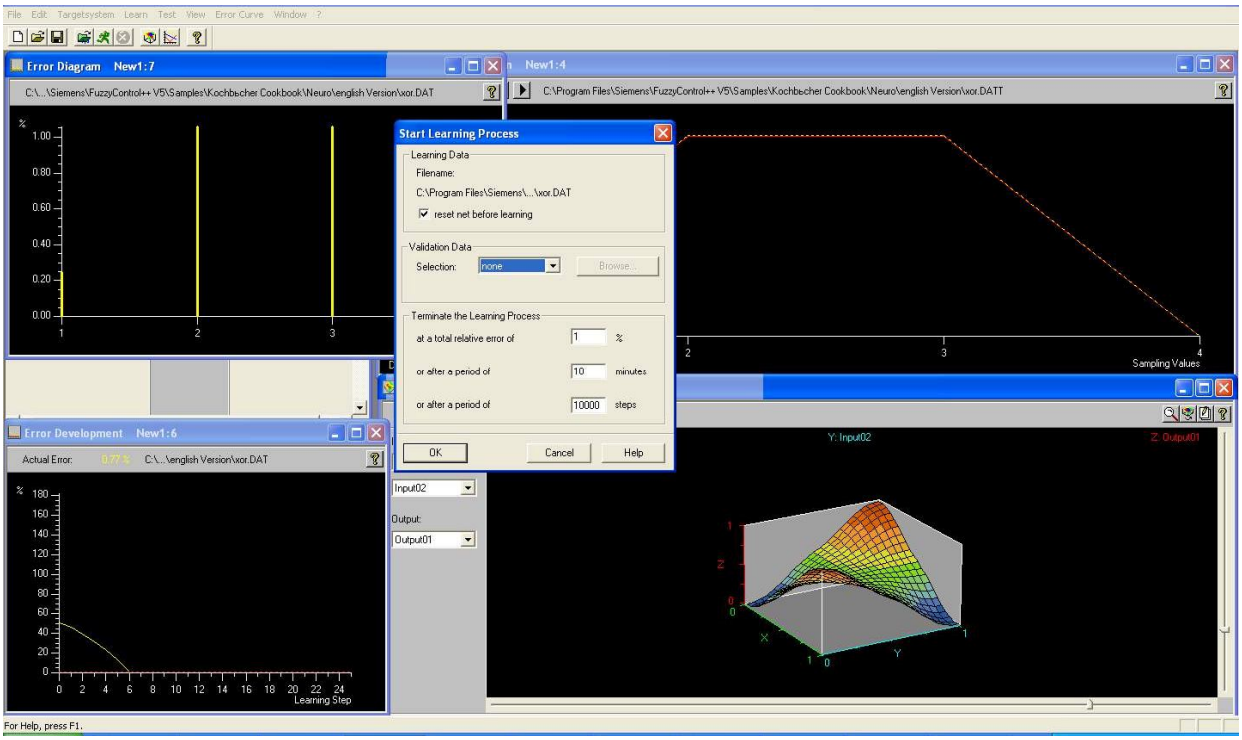


Процесс обучения достигает предела ошибки и завершается успешно после сорока шагов обучения.
А теперь ради эксперимента я увеличил размер обучающих данных простым копированием и при запуске обучения установил опять проверку данных **RANDOM**(случайный выбор).

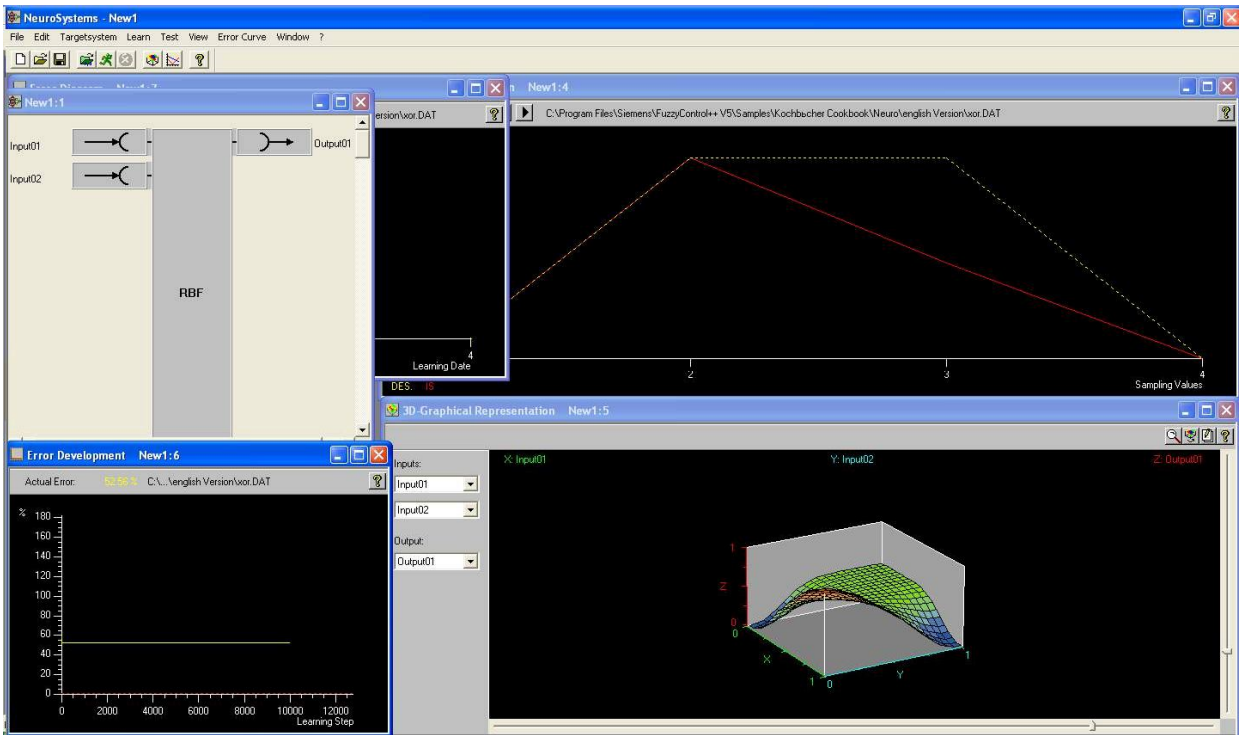


http://chipsystem.ru/

Обучение прошло успешно. Поменяв сеть на **RBF**, получил похожий результат.

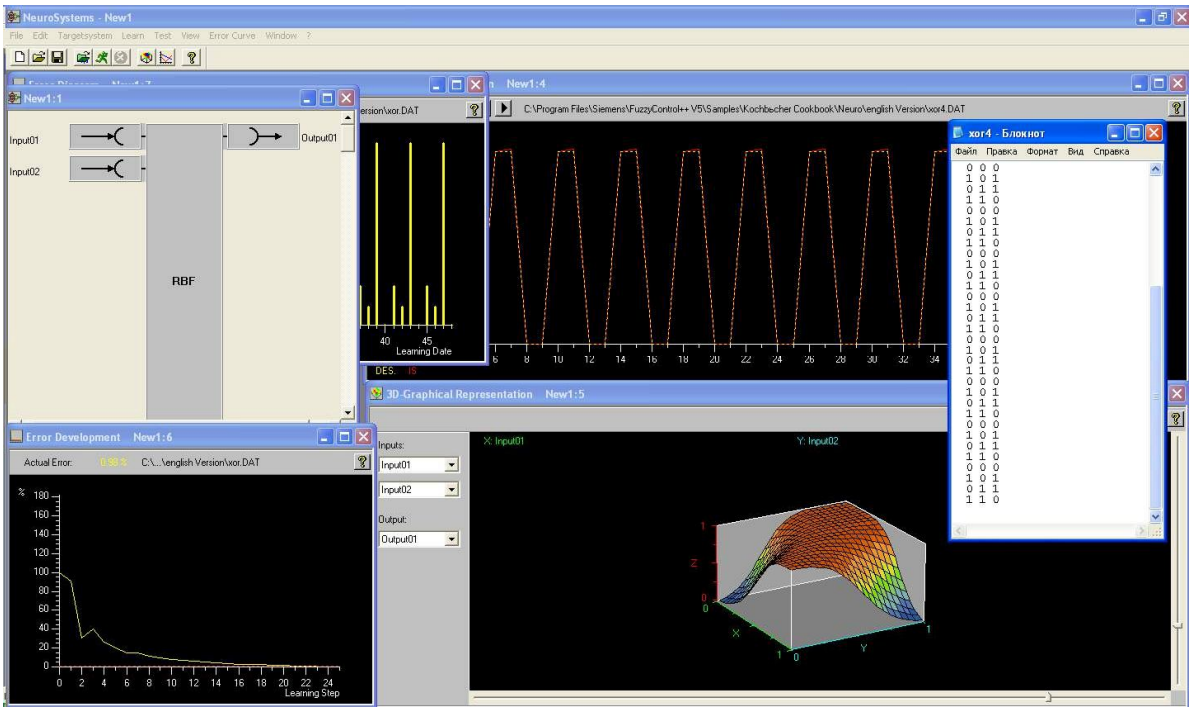


Успешное обучение. Из левого нижнего графика видно, что эта сеть **RBF** обучается за шесть шагов.

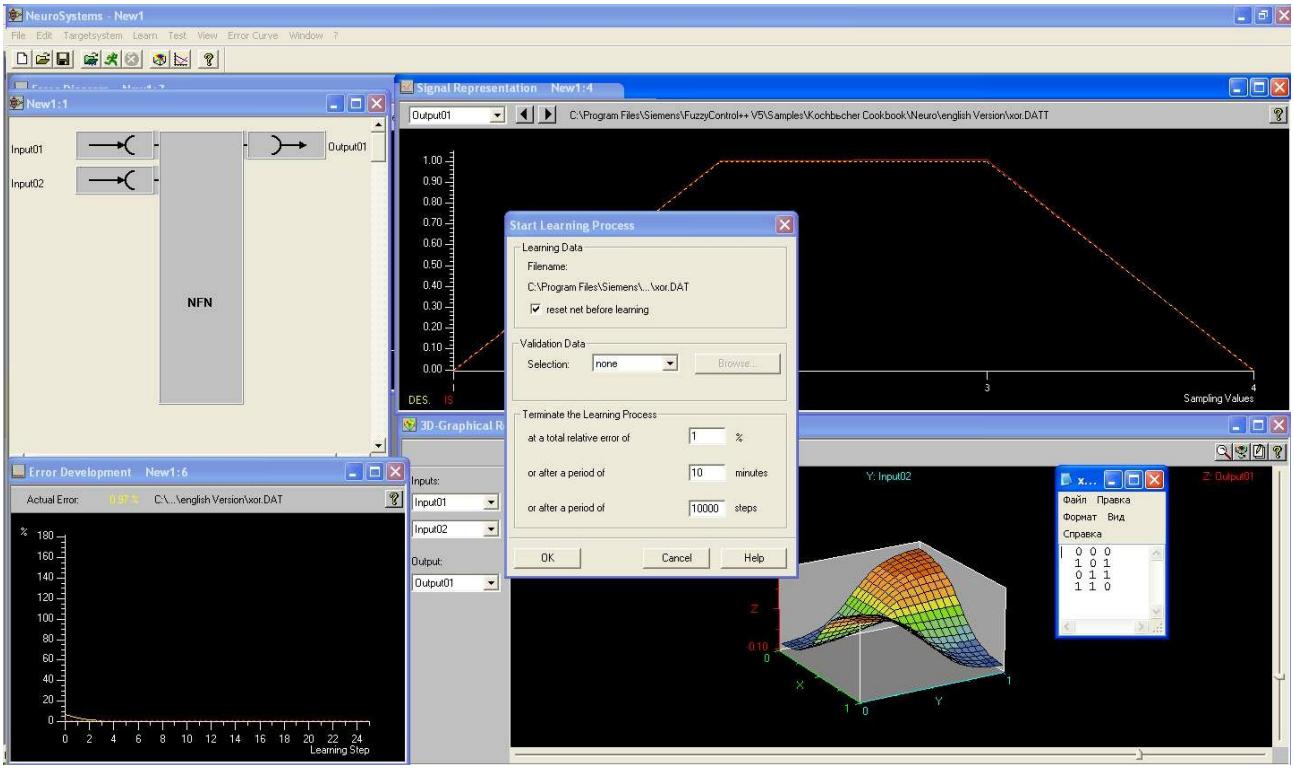


И не успешное обучение.

http://chipsystem.ru/

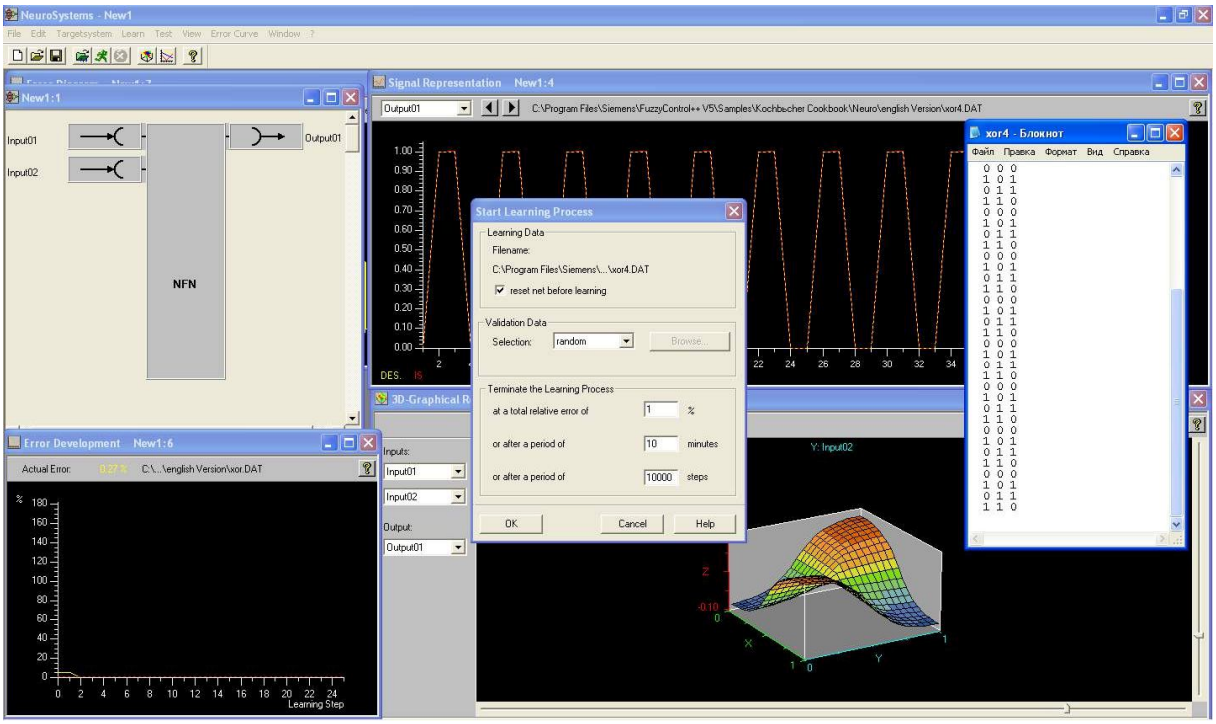


С большим количеством обучающих данных сеть **RBF** так же обучилась за меньшее количество шагов.

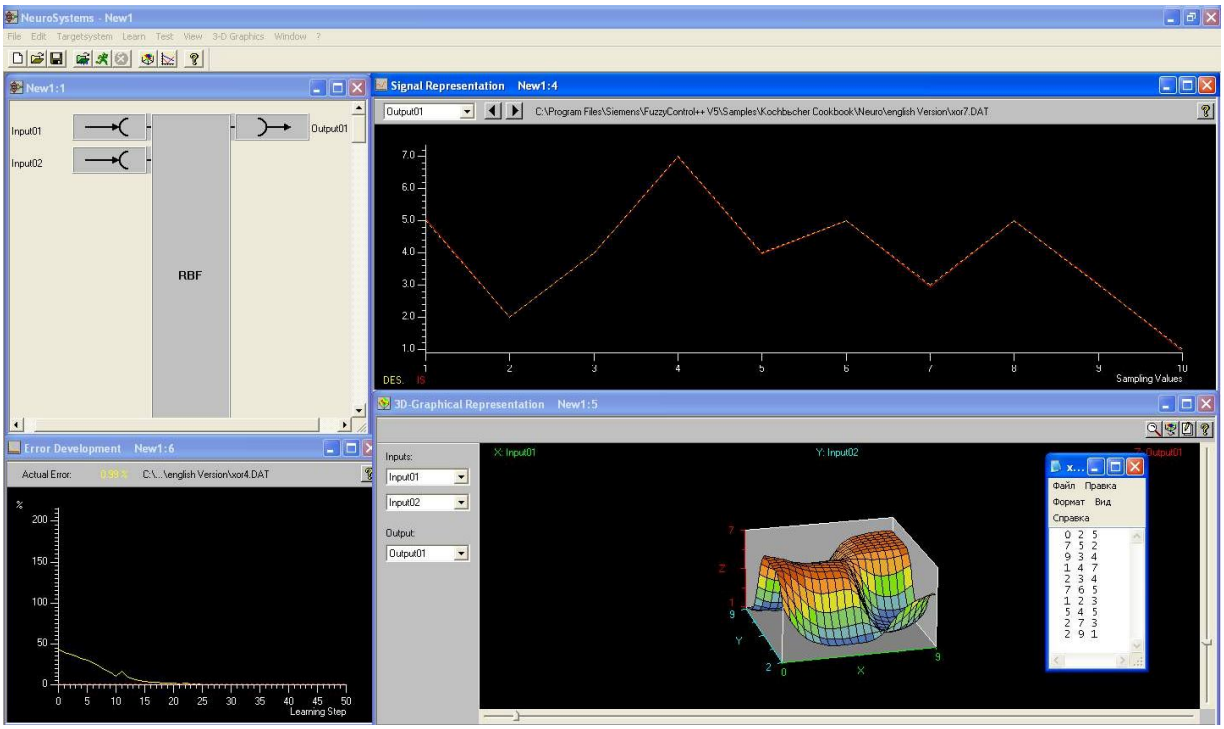


Сеть **NFN** еще быстрее обучается, почти за два шага.

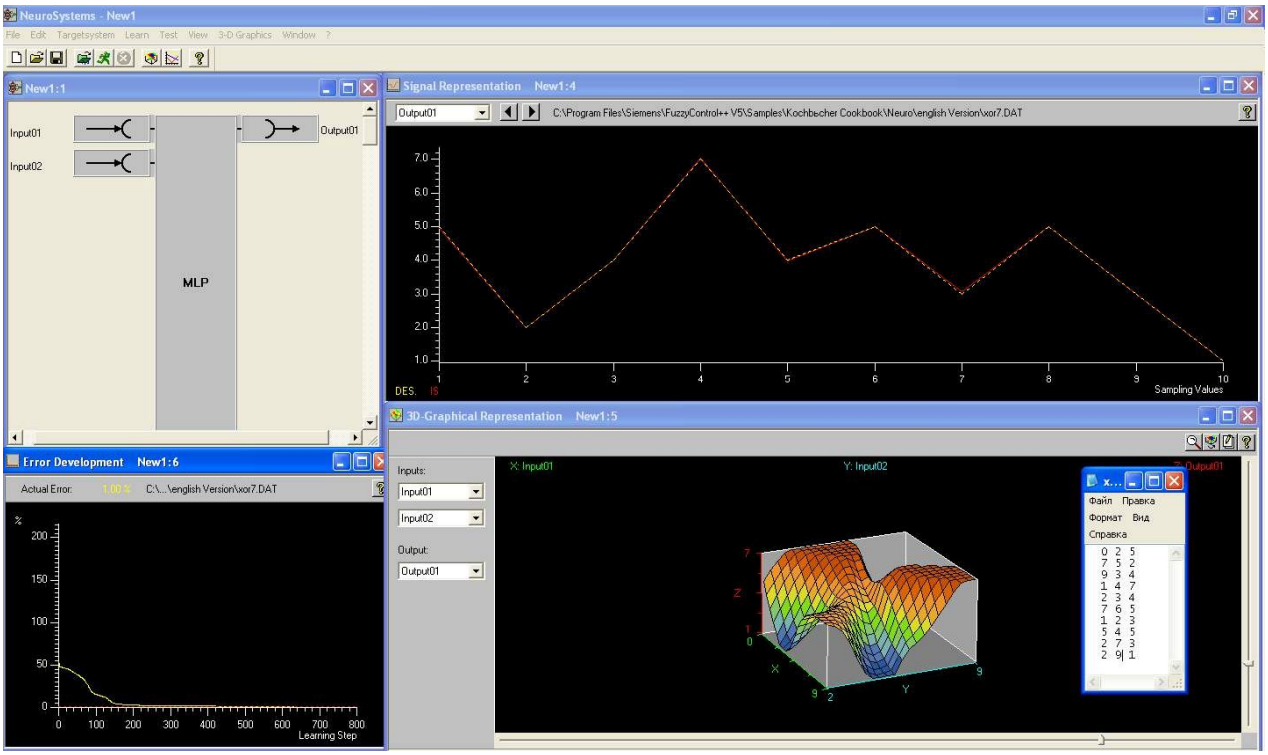
http://chipsystem.ru/



Попробовал создать хаотичные обучающие данные и попробовал их на разных сетях, меняя количество нейронов. Получился довольно неплохой результат.



http://chipsystem.ru/



Только сеть **NFN** не захотела обучаться с этими данными.

Скачать ПО можно с сайта фирмы **SIEMENS**.
http://www.industry.siemens.com/services/global/en/IT4Industry/products/process_control/neurosystems/Pages/default_tab.aspx?tabcardname=Downloads