Федеральное государственное автономное
образовательное учреждение
высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Рубан
подпись
« _____» _____ 2016 г.

ДИПЛОМНЫЙ ПРОЕКТ

230105.65 Программное обеспечение вычислительной техники и автоматизированных систем

Построение искусственной нейронной сети для решения задач планирования объемов продаж на предприятии

Пояснительная записка

Научный руководитель _____ доцент, к.ф.м. н. В.А. Красиков
подпись, дата

Нормоконтролер _____ доцент, к.т.н. О.А. Антамошкин
подпись, дата

Выпускник _____ И.С. Романов
подпись, дата

Красноярск 2016Федеральное государственное автономное
образовательное учреждение
высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Рубан
подпись
« _____» _____ 2016г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме дипломного проекта

Студенту Романову Игорю Сергеевичу

Группа ЗКИ10-15 Направление (специальность) 230105.65, Программное обеспечение вычислительной техники и автоматизированных систем.

Тема выпускной квалификационной работы: «Построение искусственной нейронной сети для решения задач планирования объемов продаж на предприятии ».

Утверждена приказом по университету № 4202/с от 28.03.2016 г.
Руководитель ВКР В.А. Красиков, доцент кафедры «Информатика», канд. Физ.мат. наук.

Перечень разделов ВКР:

- введение;
- общие сведения;
- описание работы приложения;
- техническая реализация;
- эргономика.

Перечень графического или иллюстративного материала с указанием основных чертежей, плакатов, слайдов:презентационные слайды PowerPoint.

Руководитель ВКР _____ В.А. Красиков
(подпись)

Задание принял к исполнению И.С. Романов
(подпись)

« ___ » _____ 2016 г.

АННОТАЦИЯ

Выпускная квалификационная работа по теме «Построение искусственной нейронной сети для планирования объемов продаж на предприятии» содержит 41 страница текстового документа, 9 рисунков, 13 библиографических источников.

Целью работы являлась разработка приложения для работы с нейронной сетью в учетной системе 1С: Предприятие в виде внешней обработки. Модуль позволяет создавать и редактировать искусственную нейронную сеть, а так же получать предполагаемые результаты вычислений, основываясь на исходных данных, полученных из учетной системы.

В дипломный проект входит введение, четыре главы и заключение.

Во введении определяется необходимость реализации и основные задачи проекта.

В первой главе более подробно описана доступная теория, связанная с разработкой приложения.

Во второй главе идет описание приложения, основных понятий и возможностей приложения.

В третьей главе дается описание системы, архитектуры приложения, обоснование выбора языка реализации и функциональный состав симулятора.

В четвертой главе затронута тема эргономики приложения.

Заключение посвящено подведению итогов по всей проделанной работе.

СОДЕРЖАНИЕ

Введение.....
1Глава. Общие сведения.....
1.1 Перцептроны.....
1.1.1Однослойный перцептрон.....
1.1.2Многослойный перцептрон.....
1.2 Понятие экономического прогнозирования и его сущность.....
1.2Прогнозирование с помощью нейронных сетей.....
11.3 Процесс обучения и методы предобработки данных в нейросетях.
.....
2 Глава. Описание приложения.....
2.1 Возможности приложения.....
2.2 Запуск приложения.....
2.3 Описание работы приложения.....
2.3.1 Формулировка проблемы.....
2.3.2 Сбор информации.....

ВВЕДЕНИЕ

В настоящее время все больше возрастает необходимость в системах, которые способны не только выполнять однажды запрограммированную последовательность действий над заранее определенными данными, но и способны сами анализировать вновь поступающую информацию, находить в ней закономерности, производить прогнозирование и т.д.

В этой области приложений самым лучшим образом зарекомендовали себя так называемые нейронные сети – самообучающиеся системы, имитирующие деятельность человеческого мозга.

Целью данной дипломной работы является разработка приложения, построенного с применением нейросетевого подхода для решения задачи прогнозирования объема реализации продукции предприятия.

Для достижения поставленной цели необходимо решить ряд задач:

1. Дать основные понятия прогнозирования, понятие нейронных сетей, описать принципы их работы;

2. Охарактеризовать методы прогнозирования;

3. Получить прогноз объема продаваемых товаров.

Объектом изучения является инструмент нейронных сетей как особый метод прогнозирования экономических показателей.

1 Глава. Общие сведения

1.1 Перцептроны

1.1.1 Однослойный перцептрон

Одной из первых искусственных сетей, способных к восприятию и формированию реакции на воспринятый стимул, явился PERCEPTRON Розенблатта (F.Rosenblatt, 1957). Перцептрон рассматривался его автором не как конкретное техническое вычислительное устройство, а как модель работы мозга. Простейший классический перцептрон содержит нейроподобные элементы трех типов [рис.1]

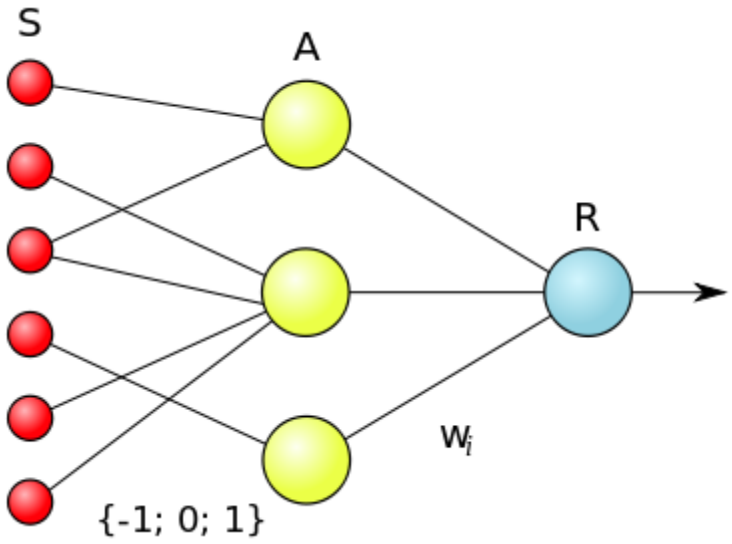

Рис. 1 Схема элементарного перцептрона

S-элементы формируют сетчатку сенсорных клеток, принимающих двоичные сигналы от внешнего мира. Далее сигналы поступают в слой ассоциативных или A-элементов. Только ассоциативные элементы, представляющие собой формальные нейроны, выполняют нелинейную обработку информации и имеют изменяемые веса связей. R-элементы с фиксированными весами формируют сигнал реакции перцептрона на входной стимул.

A-элементы называются ассоциативными, потому что каждому такому элементу, как правило, соответствует целый набор (ассоциация) S-элементов. A-элемент активизируется, как только количество сигналов от S-элементов на

его входе превысило некоторую величину θ. Таким образом, если набор соответствующих S-элементов располагается на сенсорном поле в форме буквы «Д», A-элемент активизируется, если достаточное количество рецепторов сообщило о появлении «белого пятна света» в их окрестности, то есть A-элемент будет как бы ассоциирован с наличием/отсутствием буквы «Д» в некоторой области [10].

Сигналы от возбудившихся A-элементов, в свою очередь, передаются в сумматор R, причём сигнал от i-го ассоциативного элемента передаётся с коэффициентом w_i . Этот коэффициент называется весом A—R связи. Так же как и A-элементы, R-элемент подсчитывает сумму значений входных сигналов, помноженных на веса (линейную форму). R-элемент, а вместе с ним и элементарный перцептрон, выдаёт «1», если линейная форма превышает порог θ, иначе на выходе будет «−1». Математически, функцию, реализуемую R-элементом, можно записать так:

$$f(x) = sign\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

Обучение элементарного перцептрона состоит в изменении весовых коэффициентов w_i связей A—R. Веса связей S—A (которые могут принимать значения {−1; 0; +1}) и значения порогов A-элементов выбираются случайным образом в самом начале и затем не изменяются

1.1.2 Многослойный перцептрон

Многослойный перцептрон представляет собой сеть, состоящую из нескольких последовательно соединенных слоев формальных нейронов МакКаллока и Питтса. [Рис2.]

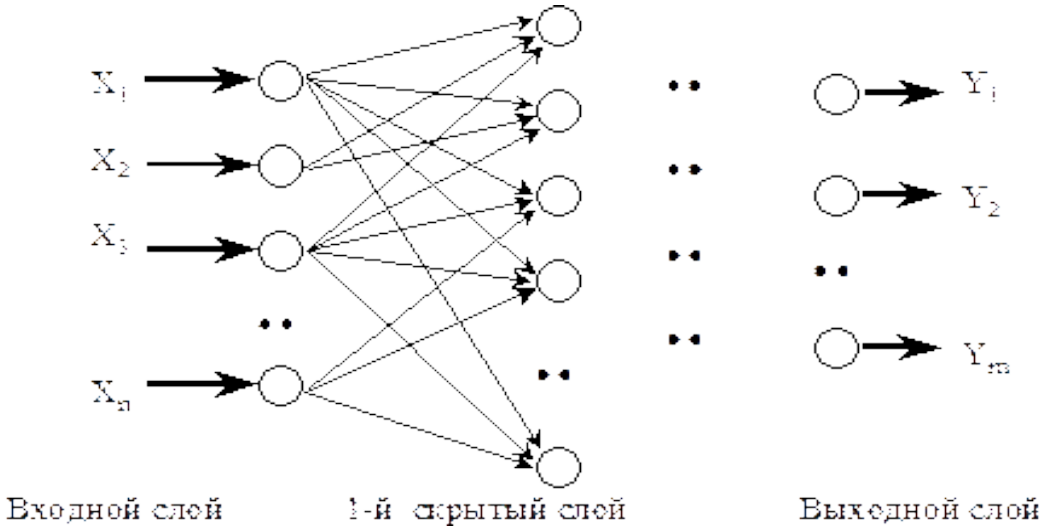

Рис. 2 Модель многослойного перцептрона

На низшем уровне иерархии находится входной слой, состоящий из сенсорных элементов, задачей которого является только прием и распространение по сети входной информации. Далее имеются один или, реже, несколько скрытых слоев. Каждый нейрон на скрытом слое имеет несколько входов, соединенных с выходами нейронов предыдущего слоя или непосредственно со входными сенсорами X1..Xn, и один выход. Нейрон характеризуется уникальным вектором весовых коэффициентов w. Веса всех нейронов слоя формируют матрицу, которую мы будем обозначать V или W. Функция нейрона состоит в вычислении взвешенной суммы его входов с дальнейшим нелинейным преобразованием ее в выходной сигнал:

$$y = 1 \Big/ \left(1 + \exp(-[\sum_i W_i x_i - \Theta])\right)$$

Многослойный перцептрон способен рассчитать выходное значение Y для входного значения X. Другими словами, сеть вычисляет значение некоторой векторной функции: Y = F(X).

Таким образом, условие задачи, которая ставится перцептрону должно быть сформулировано в виде множества векторов {x1, ..., xS}. Решение задачи будет представлено в виде векторов {y1, ..., yS}, причем для s ys = F(xs). Все, что способен сделать перцептрон, –сформировать отображение F: X ® Y для"x O X. Мы не можем «извлечь» из перцептрона данное отображение полностью, а можем только посчитать образы произвольного числа точек. Задача формализации, то есть выбора смысла, которым наделяются компоненты входного и выходного векторов, решается человеком на основе практического опыта. К сожалению, жестких рецептов формализации для нейронных сетей пока не создано. Чтобы построить многослойный перцептрон, необходимо выбрать его параметры по следующему алгоритму :

- Определить, какой смысл вкладывается в компоненты входного вектора X. Входной вектор должен содержать формализованное условие задачи, то есть всю информацию, необходимую для того, чтобы получить ответ.Выбрать выходной вектор Y таким образом, чтобы его компоненты содержали полный ответ для поставленной задачи.Выбрать вид функции активации нейронов. При этом желательно учесть специфику задачи, так как удачный выбор увеличит скорость обучения.Выбрать количество слоев и нейронов в слое.
- Задать диапазон изменения входов, выходов, весов и пороговых уровней на основе выбранной функции активации.
- Присвоить начальные значения весам и пороговым уровням. Начальные значения не должны быть большими, чтобы нейроны не оказались в насыщении (на горизонтальном участке функции активации), иначе обучение будет очень медленным. Начальные значения не должны быть и слишком малыми, чтобы выходы большей части нейронов не были равны нулю, иначе обучение тоже замедлится.

- Провести обучение, то есть подобрать параметры сети так, чтобы задача решалась наилучшим образом. По окончании обучения сеть сможет решать задачи того типа, которым она обучена.
- Подать на вход сети условия задачи в виде вектора X. Рассчитать выходной вектор Y, который и даст формализованное решение задачи [11].

1.2 Понятие экономического прогнозирования и его сущность

Любой процесс прогнозирования, как правило, строится в следующей последовательности:

1. Формулировка проблемы.

2. Сбор информации и выбор метода прогнозирования.

3. Применение метода и оценка полученного прогноза.

4. Использование прогноза для принятия решения.

5. Анализ «прогноз-факт». Все начинается с корректной формулировки проблемы. В зависимости от нее задача прогнозирования может быть сведена, например, к задаче оптимизации.

Прогнозирование — это научное определение вероятных путей и результатов предстоящего развития экономической системы и оценка показателей, характеризующих это развитие в более или менее отдаленном будущем [6, с.17]. Целью прогнозирования является уменьшение риска при принятии решений. В б ольшинстве случаев прогноз получается ошибочным, причем ошибка зависит от прогнозирующей системы и методов прогнозирования. Для уменьшения ошибки следует увеличивать количество ресурсов предоставляемых для прогноза. При некотором уровне ошибки возможно добиться минимального уровня ресурсов для прогноза [1,с.28]. Основной проблемой прогнозирования является выявление неточности прогноза. Обычно, решение, принимаемое на основании прогноза должно учитывать ошибку, о которой сообщает система прогнозирования. Таким образом, система прогнозирования должна обеспечить определение прогноза и ошибки прогнозирования[2,с.31].Процессы, перспективы которых необходимо предсказывать, чаще всего описываются **временными рядами**, то есть последовательностью значений некоторых величин, полученных в определенные моменты времени. Временной ряд включает в себя два обязательных элемента — отметку времени и значение показателя ряда, полученное тем или иным способом и соответствующее указанной отметке времени

1.2 Прогнозирование с помощью нейронных сетей

Цель анализа временных рядов – извлечь из данного ряда полезную информацию. Для этого необходимо построить математическую модель явления. Такая модель должна объяснять существо процесса, порождающего данные, в частности – описывать характер данных (случайные, имеющие тренд, периодические, стационарные и т.п.). После этого можно применять различные методы фильтрации данных (сглаживание, удаление выбросов и др.) с конечной целью – предсказать будущие значения[3,с.103].

Таким образом, этот подход основан на предположении, что временной ряд имеет некоторую математическую структуру (которая, например, может быть следствием физической сути явления). Эта структура существует в так называемом фазовом пространстве, координаты которого – это независимые переменные, описывающие состояние динамической системы. Поэтому первая задача, с которой придется столкнуться при моделировании – это подходящим образом определить фазовое пространство. Для этого нужно выбрать некоторые характеристики системы в качестве фазовых переменных. После этого уже можно ставить вопрос о предсказании или экстраполяции. Как правило, во временных рядах, полученных в результате измерений, в разной пропорции присутствуют случайные флуктуации и шум. Поэтому качество модели во многом определяется ее способностью аппроксимировать предполагаемую структуру данных, отделяя ее от шума.

Нейронные сети можно рассматривать как обобщение традиционных подходов к анализу временных рядов. Нейронные сети дают дополнительные возможности в моделировании нелинейных явлений и распознавании хаотического поведения. Благодаря своей большой гибкости (на одной топологии можно реализовать много различных отображений), сети могут моделировать в фазовом пространстве работу разнообразных структур.Любая зависимость вида X=() с непрерывной нелинейной функцией f может быть воспроизведена на многослойной сети [12].

Недавние исследования показали, что нейронные сети имеют по сравнению с классическими моделями более высокие потенциальные возможности при анализе сложной динамической структуры и при этом дают лучшие результаты на таких известных типах временных рядов, как стационарные, периодические, трендовые и некоторые другие.

Нейронные сети можно также применять для одномерного и многомерного анализа, должным образом сформировав множество независимых входов и зависящих от них выходов. Как правило, модель строится для того, чтобы предсказывать значения временного ряда для одной целевой переменной, однако, в принципе, модель может предсказывать значения и нескольких переменных (например, доходы по акциям на различное время вперед), если в сеть добавить дополнительные выходные элементы.

В настоящее время продолжаются исследования в области моделирования временных рядов при помощи сетей, но стандартных методов здесь пока не выработано. В нейронной сети многочисленные факторы взаимодействуют весьма сложным образом, и успех здесь пока приносит только эвристический подход [4,с.69].

Действия на первом этапе (предварительной обработки данных) значительно зависят от специфики задачи.

Нужно правильно выбрать число и вид показателей, характеризующих процесс, в том числе – структуру задержек. После этого надо выбрать топологию сети. Если применяются сети с прямой связью, нужно определить число скрытых элементов. В дальнейшем, для нахождения параметров модели выбирают критерий ошибки и оптимизирующий (обучающий) алгоритм. Затем, используя средства диагностики, следует проверить различные свойства модели. Наконец, необходимо интерпретировать выходную информацию [13].

1 1.3 Процесс обучения и методы предобработки данных в нейросетях.

Данные, подаваемые на вход нейросети, должны предварительно обрабатываться с целью получения качественного процесса обучения и соответственно качественных прогнозных значений выходных переменных. Предварительная обработка данных – один из важнейших этапов прогнозирования, так как реальные данные содержат шумы и бывают неравномерно распределены. Рассмотрим следующие этапы процесса предварительной обработки данных:

- сбор данных;
- анализ и очистка данных;
- преобразование данных.

Сбор данных. Самое важное решение, которое должен принять аналитик, – это выбор совокупности переменных для описания моделируемого процесса. Чтобы представить себе возможные связи между разными переменными, нужно хорошо понимать существо задачи. Относительно выбранных переменных нужно понимать, значимы ли они сами по себе, или же в них всего лишь отражаются другие, действительно существенные переменные[5,с.80]. Проверка на значимость включает в себя кросскорреляционный анализ. С его помощью можно, например, выявить временную связь типа запаздывания (лаг) между двумя рядами. То, насколько явление может быть описано линейной моделью, проверяется с помощью регрессии по методу наименьших квадратов.

В целом, можно сказать, что предварительная обработка через формирование совокупности переменных и проверка их значимости существенно улучшает качество модели. Если никаких теоретических методов проверки в распоряжении нет, переменные можно выбирать методом проб и ошибок, или с помощью формальных методов типа генетических алгоритмов[8,с. 51].

Анализ и очистка данных. Стоит начать с того, чтобы изобразить распределение переменной с помощью гистограммы или же рассчитать для

него характеристики асимметрии (симметричность распределения) и эксцесса (весомости «хвостов» распределения). В результате будет получена информация о том, насколько распределение данных близко к нормальному. Многие методы моделирования, в том числе нейронные сети, дают лучшие результаты на нормализованных данных. Выбросы могут порождаться ошибочными данными или крайними значениями, вследствие чего структура связей между переменными может нарушаться. В некоторых приложениях выбросы могут нести положительную информацию, и их не следует автоматически отбрасывать[7,с. 64].

Преобразование данных. Предварительное (до подачи на вход сети) преобразование данных с помощью статистических приемов может существенно улучшить как параметры обучения (длительность, сложность), так и работу системы. Например, если входной ряд имеет отчетливый экспоненциальный вид, то после его логарифмирования получится более простой ряд, и если в нем имеются сложные зависимости высоких порядков, обнаружить их будет гораздо легче. Очень часто не нормально распределенные данные предварительно подвергают нелинейному преобразованию: исходный ряд значений переменных преобразуется некоторой функцией, и ряд, полученный на выходе, принимается за новую входную переменную. Типичными способами преобразования также является возведение в степень, извлечение корня, взятие обратных величин, экспонент или логарифмов. После этого могут быть применены дополнительные преобразования для изменения формы кривой регрессии. Часто это на порядок уменьшает требования к обучению[9,с. 29]. Для того, чтобы улучшить информационную структуру данных, могут оказаться полезными определенные комбинации переменных – произведения, частные и т.д. С помощью таких промежуточных комбинаций можно получить более простую модель .

Обработанные данные подаются на вход нейросети, посредством чего происходит ее обучение.

На этапе обучения происходит вычисление синаптических коэффициентов в процессе решения нейронной сетью задач. Обучение – это процесс, в результате которого система постепенно приобретает способность отвечать нужными реакциями на определенные совокупности внешних воздействий, а адаптация — это подстройка параметров и структуры системы с целью достижения требуемого качества управления в условиях непрерывных изменений внешних условий.

Различают стратегии обучения: «обучение с учителем» и «обучение без учителя».

«Обучение с учителем» предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются обучающей парой. Обычно нейросеть обучается на некотором числе таких обучающих пар. Предъявляется выходной вектор, вычисляется выход нейросети и сравнивается с соответствующим целевым вектором, разность (ошибка) с помощью обратной связи подается в сеть и веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. Векторы обучающего множества предъявляются последовательно, вычисляются ошибки и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

Несмотря на многочисленные прикладные достижения, «обучение с учителем» критиковалось за свою биологическую неправдоподобность. «Обучение без учителя» является намного более правдоподобной моделью обучения в биологической системе. Развитая Кохоненом и многими другими, она не нуждается в целевом векторе для выходов и, следовательно, не требует сравнения с предопределенными идеальными ответами. Обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса нейросети так, чтобы получались согласованные выходные векторы, т. е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Процесс обучения, следовательно, выделяет

статистические свойства обучающего множества и группирует сходные векторы в классы. Предъявление на вход вектора из данного класса даст определенный выходной вектор, но до обучения невозможно предсказать, какой выход будет производиться данным классом входных векторов. Следовательно, выходы подобной сети должны трансформироваться в некоторую понятную форму, обусловленную процессом обучения .

Целью обучения нейронной сети является минимизация функции ошибок, или невязки, E на данном множестве примеров путем выбора значений весов W. Целью процедуры минимизации является отыскание глобального минимума – достижение его называется сходимостью процесса обучения. Поскольку невязка зависит от весов нелинейно, получить решение в аналитической форме невозможно, а поиск глобального минимума осуществляется посредством итерационного процесса – так называемого обучающего алгоритма, который исследует поверхность невязки и стремиться обнаружить на ней точку глобального минимума.

В качестве функции ошибки, численно определяющей сходство всех текущих выходных сигналов сети и соответствующих требуемых выходных сигналов обучающей выборки, в большинстве случаев используется среднеквадратичное отклонение. Однако в ряде нейропакетов существует либо возможность выбора, либо задания своей функции ошибки.

Реализуемые в нейропакетах алгоритмы обучения нейронных сетей можно разделить на три группы: градиентные, стохастические, генетические. Градиентные алгоритмы (первого и второго порядков) основаны на вычислении частных производных функции ошибки по параметрам сети. В стохастических алгоритмах поиск минимума функции ошибки ведется случайным образом. Генетические алгоритмы комбинируют свойства стохастических и градиентных алгоритмов: на основе аналога генетического наследования реализуют перебор вариантов, а на основе аналога естественного отбора – градиентный спуск .

При обучении нейронных сетей, как правило, используются следующие

критерии остановки:

- при достижении некоторого малого значения функции ошибки;
- в случае успешного решения всех примеров обучающей выборки (при неизменности выходных сигналов сети).

Для проверки правильности обучения построенной нейронной сети в нейроимитаторах предусмотрены специальные средства ее тестирования. В сеть вводится некоторый сигнал, который, как правило, не совпадает ни с одним из входных сигналов примеров обучающей выборки. Далее анализируется получившийся выходной сигнал сети.

Тестирование обученной сети может проводиться либо на одиночных входных сигналах, либо на тестовой выборке, которая имеет структуру, аналогичную обучающей выборке, и также состоит из пар («вход», «требуемый выход»). Обычно, обучающая и тестовая выборки не пересекаются. Тестовая выборка строится индивидуально для каждой решаемой задачи .

Из общеизвестных преимуществ нейросетевого подхода следует выделить одно, самое привлекательное в нем – отсутствие необходимости в строгой математической спецификации модели, что особенно ценно при анализе плохо формализуемых процессов.

2 Глава. Описание приложения

2.1 Возможности приложения

Приложение включает в себя следующие возможности:

- Построение и редактирование модели искусственной нейронной сети
- Получение исходных данных из учетной системы
- Вызов справки по работе приложения
- Заполнение первоначальных весов связей нейронов сети
- Возможность запуска обучения нейронной сети фоновым заданием
- Построение графиков, демонстрирующих ход обучения сети
- Вывод предполагаемых результатов непосредственно в обработке либо сохранение в табличный файл

2.2 Запуск приложения

Для запуска приложения используется интерфейс управляемых форм платформы 1С: Предприятие 8.3 Данное приложение имеет 2 варианта запуска

- Как внешний файл обработки из запущенного сеанса пользователя в учетной программе 1С: Предприятие
- Как сохраненный экземпляр объекта конфигурации 1С: Предприятие

2.3 Описание работы приложения

2.3.1 Формулировка проблемы

Прогнозирование - это ключевой момент при принятии решений в управлении. В частности, для принятия решения в отношении запасов товаров в оптово-розничной сети. Зная примерные показатели продаж руководитель принимает решение о закупке прогнозируемых товаров либо о временной приостановке поставок и дальнейшем анализе причин спада продаж. Это позволяет руководителю предприятия оптимально распределять материальные средства, затрачиваемые на закупку ассортимента товаров и потенциально увеличить прибыль от количества наиболее продаваемых товаров и снизить убытки от закупки позиций, временно не пользующихся спросом покупателей.

2.3.2 Сбор информации

В разрабатываемом приложении реализована возможность получения прогноза объема реализаций товаров из учетной системы. Далее представлены полученные значения реализации группы товаров помесячно за 3 года [рис 3]

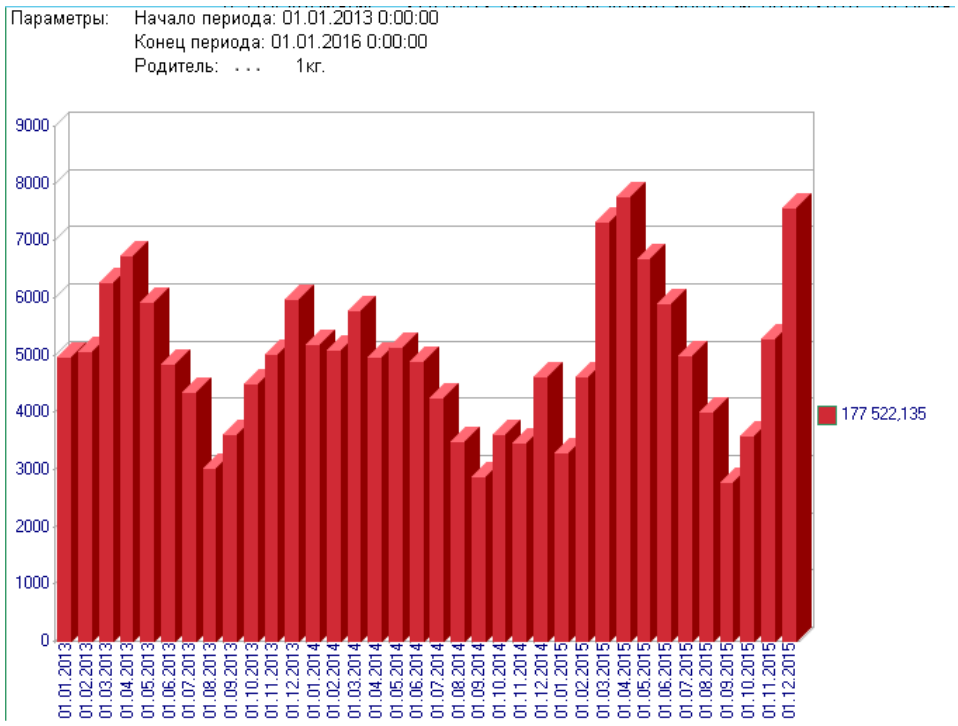

Рис.3 Данные о продажах

Взглянув на данную гистограмму можно сделать вывод о том, что пик продаж приходится на апрель и декабрь, а спад продаж наблюдается в августе-сентябре. Стоит заметить, что в 2014 в исследуемой группе товаров, в апреле, на который приходится наибольшее количество реализаций, наблюдается спад продаж по сравнению с предыдущим месяцем.

2 2.3.3 Обоснование выбора метода прогнозирования

Исходя из представленной гистограммы [рис.4] и замечаний, сделанных в предыдущем пункте следует вывод, что для исследуемой группы товаров невозможно построить точную математическую модель исследуемого объекта. Таким образом, выбор метода прогнозирования был сделан в пользу аппарата нейронных сетей, исходя из их определения, т.е. способность нейронной сети к прогнозированию напрямую следуют из ее способности к обобщению и выделению скрытых зависимостей между входными и выходными данными. После обучения сеть способна предсказать будущее значение некой последовательности на основе нескольких предыдущих значений и/или каких-то существующих в настоящий момент факторов.

3 2.3.4 Работа с приложением

В общем виде приложение работает следующим образом-мы получаем исходные данные(параметры) из учетной системы. Все переданные в качестве аргументов и результатов данные рандомно делятся перед началом обучения на 3 части (Тренировочная, Тестовая и Верификационная) в соотношении 7:2:1. Этот разбиение каждый раз разное и данные делятся перед каждым циклом обучения. Перед началом обучения Входы взвешиваются и от средневзвешенной входов получается значения выходов (всех) по функции активации. Нейрон полностью описывается своими весами и функцией активации F. Получив набор чисел (вектор) в качестве входов, нейрон выдает некоторое число на выходе. Функция активации в нашем случае логистический сигмоид с постоянным коэффициентом кривизны 2.

$$\psi = F(\psi) = \frac{1}{1+e^{-\psi}}$$

То есть первый слой нейронов - это исходные данные. Каждый нейрон первого слоя выдает свое значение без преобразований на все нейроны второго слоя. Второй слой вычисляет среднюю от пришедших в него сигналов, вычисляет от нее величину сигмоиды и раздает полученное значение третьему слою и т.д. пока в последнем слое не соберется на выходе набор результатов. То есть нейроны первого слоя - это факторы, нейроны последнего - результаты. Все внутренние слои - предмет Вашей фантазии. Суть процесса обучения сети заключается в том, что мы подаем N-ное количество известных наборов факторов на вход сети и смотрим, что получается на выходе. В зависимости от того насколько расчетное значение результата (последнего слоя) отклонилось от известного для обучения проводим коррекцию весов сигналов у каждого нейрона. Метод коррекции - обратное распространение ошибки (отклонения между фактическим и расчетным результатом). И так от набора данных к набору, от этапа к этапу сеть настраивает веса таким образом, чтобы эта ошибка была все меньше и меньше. В итоге по завершению обучения мы подаем набор переменных на вход для которого пока еще не известен результат и вычисляем его через настроенную сеть.

На вход подаем набор параметров (П) и известный результат (Р) Скажем 100 строк:

1. П1 П2 ....Пn = Р1

.....

100. П1 П2 ....Пn = Р100

на этом массиве сеть учится правильно получать результаты на основе параметров.Далее еще 100 строк:

101. П1 П2 ....Пn = Р101 результат сети С101 отклонение С101-Р101

.....

200. П1 П2 ....Пn = Р200

сеть свои результаты сравнивает с реальными, отсюда получаем значение погрешности обучения сети

на следующем массиве (10 строк)

201. П1 П2 ....Пn = ? результат сети С201

.....

210. П1 П2 ....Пn = ?

Мы имеем только массив параметров, но результатов не знаем а сеть с выявленной погрешностью предоставляет нам свои результаты расчетов для каждого набора (строки) параметров.

После запуска приложения первым делом необходимо построить модель нейронной сети. Предусмотрен визуальный интерфейс для облегчения работы пользователя и наглядности [рис.4]

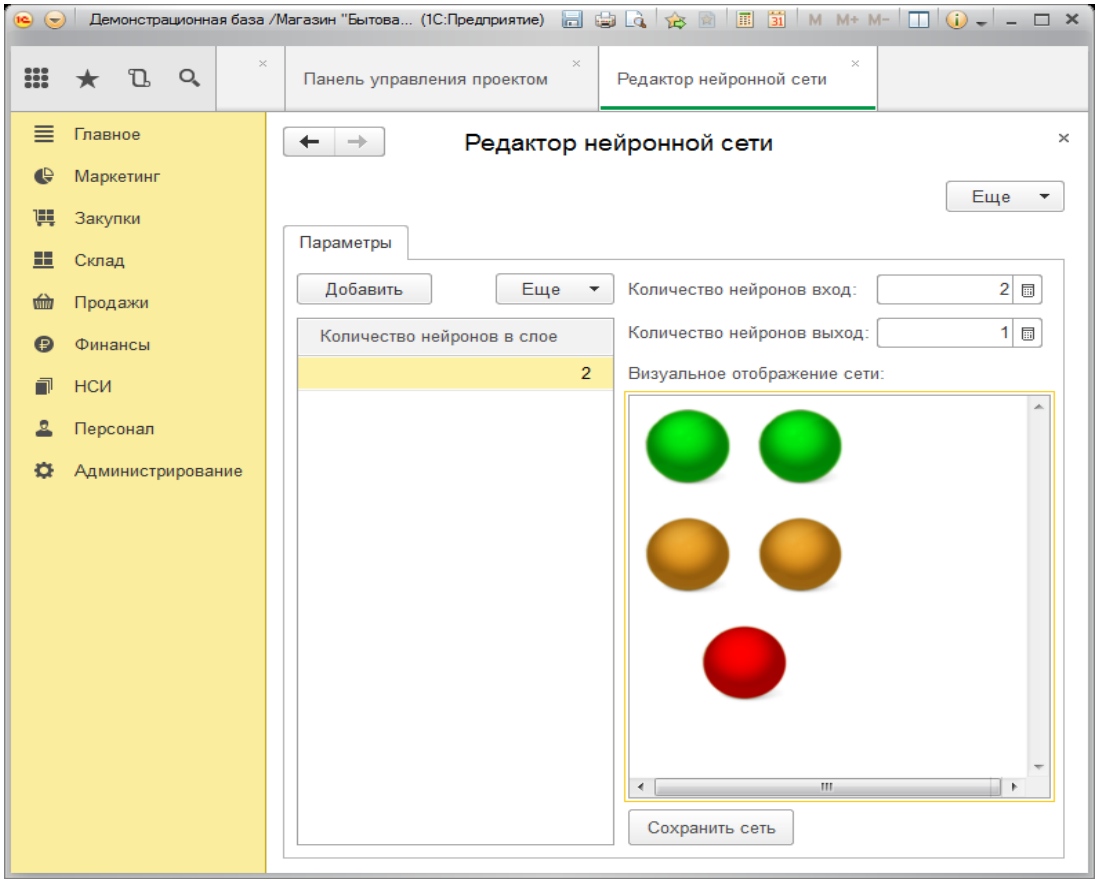

Рис.4 Визуальный интерфейс формы работы с нейросетью

Количество входных и выходных нейронов должно соответствовать количеству типов входных данных и результатов соответственно.

Внутренние слои - это синапсы (связи) для каждого из которых сеть подбирает и запоминает вес. Чем больше слоев и нейронов тем больше связей и весов и тем больше область поиска решения. Для простых задач достаточно сетей с одним скрытым слоем, а если переменных много и связи между ними далеки от линейных то гибкость и адаптивность решения повышается за счет увеличения количества внутренних нейронов. Но при этом увеличивается и время обучения.

После сохранения модели сети необходимо получить данные о продажах и остатках. В оптовых и розничных сетях нередко используют схему резервирования, поэтому количество товара на складе может быть меньше нуля. Для получения этих данных так же предусмотрена специальная форма в интерфейсе обработки [рис.5]

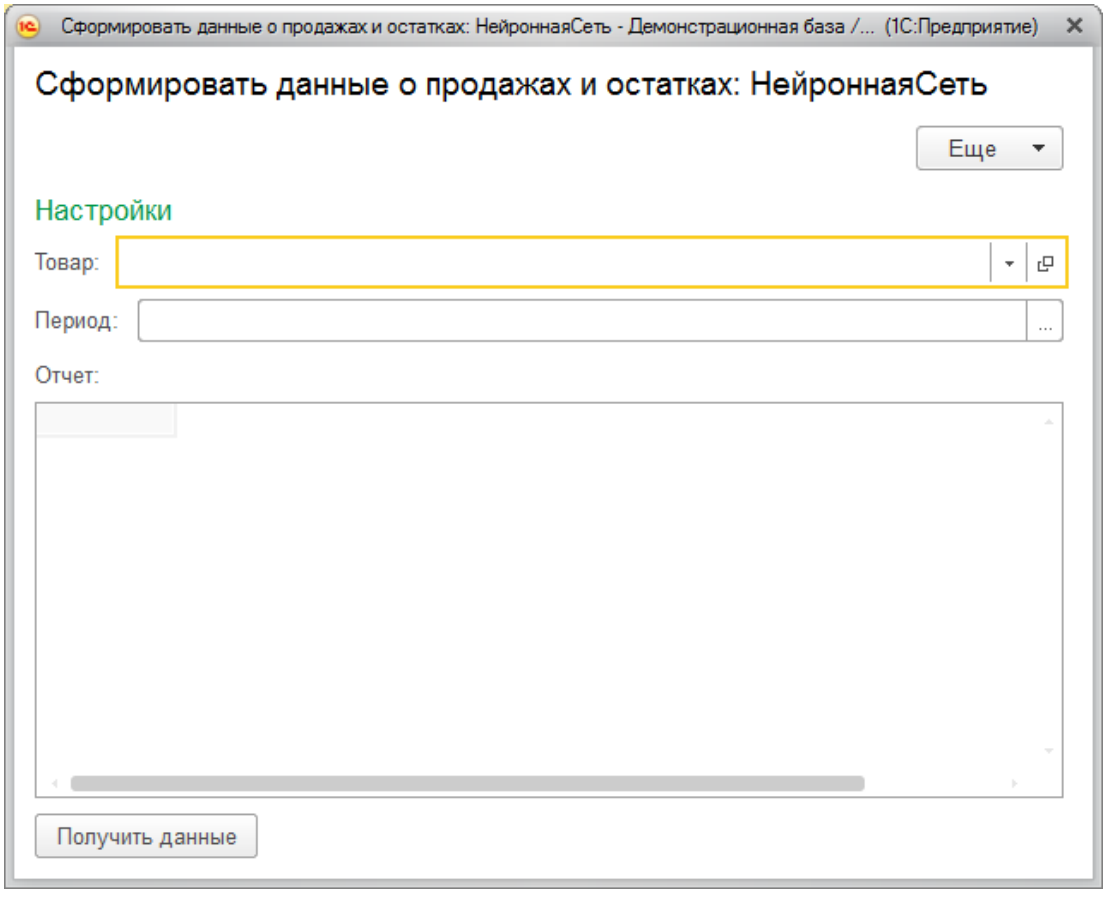

Рис.5 Форма получения исходных данных

В этой форме выбирается группа товаров, либо единичный товар. При выборе периода в запросе 1с отчет формируется помесячно, период может быть произвольным, либо заранее определенным [рис.6]

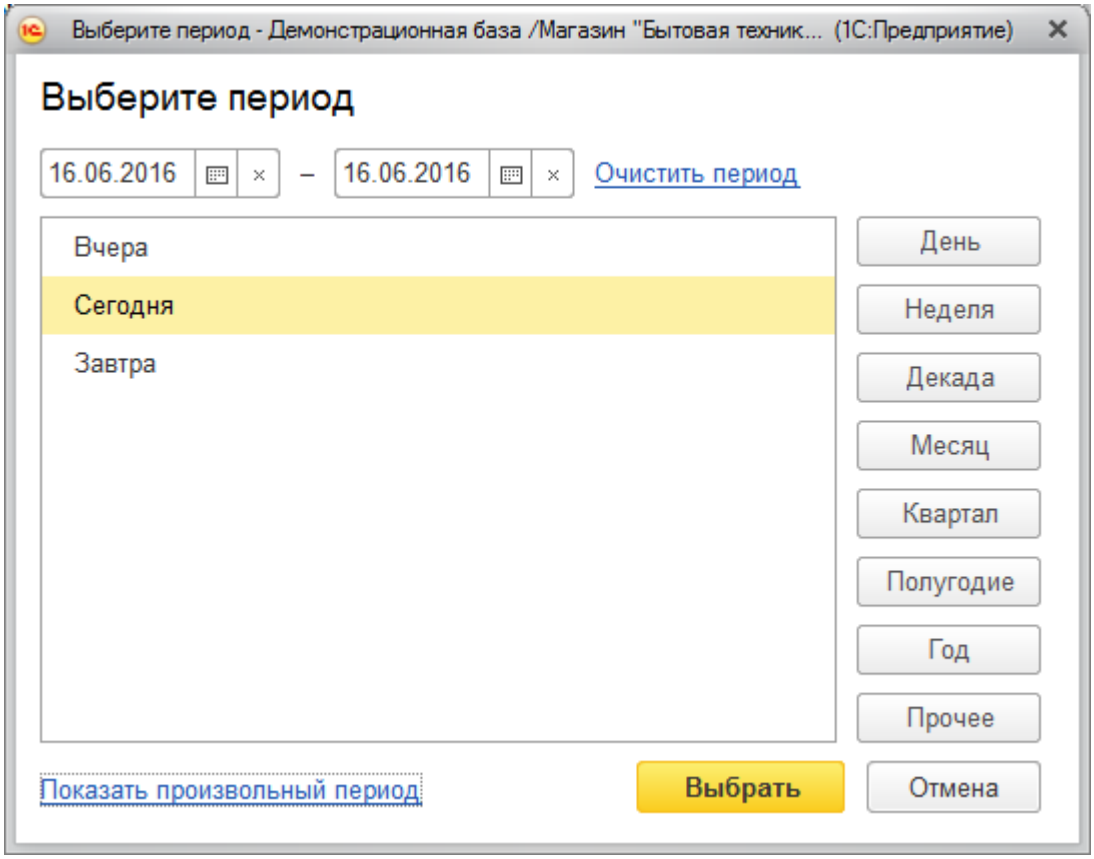

Рис.6 Выбор периода

После нажатия на кнопку «получить данные» исходные значения и результаты програмно помещаются в таблицу значений для дальнейшей работы с ними.

Обучение неронной сети происходит по алгоритму обратного распространения ошибки

$$w_{p-q}(i+1) = w_{p-q}(i) + \eta \delta_q OUT_p$$

где:

i – номер текущей итерации обучения;

w_{p-q} – величина синаптического веса, соединяющего нейрон p с нейроном q;

η – коэффициент «скорости обучения», управляет величиной изменения весов;

δ_q – ошибка q-нейрона;

OUT_p – выход нейрона p

На главной форме указываем η-«норму скорости обучения»,ошибки аппроксимации для обучения и предельное число итераций. При очень маленьких значениях обучение нейронной сети будет проходить медленно. При очень больших значениях η возникает вероятность того, что в момент достижения минимума функции ошибки E=f(W) нейронная сеть не сможет попасть в этот минимум и будет бесконечно долго «прыгать» справа и слева от него, производя перерасчеты весовых коэффициентов. На страте принимается значение, указанное в параметрах, но по ходу обучения скорость обучения может меняться. Принцип модификации скорости состоит в том, что в случае когда между шагами обучения направление градиента для ребра сети не меняется скорость обучения умножается на коэффициент К>1, а в случае изменения знака градиента множитель К становится <1. [рис.7]

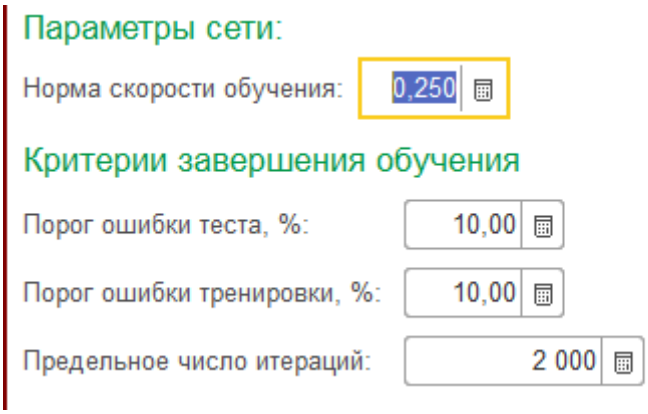

Рис.7 Форма параметров обучения сети

Для обучения нейросети предусмотрено 2 режима

- Тестовая/тренировачная
- Верификационная

Тестовая - не участвует прямо в обучении, но ее ошибка видна в процессе. Судя по значению тренировочной ошибки, а точнее видя ее динамику в сравнении с ошибкой тренировки можно судить о том не переучена ли сеть (ошибки расходятся) и вообще насколько адекватно данным построена модель сети.

Классический пример переучившейся сети это когда ошибки тренировки и теста сначала снижаются, а потом расходятся и ошибка теста бесконечно возрастает. Как правило это случается когда исходная выборка слишком мала или слишком однородна и / или количество внутренних слоев и нейронов при этом слишком велико. Для просмотра статистики обучения необходимо нажать на кнопку «Просмотр статистики», где будет представлен график зависимости ошибок от количества итераций (шагов) [рис.8]

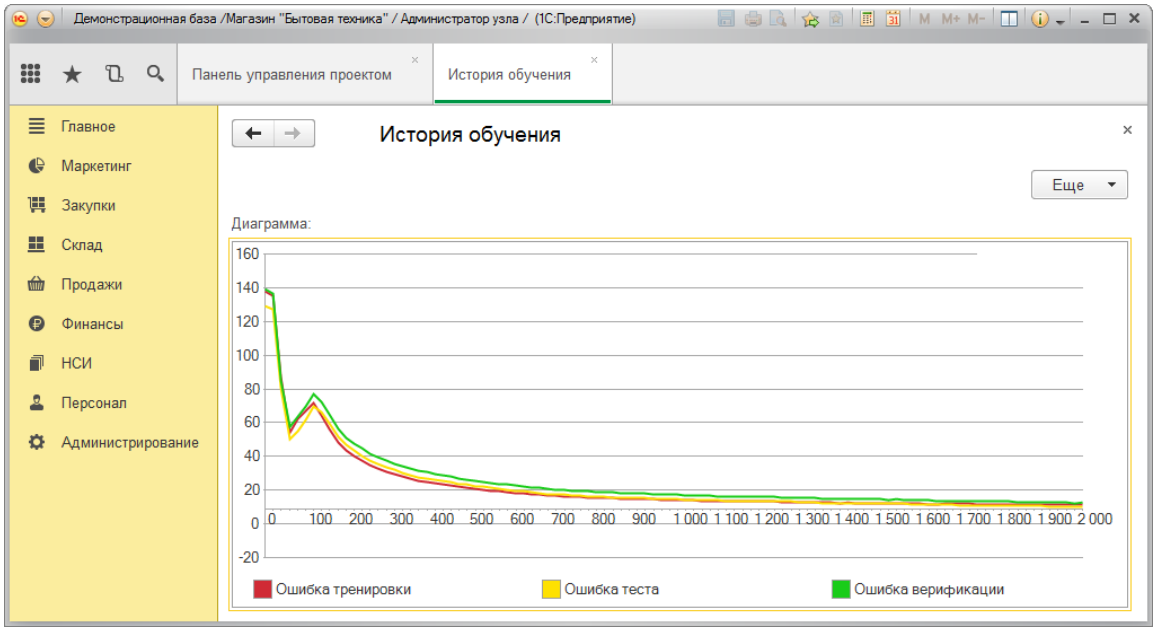

Рис. 8 Статистика обучения сети

Для получения результатов прогноза с учетом указанной ошибки необходимо нажать кнопку «расчет по данным сети» [рис. 9].

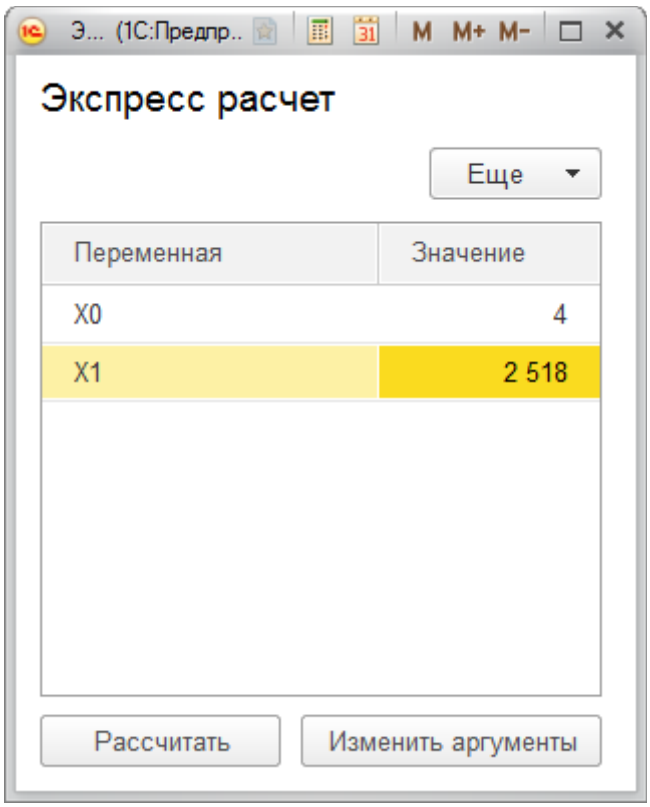

Рис.9 Форма прогнозирования значений

Для сохранения обученной сети предназначена кнопка «сохранить сеть». Выгруженные в массив внутренние и внешние слои нейронов, веса для них и активационная функция сохраняются либо в текстовый файл, либо в справочник «Хранилище значений » типовых конфигураций 1С

4 2.3.5 Анализ показателей прогноза

После обучения нейронной сети полученное значение прогнозируемой величины на прошлый месяц, т.е. апрель 2016 выдается равному 9571. [рис.10]

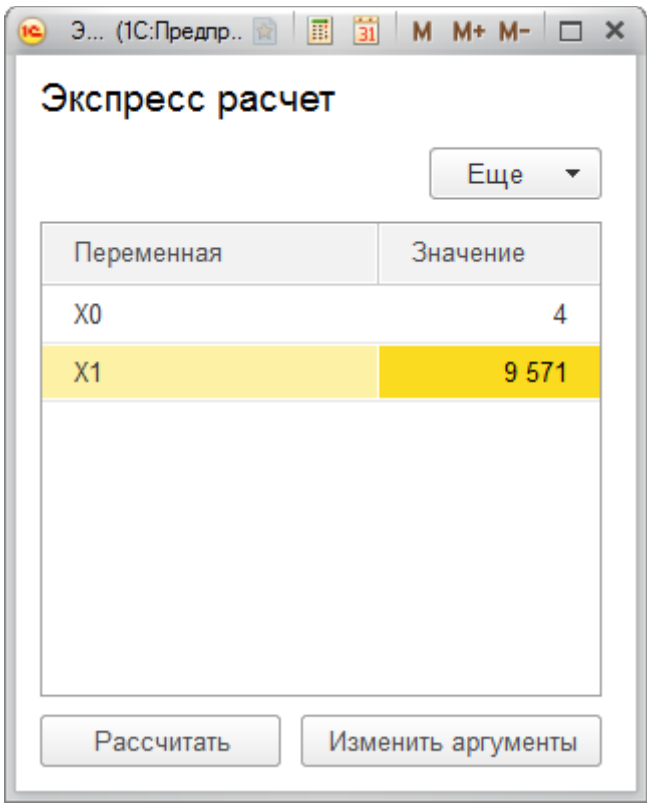

Рис.10 Прогнозируемое значение

Исходя из этих данных, следует вывод, что количество реализуемой продукции в килограммах, по сравнению со всеми прошлыми периодами [рис.3] является рекордным, и руководителю при планировании закупок данной группы товаров имеет смысл задуматься над увеличением количества поставки исследуемой продукции.

Однако, сформировав отчет в учетной системе, и сравнив данное значение с фактическим количеством видно, что [рис.11] фактический объем реализаций за заданный период равен 7057,256, что значительно отличается от прогнозируемого значения.

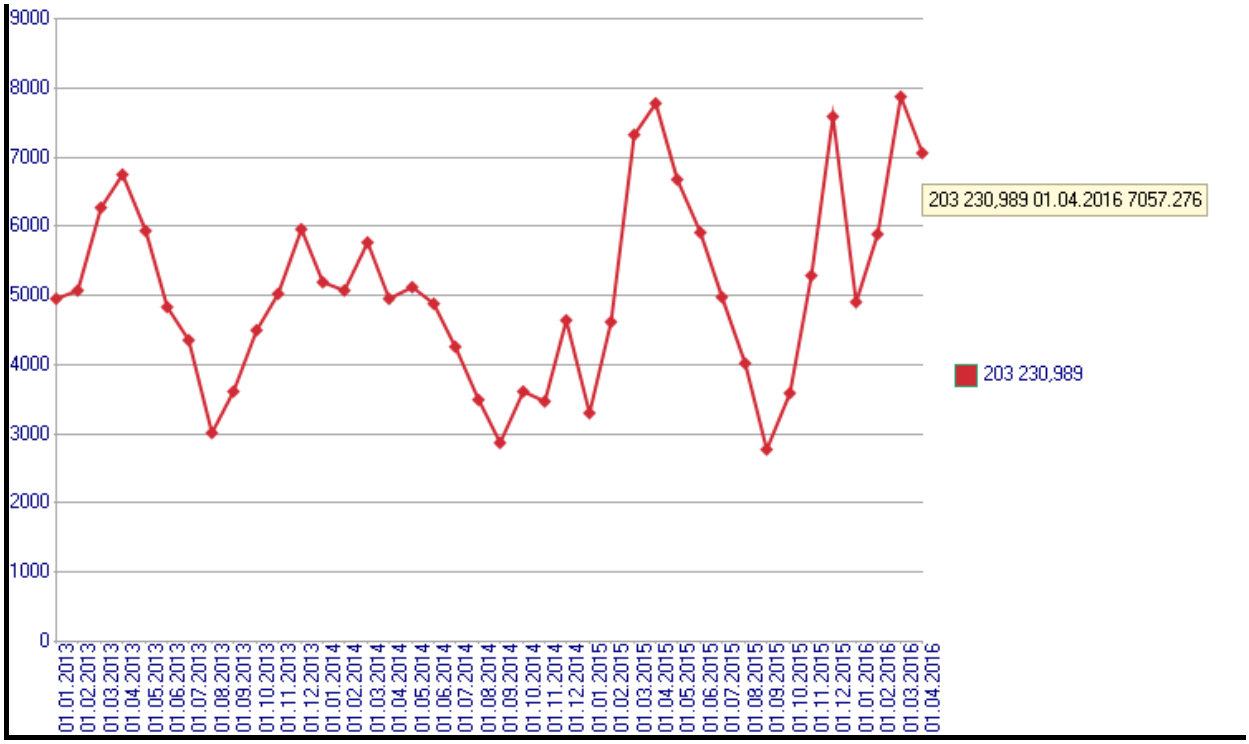

Рис.11 Фактическое количество отгруженного товара

Из этого следует вывод, что спроектированная нейронная сеть выдает результат, превышающий заданное значение ошибки 10%. Это может быть связано с тем, что не совсем корректно была спроектирована нейронная сеть (состав входных нейронов и внутренние слои) или в исходных данных оказалось недостаточно для создания обучающий выборки.

3 Глава. Техническая реализация

Функционально симулятор разбит на несколько модулей

- Редактор нейронной сети (многослойного персептона). Дает возможность создавать и редактировать нейронную сеть. Визуально сеть отображается в виде структуры из разноцветных шаров. Зеленые шары - входные ядра, красные - выходные, а желтые - ядра внутренних слоев. Имеется возможность устанавливать количество слоев и ядер в них. Для загруженных заданий и сетей, а также в случае, когда для обучения проекта уже были загружены данные, количество входных и выходных ядер принимается равным количеству существующих аргументов и результатов. При нажатии кнопки "Сохранить сеть" происходит сохранение сети в параметрах обработки и инициализация начальных весов ребер случайным образом
- Панель управления проектом позволяет редактировать компоненты обучения сети, управлять процессом обучения и контролировать результаты обучения сети. Также присутствуют возможности сохранения и загрузки заданий и готовых сетей
- Подсистема анализа хода обучения. Подсистема представляет собой график, демонстрирующий ход обучения сети в данном проекте. Шаг сбора статистики и общее число шагов обучения настраиваются в "Панели управления проектом". Статистику нельзя видеть в real time по ходу обучения. Она является такой же составной частью проекта, как сеть и массивы данных для обучения и будет обновлена и записана в проект по завершению обучения вместе с новыми параметрами сети. Соответственно если в загруженном проекте при предыдущем обучении была включена опция ведения статистики то просмотр графиков ошибок аппроксимации доступен сразу после загрузки задания (проекта).

- Подсистема вывода результатов прогноза. Задаются исходные данные и результат выводится в специальной форме при заданной пользователем ошибки

5 4 Глава Эргономика

По данным статистики за последние годы для автоматизации управления предприятием наиболее популярной учетной системой в России считается 1С:Предприятие [рис.12]

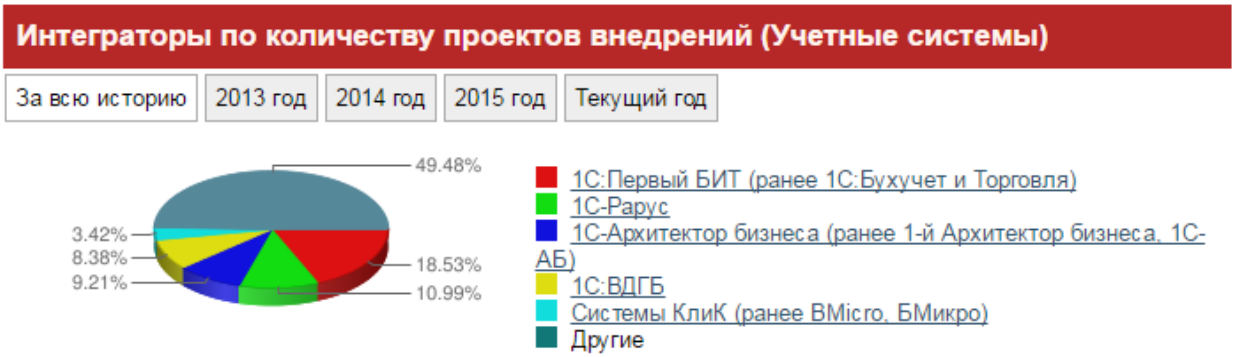

Рис.12 Статистика внедрения учетных систем

Таким образом выбор платформы я сделал в пользу 1С:Предприятие с целью упростить работу управляющему персоналу с приложением, способным работать в текущем запущенном сеансе пользователя. Данный выбор обусловлен так же потенциальной возможностью большего числа внедрений в виду популярности данной учетной системы

Разработанное приложение является встраиваемой внешней обработкой 1С:Предприятия версии 8.3, взаимодействующим с пользователем с помощью фирменного интерфейса системы

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была рассмотрена теория создания искусственных нейронных сетей, подробно исследованы алгоритмы обучения нейронной сети, ее строение. Исходя из полученных данных и поставленных задач, было реализовано приложение прогнозирования объема продаж на предприятии, основанной на теории нейронных сетей. Теоритически, приложение универсально и может быть использовано для решения разнообразных экономических прогнозов, например себестоимости товаров, прогнозирования котировки валют, вероятность возвращения долга в банковских структурах. При этом важно подать для обучения адекватную обучающую выборку, структуру которой необходимо тщательно выбрать. Результат прогноза в данной дипломной работе оказался значительно больше фактического значения, что говорит о необходимости более детального изучения аппарата нейронных сетей для решения задач прогнозирования и дальнейшей доработки приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Банковское дело: современная система кредитования: учебное пособие / О.И. Лаврушин, О.Н. Афанасьева, С.Л. Корниленко. – М.: КноРус, 2007. – 264с.

2. Вертакова, Ю.В., Кузьбожев Э.Н. Упреждающее управление на основе новых информационных технологий: Учеб. пособие/ Под ред. д-ра экон. наук Э.Н. Кузьбожева; Курск. гос. техн. ун-т. Курск, 2001. – 152 с.

3. Вертакова, Ю.В. Экономика отраслевого комплекса (прогнозирование будущего и регулирование настоящего). Монография/ Ю.В. Вертакова, Э.Н. Кузьбожев; Курск. гос. техн. ун-т. – Курск: КГТУ, 2001. – 210 с.

4 .Галушкин, А.И. Теория нейронных сетей: Учебное пособие для вузов/ А.И. Галушкин. – М.: ИПРЖР, 2000. – 416 с.

5 .Информационные технологии в бизнесе/ Под ред. М. Желена. – СПб: Питер, 2002. – 1120 с.

6 .Конюховский, П.В., Колесов Д.Н. Экономическая информатика/ Под ред. П.В. Конюховского, Д.Н. Колесова. – СПб: Питер, 2001. – 560 с.

7. Романов, В.П. Интеллектуальные информационные системы в экономике: Учебное пособие/ Под ред. д.э.н., проф. Н.П. Тихомирова. – М.: Изд-во «Экзамен», 2003. – 496 с.

8. Стратегическое моделирование и прогнозирование: Учеб. пособие/Г.М. Гамбаров, Н.М. Журавель, Ю.Г. Королев и др.; Под ред. А.Г. Гранберга. – М.: Финансы и статистика, 1990. – 383 с.

9. Уразбахтин, И.Г. Алгоритм предобработки данных для целей прогнозирования на базе технологии нейронных сетей/ И.Г. Уразбахтин, Н.И. Рыков// Современные инструментальные системы, информационные технологии и инновации. – Курск: КурскГТУ, 2003. – С. 64-67.

10. Перцептрон [Электронный ресурс] //. Перцептрон – Режим доступа: https://ru.wikipedia.org/Wiki/Перцептрон (Дата обращения 02.04.2016)

11. Нейронные сети [Электронный ресурс] //. Нейронные сети- Режим доступа: http://www.basegroup.ru (Дата обращения 26.04.2016).
12. Прогнозирование с использованием нейронных сетей. [Электронный ресурс] //. Прогнозирование с использованием нейронных сетей. –Режим доступа: http://treide.ru. (Дата обращения 13.05.2016).
13. Аналитические технологии для прогнозирования и анализа данных. [Электронный ресурс] //.Аналитические технологии для прогнозирования и анализа данных. –Режим доступа: http://www.neuroproject.ru (Дата обращения 26.04.2016).

6 Приложение 1

Листинг основных модулей

```
&НаСервере
Процедура ПодготовитьДанныеДляОбученияПример() Экспорт
	//Моделируем функции (X1^2 - X2^3 + X3^4 - X4^5 + X5) и  Sqrt(X1*X2*X3*X4) - (X1*X2*X3*X4)/X5^5
	Сл = Новый ГенераторСлучайныхЧисел();
	ВерхГ = 16;
	НижнГ = 0;
	КвоСтрок = 100;

	Т = Новый ТаблицаЗначений;
	Т.Колонки.Добавить("X1",Новый ОписаниеТипов("Число"));
	Т.Колонки.Добавить("X2",Новый ОписаниеТипов("Число"));
	Т.Колонки.Добавить("X3",Новый ОписаниеТипов("Число"));
	//Т.Колонки.Добавить("X4",Новый ОписаниеТипов("Число"));
	//Т.Колонки.Добавить("X5",Новый ОписаниеТипов("Число"));


	Р = Новый ТаблицаЗначений;
	Р.Колонки.Добавить("Y1",Новый ОписаниеТипов("Число"));
	//Р.Колонки.Добавить("Y2",Новый ОписаниеТипов("Число"));

	Для сч = 1 По КвоСтрок Цикл
		НС = Т.Добавить();
		НС.X1 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
		НС.X2 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
		НС.X3 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
		//НС.X4 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
		//НС.X5 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
		НР = Р.Добавить();
		//НР.Y1 = Pow(НС.X1,2) - Pow(НС.X2,2) + Pow(НС.X3,3) - Pow(НС.X4,3) + НС.X5;
		//НР.Y2 = Sqrt(НС.X1*НС.X2*НС.X3*НС.X4) - (НС.X1*НС.X2*НС.X3*НС.X4) + Pow(НС.X5,5);
		НР.Y1 = 2*НС.X1+4*НС.X2-8*НС.X3;

	КонецЦикла;
	ДобавитьТаблицуНаФорму("ИсходныеДанные","ТаблицаИсходныхДанных", Т,"грДанныеДляОбучения");
	ДобавитьТаблицуНаФорму("Результаты","ТаблицаРезультатов", Р,"грДанныеДляОбучения");
	ТекОбъект = РеквизитФормыВЗначение("Объект");
	ТекОбъект.ДанныеДляОбучения.Вставить(0,ЭтаФорма.ТаблицаИсходныхДанных);
	ТекОбъект.ДанныеДляОбучения.Вставить(1,ЭтаФорма.ТаблицаРезультатов);
	ТекОбъект.ПараметрыСети.Вставить(0,т.Колонки.Количество());
	ТекОбъект.ПараметрыСети.Вставить(1,р.Колонки.Количество());
	ТекОбъект.ПараметрыСети.Вставить(2, Новый СписокЗначений);
	ТекОбъект.ПараметрыСети.Вставить(3, "Линейная");
	ЗначениеВРеквизитФормы(ТекОбъект,"Объект");
```

```
	ТекОбъект = Неопределено;
	спОписание = ПолучитьДанныеДляОбученияСетиИзДанныхОбъекта();
КонецПроцедуры


&НаСервере
Процедура ПровестиКоррекциюВесовДляРебер(ФактическийВыход, тОписаниеРебер, тСостоянияЯдер, СкоростьОбучения)
	Перем Конст, КоэффициентИнерции;
	КоэффициентИнерции = 0.2;
	АддитивныйБонус = 0.05;
	МультипликативныйШтраф = (1-АддитивныйБонус);

	Конст = 2;
	ФВ = Новый Массив;
	Для каждого Элем Из ФактическийВыход Цикл
		ФВ.Добавить(Элем);
	КонецЦикла;
	//Вычисляем сигма смещения для ядер
	сч = тСостоянияЯдер.Количество() - 1;
	Пока сч >=0 Цикл
		СтрокаЯдра = тСостоянияЯдер[сч];
		Если СтрокаЯдра.ВыходнойСлой Тогда
			ФактЯдра = ФВ[ФВ.Количество() - (тСостоянияЯдер.Количество() - Сч)];
			СтрокаЯдра.СигмаЯдра = Конст * СтрокаЯдра.ЗначениеВыходаЯдра * (1 - СтрокаЯдра.ЗначениеВыходаЯдра) * (ФактЯдра - СтрокаЯдра.ЗначениеВыходаЯдра);
		Иначе
			стрВыходы = Новый Структура("НомерНейронаВход", СтрокаЯдра.НомерЯдра);
			СтрокиВыходов = тОписаниеРебер.НайтиСтроки(стрВыходы);
			СуммаВзвешенныхСигмаВерхнегоУровня = 0;
			Для Каждого Строка Из СтрокиВыходов Цикл
				СтрокаВыходногоНейрона = тСостоянияЯдер.Найти(Строка.НомерНейронаВыход, "НомерЯдра");
				СуммаВзвешенныхСигмаВерхнегоУровня = СуммаВзвешенныхСигмаВерхнегоУровня +
				СтрокаВыходногоНейрона.СигмаЯдра * Строка.ВесДляРебра;
			КонецЦикла;
			СтрокаЯдра.СигмаЯдра = Конст * СтрокаЯдра.ЗначениеВыходаЯдра * (1 - СтрокаЯдра.ЗначениеВыходаЯдра) * СуммаВзвешенныхСигмаВерхнегоУровня;
		КонецЕсли;
		сч = сч - 1;
	КонецЦикла;
	//Корректируем веса ребер
	Для Каждого строка Из тОписаниеРебер Цикл
		НачальноеЯдроРебра = тСостоянияЯдер.Найти(Строка.НомерНейронаВход, "НомерЯдра");
		КонечноеЯдроРебра = тСостоянияЯдер.Найти(Строка.НомерНейронаВыход, "НомерЯдра");
```

```
            ДельтаВеса = СкоростьОбучения * КонечноеЯдроРебра.СигмаЯдра *
НачальноеЯдроРебра.ЗначениеВыходаЯдра;

            МодификаторСкоростиОбучения = ?( ДельтаВеса * Строка.ДельтаВеса > 0,
Мин(Строка.МодификаторСКоростиОбучения + АддитивныйБонус,100),
Макс(Строка.МодификаторСКоростиОбучения * МультипликативныйШтраф,0.01));
            ДельтаВеса = ДельтаВеса * МодификаторСкоростиОбучения;
            Строка.ДельтаВеса = ДельтаВеса;
            Строка.МодификаторСКоростиОбучения = МодификаторСкоростиОбучения;

            Строка.ВесДляРебра = Строка.ВесДляРебра + ДельтаВеса;

        КонецЦикла;

КонецПроцедуры

Процедура ОптимизироватьНачальноеРешение(ЗаданиеНаОбучение,
ПараметрыПоискаРешения = Неопределено) Экспорт

        Если ПараметрыПоискаРешения = Неопределено Тогда
            ПараметрыПоискаРешения = Новый Структура;
            ПараметрыПоискаРешения.Вставить("РазмерПопуляции", 30);
            ПараметрыПоискаРешения.Вставить("ТочностьВычислений", 1);
            ПараметрыПоискаРешения.Вставить("КоличествоПоколений", 50);
            ПараметрыПоискаРешения.Вставить("СходимостьПопуляции", 0.8);
            ПараметрыПоискаРешения.Вставить("ИнтенсивностьМутаций", 0.3);
            ПараметрыПоискаРешения.Вставить("ОДЗ", Новый Структура("Верх, Низ",
3, -3));
        КонецЕсли;

        ОР =
ПолучитьОпорноеРешениеГенетическимАлгоритмом(ПараметрыПоискаРешения,
ЗаданиеНаОбучение);
        тРебер = ЗаданиеНаОбучение[3].Значение[0].Значение;
        тРебер.ЗагрузитьКолонку(ОР, "ВесДляРебра");
        ЗаданиеНаОбучение[3].Значение.Вставить(0, тРебер, "ОписаниеСтруктурыРебер");

КонецПроцедуры

&НаСервере
Процедура ВычислитьСостоянияЯдер(НормированныйВход, Знач тРебер, тЯдер)
        тВременная = тРебер;
        мВходов      = тРебер.ВыгрузитьКолонку("НомерНейронаВход");
        мВыходов     = тРебер.ВыгрузитьКолонку("НомерНейронаВыход");

        Для Каждого Ядро Из тЯдер Цикл
            Если мВыходов.Найти(Ядро.НомерЯдра) = Неопределено Тогда
                //Ядро входа
                Ядро.ЗначениеВыходаЯдра =
ПолучитьЗначениеФункцииАктивации(НормированныйВход[Ядро.НомерЯдра]);
            Иначе
                стрПоиска = Новый Структура("НомерНейронаВыход",
```

```
Ядро.НомерЯдра);
                    ВходящиеРебра = тВременная.НайтиСтроки(стрПоиска);
                    КомбинированныйВход = 0;
                    Для Каждого Ребро Из ВходящиеРебра Цикл
                        ТекЯдро = тЯдер.Найти(Ребро.НомерНейронаВход,
"НомерЯдра");
                        КомбинированныйВход = КомбинированныйВход +
ТекЯдро.ЗначениеВыходаЯдра * Ребро.ВесДляРебра;
                    КонецЦикла;
                    Ядро.КомбинированныйВход = КомбинированныйВход;
                    Ядро.ЗначениеВыходаЯдра =
ПолучитьЗначениеФункцииАктивации(КомбинированныйВход);
                КонецЕсли;
            КонецЦикла;
КонецПроцедуры

&НаСервере
Процедура ПодготовитьДанныеДляОбученияПример() Экспорт
        //Моделируем функции (X1^2 - X2^3 + X3^4 - X4^5 + X5) и  Sqrt(X1*X2*X3*X4) -
(X1*X2*X3*X4)/X5^5
        Сл = Новый ГенераторСлучайныхЧисел();
        ВерхГ = 16;
        НижнГ = 0;
        КвоСтрок = 100;

        Т = Новый ТаблицаЗначений;
        Т.Колонки.Добавить("X1",Новый ОписаниеТипов("Число"));
        Т.Колонки.Добавить("X2",Новый ОписаниеТипов("Число"));
        Т.Колонки.Добавить("X3",Новый ОписаниеТипов("Число"));
        //Т.Колонки.Добавить("X4",Новый ОписаниеТипов("Число"));
        //Т.Колонки.Добавить("X5",Новый ОписаниеТипов("Число"));


        Р = Новый ТаблицаЗначений;
        Р.Колонки.Добавить("Y1",Новый ОписаниеТипов("Число"));
        //Р.Колонки.Добавить("Y2",Новый ОписаниеТипов("Число"));

        Для сч = 1 По КвоСтрок Цикл
            НС = Т.Добавить();
            НС.X1 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
            НС.X2 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
            НС.X3 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
            //НС.X4 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
            //НС.X5 = Сл.СлучайноеЧисло(НижнГ, ВерхГ)- ВерхГ / 2;
            НР = Р.Добавить();
            //НР.Y1 = Pow(НС.X1,2) - Pow(НС.X2,2) + Pow(НС.X3,3) - Pow(НС.X4,3) +
НС.X5;
            //НР.Y2 = Sqrt(НС.X1*НС.X2*НС.X3*НС.X4) -
(НС.X1*НС.X2*НС.X3*НС.X4) + Pow(НС.X5,5);
            НР.Y1 = 2*НС.X1+4*НС.X2-8*НС.X3;

        КонецЦикла;
```

```
        ДобавитьТаблицуНаФорму("ИсходныеДанные","ТаблицаИсходныхДанных",
Т,"грДанныеДляОбучения");
        ДобавитьТаблицуНаФорму("Результаты","ТаблицаРезультатов",
Р,"грДанныеДляОбучения");
        ТекОбъект = РеквизитФормыВЗначение("Объект");
        ТекОбъект.ДанныеДляОбучения.Вставить(0,ЭтаФорма.ТаблицаИсходныхДанных);
        ТекОбъект.ДанныеДляОбучения.Вставить(1,ЭтаФорма.ТаблицаРезультатов);
        ТекОбъект.ПараметрыСети.Вставить(0,т.Колонки.Количество());
        ТекОбъект.ПараметрыСети.Вставить(1,р.Колонки.Количество());
        ТекОбъект.ПараметрыСети.Вставить(2, Новый СписокЗначений);
        ТекОбъект.ПараметрыСети.Вставить(3, "Линейная");
        ЗначениеВРеквизитФормы(ТекОбъект,"Объект");
        ТекОбъект = Неопределено;
        спОписание = ПолучитьДанныеДляОбученияСетиИзДанныхОбъекта();
КонецПроцедуры


&НаСервере
Процедура ПровестиКоррекциюВесовДляРебер(ФактическийВыход, тОписаниеРебер,
тСостоянияЯдер, СкоростьОбучения)
        Перем Конст, КоэффициентИнерции;
        КоэффициентИнерции = 0.2;
        АддитивныйБонус = 0.05;
        МультипликативныйШтраф = (1-АддитивныйБонус);

        Конст = 2;
        ФВ = Новый Массив;
        Для каждого Элем Из ФактическийВыход Цикл
                ФВ.Добавить(Элем);
        КонецЦикла;
        //Вычисляем сигма смещения для ядер
        сч = тСостоянияЯдер.Количество() - 1;
        Пока сч >=0 Цикл
                СтрокаЯдра = тСостоянияЯдер[сч];
                Если СтрокаЯдра.ВыходнойСлой Тогда
                        ФактЯдра = ФВ[ФВ.Количество() - (тСостоянияЯдер.Количество() -
Сч)];
                        СтрокаЯдра.СигмаЯдра = Конст * СтрокаЯдра.ЗначениеВыходаЯдра
* (1 - СтрокаЯдра.ЗначениеВыходаЯдра) * (ФактЯдра -
СтрокаЯдра.ЗначениеВыходаЯдра);
                Иначе
                        стрВыходы = Новый Структура("НомерНейронаВход",
СтрокаЯдра.НомерЯдра);
                        СтрокиВыходов = тОписаниеРебер.НайтиСтроки(стрВыходы);
                        СуммаВзвешенныхСигмаВерхнегоУровня = 0;
                        Для Каждого Строка Из СтрокиВыходов Цикл
                                СтрокаВыходногоНейрона =
тСостоянияЯдер.Найти(Строка.НомерНейронаВыход, "НомерЯдра");
                                СуммаВзвешенныхСигмаВерхнегоУровня =
СуммаВзвешенныхСигмаВерхнегоУровня +
                                СтрокаВыходногоНейрона.СигмаЯдра * Строка.ВесДляРебра;
                        КонецЦикла;
```

```
                    СтрокаЯдра.СигмаЯдра = Конст * СтрокаЯдра.ЗначениеВыходаЯдра
* (1 - СтрокаЯдра.ЗначениеВыходаЯдра) * СуммаВзвешенныхСигмаВерхнегоУровня;
                КонецЕсли;
                сч = сч - 1;
        КонецЦикла;
        //Корректируем веса ребер
        Для Каждого строка Из тОписаниеРебер Цикл
                НачальноеЯдроРебра = тСостоянияЯдер.Найти(Строка.НомерНейронаВход,
"НомерЯдра");
                КонечноеЯдроРебра =
тСостоянияЯдер.Найти(Строка.НомерНейронаВыход, "НомерЯдра");

                ДельтаВеса = СкоростьОбучения * КонечноеЯдроРебра.СигмаЯдра *
НачальноеЯдроРебра.ЗначениеВыходаЯдра;

                МодификаторСкоростиОбучения = ?( ДельтаВеса * Строка.ДельтаВеса > 0,
Мин(Строка.МодификаторСКоростиОбучения + АддитивныйБонус,100),
Макс(Строка.МодификаторСКоростиОбучения * МультипликативныйШтраф,0.01));
                ДельтаВеса = ДельтаВеса * МодификаторСкоростиОбучения;
                Строка.ДельтаВеса = ДельтаВеса;
                Строка.МодификаторСКоростиОбучения = МодификаторСкоростиОбучения;

                Строка.ВесДляРебра = Строка.ВесДляРебра + ДельтаВеса;

        КонецЦикла;

КонецПроцедуры

Процедура ОптимизироватьНачальноеРешение(ЗаданиеНаОбучение,
ПараметрыПоискаРешения = Неопределено) Экспорт

        Если ПараметрыПоискаРешения = Неопределено Тогда
                ПараметрыПоискаРешения = Новый Структура;
                ПараметрыПоискаРешения.Вставить("РазмерПопуляции", 30);
                ПараметрыПоискаРешения.Вставить("ТочностьВычислений", 1);
                ПараметрыПоискаРешения.Вставить("КоличествоПоколений", 50);
                ПараметрыПоискаРешения.Вставить("СходимостьПопуляции", 0.8);
                ПараметрыПоискаРешения.Вставить("ИнтенсивностьМутаций", 0.3);
                ПараметрыПоискаРешения.Вставить("ОДЗ", Новый Структура("Верх, Низ",
3, -3));
        КонецЕсли;

        ОР =
ПолучитьОпорноеРешениеГенетическимАлгоритмом(ПараметрыПоискаРешения,
ЗаданиеНаОбучение);
        тРебер = ЗаданиеНаОбучение[3].Значение[0].Значение;
        тРебер.ЗагрузитьКолонку(ОР, "ВесДляРебра");
        ЗаданиеНаОбучение[3].Значение.Вставить(0, тРебер, "ОписаниеСтруктурыРебер");

КонецПроцедуры

&НаСервере
```

```
Процедура ВычислитьСостоянияЯдер(НормированныйВход, Знач тРебер, тЯдер)
	тВременная = тРебер;
	мВходов	= тРебер.ВыгрузитьКолонку("НомерНейронаВход");
	мВыходов	= тРебер.ВыгрузитьКолонку("НомерНейронаВыход");

	Для Каждого Ядро Из тЯдер Цикл
		Если мВыходов.Найти(Ядро.НомерЯдра) = Неопределено Тогда
			//Ядро входа
			Ядро.ЗначениеВыходаЯдра = ПолучитьЗначениеФункцииАктивации(НормированныйВход[Ядро.НомерЯдра]);
		Иначе
			стрПоиска = Новый Структура("НомерНейронаВыход", Ядро.НомерЯдра);
			ВходящиеРебра = тВременная.НайтиСтроки(стрПоиска);
			КомбинированныйВход = 0;
			Для Каждого Ребро Из ВходящиеРебра Цикл
				ТекЯдро = тЯдер.Найти(Ребро.НомерНейронаВход, "НомерЯдра");
				КомбинированныйВход = КомбинированныйВход + ТекЯдро.ЗначениеВыходаЯдра * Ребро.ВесДляРебра;
			КонецЦикла;
			Ядро.КомбинированныйВход = КомбинированныйВход;
			Ядро.ЗначениеВыходаЯдра = ПолучитьЗначениеФункцииАктивации(КомбинированныйВход);
		КонецЕсли;
	КонецЦикла;
КонецПроцедуры
```