Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра прикладной математики

РАБОТА ПРОВЕРЕНА
Рецензент, доцент кафедры МиФА,
д.ф.-м.н.
_____ (В.Л. Дильман)
«01» 06 2016 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____(Л.А. Прокудина)
«28» мая 2016 г.

Прогнозирование временных рядов с помощью нелинейной авторегрессионной модели нейронной сети с внешними входами

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–010400.62.2016.881.ПЗ ВКР

Руководитель работы, к.ф.-м.н.,
доц. каф.прикладной математики
_____(С.У. Турлакова)
«28» мая 2016 г.
Автор работы
Студент группы ММиКН-472
_____(А.В. Астахова)
«28» мая 2016 г.
Нормоконтролер, к.ф.-м.н., доц.
каф.прикладной математики
_____(С.У. Турлакова)
«28» мая 2016 г.

Челябинск 2016

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра прикладной математики
Направление подготовки Прикладная математика и информатика

УТВЕРЖДАЮ
Заведующий кафедрой
_____ (Л.А. Прокудина)
29 мая 2015 г.

З А Д А Н И Е

на выпускную квалификационную работу студента
Астаховой Анастасии Владимировны
Группа ММиКН-472

1. **Тема работы** Прогнозирование временных рядов с помощью нелинейной авторегрессионной модели нейронной сети с внешними входами
утверждена приказом по университету от 15 апреля 2016 г. № 661.
2. **Срок сдачи студентом законченной работы** 1 июня 2016 г.
3. **Исходные данные к работе**
3.1. Среда программирования – программа Matlab.
4. **Перечень вопросов, подлежащих разработке**
4.1. Обзор исследований, посвященных прогнозированию временных рядов с помощью нейронных сетей.
4.2. Изучение основных понятий нейронных сетей и временных рядов.
4.3. Выбор и подготовка временного ряда к прогнозированию. Изучение возможностей пакетов программных продуктов.
4.4. Выбор нейронной сети и ее характеристик.
4.5. Написание и отладка программы, реализующей прогнозирование временного ряда с помощью нейронной сети.
4.6. Оценка полученного прогноза и его сравнение с результатами аналогичных исследований.
5. **Иллюстративный материал** (плакаты, альбомы, раздаточный материал, макеты, электронные носители и др.)
5.1. Мультимедийная презентация – 13 слайдов.
Общее количество иллюстраций – 13.

6. **Дата выдачи задания** 1 октября 2015 г.

Руководитель _____ /С.У. Турлакова/
(подпись)

Задание принял к исполнению _____ /А.В. Астахова/
(подпись)

7. **Календарный план**

Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Отметка о выполнении руководителя
1. Обзор литературы.	01.10.15 – 01.11.15	
2. Изучение основных понятий.	02.11.15 – 15.11.15	
3. Выбор и подготовка временного ряда к прогнозированию. Изучение возможностей пакетов программных продуктов.	16.11.15 – 06.12.15	
4. Выбор нейронной сети и ее характеристик.	07.12.15 – 31.01.16	
5. Написание и отладка программы, реализующей прогнозирование временного ряда с помощью нейронной сети.	01.02.16 – 01.04.16	
6. Оценка полученного прогноза и его сравнение с результатами аналогичных исследований.	01.04.16 – 01.05.16	
7. Подготовка пояснительной записки дипломной работы.	01.04.16 – 15.05.16	
8. Проверка работы руководителем, исправление замечаний.	16.05.16 – 19.05.16	
9. Нормоконтроль.	20.05.16 – 26.05.16	
10. Подготовка иллюстративного материала и доклада.	23.05.16 – 28.05.16	
11. Рецензирование, представление зав. кафедрой.	30.05.16 – 01.06.16	

Заведующий кафедрой _____ /Л.А. Прокудина/
(подпись)

Руководитель работы _____ /С.У. Турлакова/
(подпись)

Студент _____ /А.В. Астахова/
(подпись)

АННОТАЦИЯ

Астахова А.В. Прогнозирование временных рядов с помощью нелинейной авторегрессионной модели нейронной сети с внешними входами.– Челябинск: ЮУрГУ, ММиКН-472, 48 с., 7 ил., 6 табл., библиогр. список – 15 наим., 2 прил.

В данной работе ставится задача получения прогноза временного ряда, представленного в виде цен на недвижимость, с помощью нелинейной авторегрессионной модели нейронной сети с внешними входами (NARX). Анализ различного числа нейронов в скрытом слое и алгоритмов обучения показал, что наилучший прогноз получается при 18 нейронах в скрытом слое в случае обучения алгоритмом Левенберга-Марквардта, при 15 нейронах – с использованием Байесовской регуляризации и при 18 нейронах при обучении методом масштабируемых сопряженных градиентов. В результате сравнения трех вышеуказанных сетей и их прогнозов, было выявлено, что нейронная сеть, обученная алгоритмом Левенберга-Марквардта, оказывается менее трудоемкой и показывает наименьшую ошибку прогноза. Таким образом, был получен прогноз цен на первичное и вторичное жилье в Челябинской области на 2015-2020 гг. Ошибка прогноза составила 1,4%, что свидетельствует об эффективности применения нейронной сети с внешними входами для прогнозирования временных рядов.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современный человек большое внимание уделяет своему будущему. Иногда нам необходимо знать, как изменится ситуация в будущем, чтобы предотвратить возможные проблемы или, по крайней мере, быть к ним готовыми. Мы узнаем прогноз погоды на несколько дней вперед, нам небезразлично, как изменится стоимость того или иного продукта в ближайшее время, да и вообще, никто не откажется от возможности «заглянуть» в будущее. Но никто не может точно сказать, что будет завтра, через час, минуту или, даже, секунду. Тем не менее, есть возможность «предсказать» будущее путем построения прогноза.

Прогноз сам по себе не является точным описанием будущего. Он показывает, что может случиться, учитывая предыдущие события, поведение, особенности и другие детали того или иного события, т.е. является обоснованным с научной точки зрения предсказанием, основанным на фактах. И, пока не изобрели машину времени, прогнозирование является единственной разумной нитью, связывающей прошлое, настоящее и будущее.

Стоит также отметить природную способность человека к прогнозированию. Мы можем предсказать какие-то события, основываясь на накопленном опыте, выявленных закономерностях, или иногда даже совершенно случайно. Конечно, речь идет не о предсказании курса валютного рынка на следующий день или подобных вещах, а о более простых жизненных ситуациях. Эта способность есть у человека благодаря нервной системе. Но мы не можем предсказать все. Для этого человек изобрел нейронные сети, которые имитируют структуру и свойства организации нервной системы, позволяя использовать эту модель для решения сложнейших задач. Но реализовать такую модель в идеальном виде далеко не тривиальная задача. Тем не менее, ее мощность, обширная область применения и огромный потенциал делает задачу построения нейронной сети очень актуальной в эпоху развития искусственного интеллекта. Именно поэтому в качестве способа прогнозирования в данной работе была выбрана нейронная сеть.

Цель работы: построить прогноз и оценить эффективность прогнозирования временного ряда посредством нелинейной авторегресионной модели нейронной сети с внешними входами.

Существует множество моделей нейронных сетей. Очень часто именно для прогнозирования временных рядов используют нелинейную авторегрессионную модель нейронной сети с внешними входами (NARX). Так как речь идет об оценке эффективности полученного прогноза, необходимо провести ряд сравнений. В данной работе мы будем сравнивать прогнозы, полученные с помощью разных архитектур выбранной модели нейронной сети (а именно – при разном количестве нейронов в скрытом слое) и с помощью разных алгоритмов обучения (Левенберга-Марквардта, Байесовской регуляризации и метода масштабируемых сопряженных градиентов). В результате необходимо будет выбрать наилучшую по оценкам архитектуру нейронной сети и соответствующий ей алгоритм обучения и предоставить полученный прогноз.

В качестве временного ряда выбраны данные о цене на недвижимость в Челябинской области с 2000 по 2015 гг. с сайта Федеральной службы государственной статистики [1]. В нашем распоряжении средняя цена за 1 кв.м первичного и вторичного жилья за четыре квартала каждого года из вышеуказанного диапазона. С помощью нейронной сети необходимо получить предполагаемые значения цены в кварталы 2015 года на первичное и вторичное жилье и сравнить их с исходными данными, чтобы оценить качество прогноза.

Таким образом, для достижения цели были поставлены следующие задачи:

1) освоить методы: создание и обучение нелинейной авторегрессионной модели нейронной сети с внешними входами, прогнозирование временного ряда с помощью нейронной сети;
2) ознакомиться с пакетами программных продуктов в Matlab;
3) собрать данные для прогнозирования;
4) получить прогноз временного ряда, используя нейронные сети с различными архитектурами и алгоритмами обучения;
5) сравнить полученные прогнозы и сделать выводы.

1 ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ РЯДОВ С ПОМОЩЬЮ НЕЙРОННОЙ СЕТИ

1.1 Обзор литературы

Прогнозирование является актуальной задачей в наше время. Это объясняет большое разнообразие исследований на данную тему.

Основные направления в прогнозировании временных рядов, для которых применяют нейронные сети, связаны с предметной областью исследования.

1) Экономические прогнозы. Сюда относятся прогнозы различных фондовых рынков, стоимости товаров и услуг и т.д. Характерной особенностью таких временных рядов является их высокая волатильность. Примером могут послужить многочисленные исследования, посвященные прогнозированию цен на электроэнергию [2]. Актуальность этой темы заключается в том, что цены на электроэнергию являются ключевым показателем состояния рынка электроэнергии. В работе [2] для прогнозирования была выбрана локальная линейная вейвлетная нейронная сеть (LLWNN), которую обучали с помощью алгоритма обратного распространения с градиентным спуском. В результате прогноз нейронных сетей оказался гораздо точнее прогнозов, полученных другими способами (эвристический метод, метод Independent Electricity System Operator (IESO), множественная линейная регрессия (MLR)). В некоторых случаях ошибка прогноза, полученного нейронной сетью, оказывалась меньше ошибок других прогнозов почти в 3 раза. Это объясняется тем, что выбранная модель нейронной сети показывает наилучшие результаты при моделировании нестационарных высокочастотных сигналов, какими являются цены на электроэнергию. В качестве экспериментальных данных были взяты данные за 2004 г. в Онтарио и за 2013-2014 гг. в Индии. Еженедельная средняя абсолютная ошибка полученного прогноза цен на электроэнергию в Индии составила 7,66%, а в Онтарио – 14,11%.

2) Медицинские прогнозы. К этому типу можно отнести источники, в которых рассматривается прогнозирование количества пациентов, предсказание заболеваемости, смертности, рождаемости и прочее. Например, в [3] поднят вопрос о краткосрочном и долгосрочном предсказании уровня заболеваемости посредством нейронной сети. В качестве примера были использованы данные о заболеваемости скарлатиной за 1996-2008 гг. Для прогнозирования был выбран двухслойный перцептрон, который отличается незамысловатой структурой и простотой обучения и изменения, а в основе обучения нейронной сети лежит метод оптимизации Левенберга-Марквардта. Также предлагалось три варианта структуры самого перцептрона, результаты прогнозов каждой из которых были проанализированы. Ошибка прогноза на два года вперед составила 40%. Тем не менее, в результате исследования было показано, что при специально подобранной обучающей выборке можно получить вполне удовлетворительные

результаты, что доказывает эффективность применения нейронных сетей в данной области.

3) Социальные прогнозы. Одним из главных направлений в этой категории является оценка группового поведения. В качестве примера можно привести исследование о прогнозировании посещаемости футбольного матча [4]. В работе была проанализирована мощность прогнозирования различных моделей нейронных сетей (трехслойный перцептрон, трехслойная сеть прямого распространения с задержкой времени, рекуррентная нейронная сеть, двухслойная нейронная сеть с радиально базисной функцией). Средние ошибки полученных прогнозов в зависимости от выбранной нейронной сети и входных данных варьируются от 3,5% и до 12%. Было показано, что обученная нейронная сеть может успешно учитывать закономерности посещаемости и моделировать ее динамику. Еще к этому же разделу можно отнести, скажем, выбор спортсмена, спрогнозировав его возможные результаты.

4) Природные или естественные прогнозы. К данному разделу можно отнести исследования по таким темам, как предсказание прогноза погоды, предсказание землетрясений, наводнений, осадков, оценка качества воды, количества древесины, радиоактивности и т.д. Например, в статье [5] предсказывают метеорологические переменные. В работе были использованы две искусственных нейронных сети для предсказания максимального и минимального атмосферного давления и солнечного излучения, соответственно. Все данные были взяты с турецкой метеорологической станции. В качестве нейронной сети был выбран многослойный перцептрон, который обучался тремя различными способами: алгоритмом обратного распространения ошибки, методами масштабируемых градиентов (SCG) и Левенберга-Марквардта (LM). Также экспериментировали и с количеством нейронов на скрытом уровне: от 7 до 9 нейронов. В итоге, лучшие результаты для первого и второго прогноза показали нейронная сеть с семью нейронами на скрытом уровне, обученная с помощью метода LM, и с восемью нейронами, обученная на основе того же метода. Средние ошибки прогнозов составили 0,18% и 0,19% в первом случае и 5,78% – во втором. Разница между итоговыми полученными и расчетными значениями очень мала, что показывает высокую эффективность применения нейронной сети для решения данной задачи.

5) Оптимизационные прогнозы. Термин «оптимизационные» используется потому, что на основе полученных в прогнозе данных можно подобрать значения параметров, которые могут улучшить результат. Скажем, это прогноз транспортных потоков, прогнозирование времени проезда по шоссе, оценка пропускной способности железных дорог. Например, в [6] находится краткосрочный прогноз транспортного потока с помощью двоичной нейронной сети с помощью метода AURA k-NN, путем объединения данных с нескольких датчиков движения и прогнозирования временных рядов. Основная задача – получить быстрый и точный прогноз транспортного потока. В данной работе было расширено применение метода AURA k-NN, который раньше использовался только для прогнозирования невременных рядов. Помимо этого данный метод

применялся только для поиска по шаблону, поэтому пришлось обходить и эту проблему, путем получения многомерных прогнозов. Таким образом, в результате был получен модифицированный метод AURA k-NN, который оказался самым эффективным способом получения прогноза с точки зрения скорости и точности.

6) Физические или механические прогнозы. Сюда можно отнести предсказание работы газотурбинного двигателя, эффективность использование ветровой фермы, прогнозирование прочности сжатых полимеров из переработанных отходов, оценка скорости сжатия волны, скорости запуска ракеты, влияние различных составляющих на прочность сжатия бетона, предсказание положения энергетического центра изображения лазерного луча, расход топлива транспортного средства. Например, в [7] прогнозируется скорость и разлет осколков камней при взрывных работах, что необходимо для обеспечения безопасности рабочих, т.е. для определения периметра поражения при взрыве. Данные были взяты из мониторинга взрывных работ в медных рудниках и известняковых карьеров в Сербии за 2011-2013 гг. В качестве нейронной сети был выбран двухслойный перцептрон с 14 нейронами в скрытых уровнях. Результаты показали, что созданная и обученная нейронная сеть предсказывает скорость с высокой точностью. (Только 6% из предсказанных значений выходили за допустимые пределы скорости, то есть не попадали в доверительный интервал). Но недостаток заключается в том, что она не является универсальной: ее предсказания верны только для объектов со схожими свойствами, для которых строился прогноз. В противном случае, для получения точного прогноза сеть необходимо дообучать данными с соответствующего места.

Стоит также заметить, что прогнозирование конкретных явлений – это не единственное направление развития теории прогнозирования. Существует множество работ, цель которых направлена на развитие, модификацию и улучшение самих способов прогнозирования. Иными словами, это главным образом изменение уже существующих нейронных сетей, или же создание новых, что реже. Один из примеров модификации существующей сети указан в [6] и уже был описан выше.

1.2 Понятие временного ряда и способы его прогнозирования

Если данные в модели, на которых она построена, характеризуют какой-то объект за ряд последовательных моментов времени, то мы имеем дело с временным рядом.

Временной ряд – это совокупность значений какого-либо показателя за несколько последовательных моментов (периодов) времени. Временной ряд состоит из нескольких уровней, каждый из которых формируется под воздействием большого числа факторов. Эти факторы условно разбиваются на 3 подгруппы:

1) факторы, формирующие тенденцию ряда;

2) факторы, формирующие циклические колебания ряда;

3) случайные факторы.

В случае различных сочетаний вышеуказанных факторов, зависимость уровней ряда от времени может принимать разные формы:

1) Для некоторых данных, определенная тенденция справедлива только тогда, когда их рассматривают в совокупности. А по отдельности они могут оказывать разнонаправленное воздействие на исследуемый показатель.

2) Для изучаемого показателя могут быть характерны циклические колебания. Если говорить об экономических временных рядах, то такие колебания могут носить сезонный характер. Но эти колебания можно распознать, при наличии больших массивов данных за длительные промежутки времени.

3) Временной ряд может не содержать тенденции или циклов, а каждый следующий их уровень образуется как сумма среднего уровня ряда и некоторой случайной компоненты (положительной или отрицательной).

Но это в теории. В реальности данные зачастую не соответствуют полностью ни одной из вышеперечисленных моделей. Скорее, они содержат в себе сразу все три компоненты. Потому модель, временной ряд которой представлен в виде суммы перечисленных выше компонент, называется аддитивной, а если в виде произведения, то мультипликативной моделью временного ряда.

Основная задача исследования временного ряда заключается в том, чтобы выявить и придать количественное выражение каждой из указанных выше компонент. Полученная информация будет необходима в первую очередь для прогнозирования будущих значений временного ряда. А так же при других исследованиях временных рядов. [8]

Целью задачи прогнозирования временного ряда является по данным наблюдений предсказать будущее значение измеряемых характеристик изучаемого объекта. В настоящее время известно несколько методов прогнозирования временных рядов. Но все их можно разбить на две большие группы: локальные и глобальные. Такое деление обуславливается областью определения параметров аппроксимирующей функции, которая рекуррентно устанавливает следующее значение временного ряда, основываясь на предыдущих.

Вот некоторые модели для прогнозирования временного ряда [9]:

- регрессионные модели;
- авторегрессионные модели (ARMA, ARIMA, GARCH, ARDLM);
- модели экспоненциального сглаживания (ES, модель Хольта);
- нейросетевые модели;
- модели на основе цепей Маркова;
- модели на базе классификационно-регрессионных деревьев (CART);
- метод опорных векторов (SVM);
- генетический алгоритм (GA);
- дискретное преобразование Фурье.

В рамках данной работы мы будем использовать модели прогнозирования с помощью нейронных сетей.

1.3 Понятие нейронной сети и ее особенности

Интерес к разработке искусственных нейронных сетей появился сразу с момента их появления. Причина заключалась в том, что, как выяснилось, человеческий мозг работает совсем не так, как цифровой компьютер. В голове производятся неимоверно сложные, нелинейные и параллельные вычисления. Для выполнения некоторых из них (например, распознавание образов, восприятие, контроль движения) используются специальные структурные частицы – нейроны, позволяющие производить вычисления во много раз быстрее, чем самый мощный современный компьютер. Как же человеческий мозг делает это? С рождения он выступает как значимая структура и способен формировать свои собственные правила поведения, что обычно мы называем опытом. Действительно, опыт накапливается с течением времени, и хотя человеческий мозг формируется первые два года, развивается он гораздо дольше. Пластичность нервной системы позволяет адаптировать ее к окружающей среде.

Таким образом, нейронная сеть – это машина, которая моделирует способ, с помощью которого мозг решает конкретные задачи; обычно сеть реализуется с использованием электронных компонент или моделируется в компьютерной программе. В нашей работе для создания нейронной сети будет использоваться программа Matlab.

Для достижения высокой производительности, нейронные сети используют массив взаимосвязанных просто вычисляющихся элементов, называющихся нейронами. Таким образом, возможно следующее определение нейронной сети с точки зрения адаптации: нейронная сеть – это массивный параллельный распределительный процессор, состоящий из простых обрабатывающих блоков, имеющих природную способность хранения экспериментальных данных и обрабатывать их для дальнейшего использования.

У нее есть две общие черты с мозгом:

1) нейронная сеть получает знания из окружающей среды в процессе обучения;

2) сильная связь между нейронами, известная как синаптический вес, позволяет хранить полученные знания.

Процедура, реализующая процесс обучения, называется алгоритмом обучения, т.е. функцией, которая изменяет синаптические веса сети так, чтобы получать желаемые результаты .

Опишем некоторые преимущества нейронной сети. Очевидно, что она обладает высокой вычислительной мощностью за счет, во-первых, своей массивной параллельной распределительной структуры, и во-вторых, способности к обучению и, следовательно, к обобщению. Обобщение подразумевает получение разумных выходных данных для входных данных, не участвующих в процессе обучения. Эти две способности обработки информации позволяют нейронным сетям успешно находить приближенное решение трудных проблем.

Нейронные сети обладают следующими полезными свойствами и возможностями.

1) Нелинейность. Искусственный нейрон может быть как линейным, так и нет.

2) Отображение входа в выход. Популярный метод обучения с учителем, или еще называемый контролируемым обучением, заключается в изменении синаптических весов нейронной сети за счёт использования набора обучающих (тестовых) примеров. Каждый из таких примеров состоит из уникальных входных данных и соответствующих им нужных выходных данных. Первоначально нейронная сеть инициализируется случайными синаптическими весами (свободными параметрами), которые потом в процессе обучения изменяются так, чтобы минимизировать разницу между желаемым ответом и фактическим результатом работы сети для конкретного входного сигнала. Обучение сети повторяется много раз для всех тестовых примеров, пока сеть не достигнет стабильного состояния, то есть веса перестанут значительно изменяться. Ранее применяемые обучающие примеры могут повторяться в процессе обучения, но в другом порядке. Таким образом, сеть учится с помощью примеров, позволяющих построить отображение ввод-вывод для поставленной задачи.

3) Адаптивность. Нейронная сеть обладает внутренней способностью приспосабливать свои синаптические веса к изменениям в окружающей среде. В частности, обученная под определенную среду нейронная сеть может легко переобучиться к незначительно измененным условиям окружающей среды. Более того, когда она работает в нестационарной среде (т.е. изменяющейся со временем), то может изменять синаптические веса в режиме реального времени.

4) Не вызывающий сомнений ответ. Выход нейронной сети всегда можно проверить на корректность, подав на вход определенный пример.

5) Контекстная информация. Вся информация заключена в самой структуре сети с помощью ее состояния активации.

6) Устойчивость к ошибкам. Реализованная аппаратно нейронная сеть по своей природе устойчива к ошибкам, иными словами, способна к точным вычислениям, в том смысле, что ее производительность медленно снижается при неблагоприятных условиях.

8) Однотипность анализа и проектирования. На самом деле, нейронные сети используют универсальность как информационные процессоры. В том смысле, что одни и те же обозначения используются во всех областях, которые подразумевают применение нейронной сети.

9) Нейробиологическая аналогия. Нейронная сеть проектируется на основе головного мозга, который является живым доказательством того, что устойчивая к ошибкам параллельная обработка не только возможна, но еще и может быть быстрой и мощной. [10]

Основными элементами нейронной сети являются нейроны. Отдельный нейрон в составе искусственной нейронной сети представляет из себя функциональное преобразование вида:

$$O = F(< X; W >),$$

где $X \in R^N$ – входные сигналы, поступающие на данный нейрон с N пресинаптических нейронов (либо со входов всей сети);
$W \in R^N$ – веса пресинаптических связей;
F – нелинейное функциональное преобразование или активационная функция нейрона;
$O \in R$ – выходной сигнал нейрона, который передается на постсинаптические нейроны (или на выход всей сети).

Выбор активационной функции зависит от специфики задачи, удобства реализации, способа обучения. В данной работе используются линейная и сигмоидальная функции. Опишем их подробнее.

1) Линейная функция представлена на рисунке 1 и определяется формулой (1).

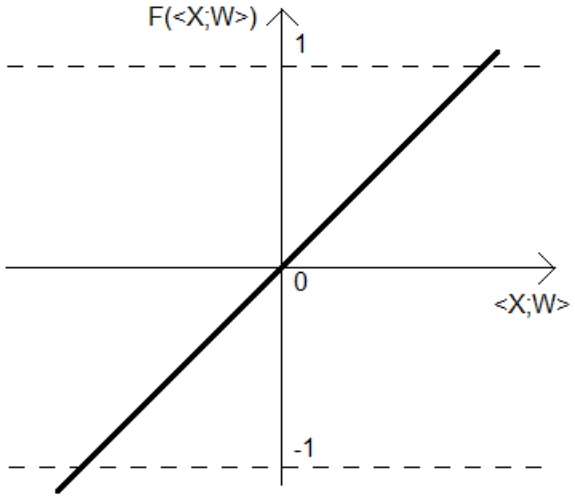

Рисунок 1 – Линейная активационная функция

$$F(< X; W >) =< X; W > \quad (1)$$

2) Сигмоидальная функция представлена на рисунке 2 и определяется формулой (2).

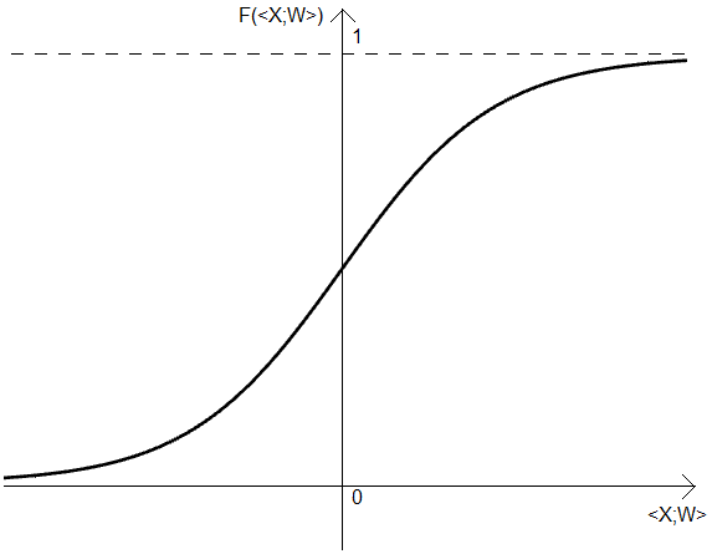

Рисунок 2 – Сигмоидальная активационная функция

$$F(< X; W >) = \frac{1}{1 + e^{-<X;W>}} \quad (2)$$

Все активационные функции должны обладать двумя важными свойствами.

1) Нелинейность. В случае линейности активационной функции последовательное распространение сигнала по сети означает лишь его масштабирование. Помимо этого, линейность не позволяет добиться дополнительных эффектов за счет увеличения количества нейронов в сети (т.е. ее мощности). А благодаря нелинейным активационным функциям можно получать все более сложную функциональную зависимость по мере прохождения сигнала через каждый нейрон, которая и воспроизводится нейронной сетью.

2) Ограниченная область допустимых значений при неограниченной области определения. Величина выходного сигнала всегда остается в заданных пределах, не зависимо от того, какое количество пресинаптических нейронов суммируется по мере прохождения сигнала по сети. В связи с этим нет ограничения на количество связей между нейронами – оно может быть сколь угодно большим, вплоть до полносвязной архитектуры, когда каждый нейрон является пресинаптическим и постсинаптическим для всех остальных нейронов сети. [11]

1.4 Использование нейронной сети для получения прогноза

Пусть на интервале $[0; T_0]$ задана функция $f(x)$, где T_0 – текущее значение времени. Задача состоит в том, чтобы предсказать значение функции при $t > T_0$. Чтобы было возможным применить нейронную сеть, необходимо предварительно обработать входные данные. А именно, необходимо дискретезировать время. Для этого интервал разбивается на *n-1* временных отрезков. Таким образом, считается известным значение функции в конкретный момент времени t_k:

$$x = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

где $x_k = f(t_k),\ k = \overline{0, n}$;
$t_0 = 0$;
$t_n = T_0$;
$t_k = t_{k-1} + 1,\ k = \overline{1, n}$.

Задача состоит в определении значения функции в момент времени, равный $(T_0 + a_0)$, где $a_0 > 0$. Иными словами, a_0 – интервал прогнозирования.

Таким образом, необходимо решить уравнение:

$$f(T_0 + a_0) = y,$$

где y – решение поставленной задачи.

Пусть вектор X – входной вектор, а Y – выходной. Для того чтобы прогнозировать функцию в виде дискретного процесса во времени и предсказывать несколько значений функции в разных точках, необходимо выбрать все интервалы a_i одинаковыми и постоянными (в нашем случае все $a_i = 1$). Далее необходимо построить и обучить сеть. На вход подается вектор X со значениями функций в известных точках и вычисляется выход сети, т.е. $f(T_0 + a_0)$. Затем «сдвигаются» (т.е. изменяются) компоненты входных и выходных векторов по следующим формулам:

$$\begin{cases} x_0 = x_1 \\ x_1 = x_2 \\ \quad \vdots \\ x_{n-1} = x_n \\ x_n = y \end{cases}$$

Теперь выходное значение стало известным, и можно спрогнозировать следующее значение $f(T_0 + 2a_0)$. Повторяя эту операцию, можно прогнозировать функцию в бесконечном количестве точек с дискретным шагом по времени, равным a_0.

Для многомерных функций задача решается аналогично. [12]

Выводы по разделу

Изучение исследований, посвященных прогнозированию временных рядов с помощью нейронных сетей, показало, что нейронные сети очень популярны для решения задач прогнозирования в различных сферах, но пока не удается получить универсальную модель, которая бы точно предсказывала любой временной ряд, независимо от его параметров, как хотелось бы в идеале.

Можно выделить два самых популярных типа нейронных сетей, которые используются для решения задач прогнозирования. Это многослойный перцептрон и NARX сеть.

Одним из преимуществ нейронной сети, несомненно, является то, что для получения прогноза нет необходимости выделять во временном ряду тренд, случайные и циклические составляющие. Предполагается, что нейронная сеть в процессе обучения сама определяет все вышеуказанное и должна учитывать при построении прогноза. Также, мы не будем брать во внимание какие-либо внешние факторы, оказывающие влияние на поведение временного ряда, исходя из тех же соображений. Задача состоит в тома, чтобы получить высококачественный прогноз для нейронной сети, используя минимум входных данных.

Важным выводом является то, что для решения конкретной задачи прогнозирования необходимо выполнить настройку сети для получения наилучших результатов. Под настройкой сети подразумевается определение

главным образом количества нейронов в скрытых слоях. Кроме того, сюда же можно добавить выбор алгоритма обучения нейронной сети.

Таким образом, для исследуемой задачи необходимо разработать и настроить нейронную сеть, решающую задачу прогнозирования временного ряда.

2 МОДЕЛЬ И ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

2.1 Нелинейная авторегрессионная модель нейронной сети с внешними входами

Для прогнозирования временного ряда была выбрана модель нелинейной авторегрессии с внешними входами (NARX). Она представляет собой рекуррентную динамическую сеть с обратной связью, состоящую из нескольких слоев, в ее основе лежит авторегрессионная модель, используемая для описания систем, обладающих инерцией:

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n), x(t-1), \dots, x(t-n))$$

Прогнозируемое значение $y(t)$ зависит от n предыдущих значений выхода и от n предыдущих значений временного ряда. На рисунке 3 представлена схема нейронной сети, использованной в данной работе.

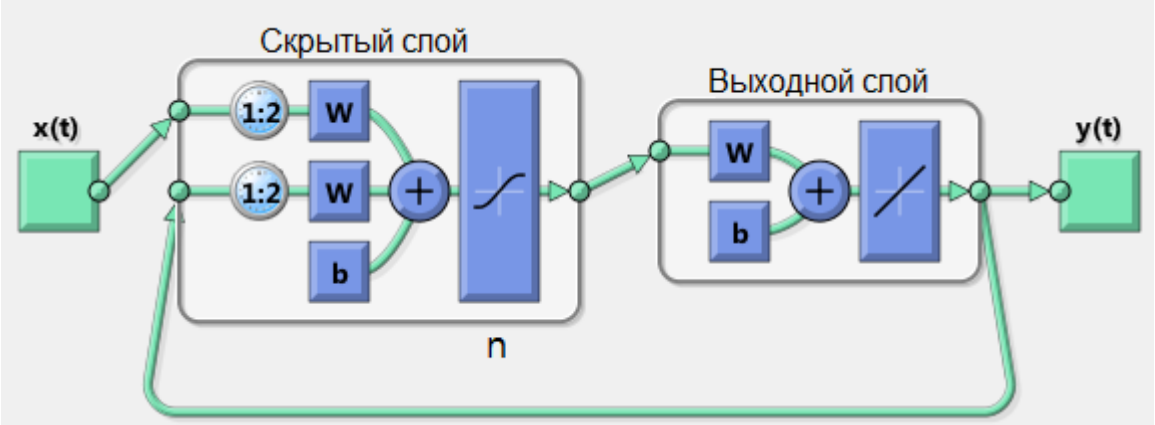

Рисунок 3 – NARX с n нейронами в скрытом слое

Для построения вышеописанной нейронной сети рассмотрим последовательность действий, которую нужно совершить. Сначала нужно определиться с входными и выходными данными. Так как задача состоит в прогнозировании временного ряда, то в качестве входных данных будут использованы номера временных промежутков по порядку, а в качестве выходных – значение рассматриваемого показателя, соответствующее времени, поданному на вход сети.

Для эффективного обучения сети необходимо разделить все данные на три подмножества. Первое понадобится для обучения сети (корректировки весов). Традиционно к этому подмножеству отнесется большая часть тестовой выборки. В данной работе мы использовали 80% данных. Второе подмножество необходимо для контроля обучения сети, которое осуществляется путем вычисления ошибки на данных из этого подмножества. На начальном этапе обучения нейронной сети ошибка должна уменьшаться. Но рано или поздно сеть может переобучиться, о чем будет свидетельствовать увеличение ошибки. Именно для того, чтобы избежать переобучения сети, и необходимо вводить это

второе подмножество данных. Поэтому обученная сеть будет иметь веса, при которых ошибка на проверочном подмножестве была минимальна. И третье подмножество – тестовое. Оно не используется в процессе обучения нейронной сети, но может быть полезным при сравнении различных моделей как полностью независимое тестирование обученной сети. Также тестовое подмножество может свидетельствовать о плохом разделении исходных данных, если минимум ошибки на данном и проверочном множестве достигаются в существенно удаленных друг от друга итерациях. В нашей работе на второе и третье подмножества было выделено по 10% данных.

Стандартная сеть NARX представляет из себя двухслойную сеть обратного распространения. В качестве передаточной функции в скрытом слое выступает сигмоидальная функция (2), а в выходном слое – линейная (1).

Данная модель сети использует линии задержки с отводами для хранения предыдущих значений *x(t)* и *y(t)*. Стоит отметить, что выходное значение *y(t)*подается обратно на вход сети (через задержку), так как *y(t)*является функцией, зависящей в том числе от предыдущих значений: *y(t-1), y(t-2),…, y(t-d)*. Однако, для эффективного обучения это кольцо обратной связи может быть разомкнуто (рисунок 4). Поскольку истинный выход доступен во время обучения нейронной сети, то можно использовать архитектуру с разомкнутым кольцом, в которой получаем выходные значения вместо того, чтобы возвращать их на вход сети. У такого подхода есть два преимущества. Во-первых, входные данные такой сети будут более точными. Во-вторых, результирующая сеть имеет просто архитектуру сети обратного распространения, поэтому можно применять более эффективные алгоритмы обучения.

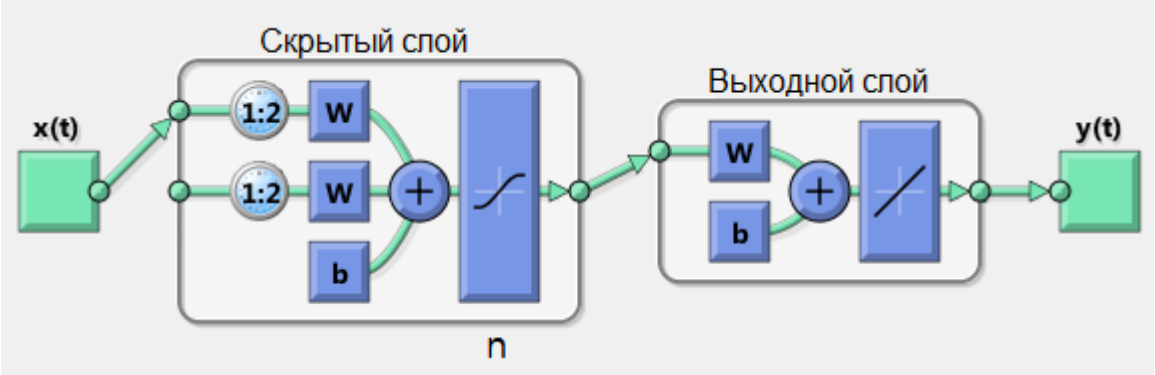

Рисунок 4 – Нейронная сеть с разомкнутым кольцом обратной связи

В сети присутствуют два входа. Один – внешний, а второй – соединен с выходом сети, откуда и получает входные значения. Для каждого из входов имеется линия задержки с отводами для хранения предыдущих значений.

Количество нейронов на скрытом уровне будем устанавливать экспериментально, а число задержек будет равно 2. Прежде чем начать обучение сети, содержащей линии задержки с отводами, необходимо заполнить их начальными значениями входа и выхода.

Теперь осталось выбрать алгоритм обучения нейронной сети. Рассмотрим три алгоритма. Алгоритм Левенберга-Марквардта работает быстро; метод

Байесовской регуляризации занимает больше времени, но может показать лучшие результаты для зашумленных и небольших задач; методу масштабируемых сопряженных градиентов требуется меньше памяти. Суть данных методов будет описана ниже.

Итак, после того, как построили, обучили и протестировали нейронную сеть, можно получить прогноз. Напомним, что обучение и тестирование сети происходит с разомкнутым кольцом. Теперь обратно замкнем кольцо обратной связи, то есть соединим второй вход сети с ее выходом (рисунок 3), чтобы сеть стала способна к многошаговым предсказаниям. Когда кольцо было разомкнуто, то сеть могла предсказать лишь одно будущее значение. Закрывая кольцо, мы будем подавать на вход вновь посчитанные значения *y(t)*, которые были неизвестны ранее, и эти данные помогут получать вновь и вновь новые значения предсказаний.

Каждый раз в результате обучения нейронной сети будут получаться другие решения. Это связано с различными начальными весами и различными способами разделения исходных данных на обучающее, проверяющее и тестовое подмножества. В результате для каждой нейронной сети одним и тем же входным данным будут соответствовать разные выходные значения. Чтобы гарантировать хорошую точность обучения сети, необходимо переобучать ее несколько раз. [13]

2.2 Алгоритм Левенберга-Марквардта

Алгоритм Левенберга-Марквардта представляет из себя разновидность метода Ньютона. Он был разработан для минимизации функций, которые представляют собой суммы квадратов других нелинейных функций (то есть решает задачу наименьших квадратов). Этот алгоритм очень хорошо подходит для обучения нейронной сети, при котором в качестве оптимизации используется среднеквадратичная ошибка. [14]

Чтобы описать данный алгоритм, необходимо иметь представление о базовом методе Ньютона, который минимизирует сумму квадратов.

Очередное приближение считается по формуле:

$$x_{k+1} = x_k - {A_k}^{-1} g_k \,, \tag{3}$$

где $A_k \equiv \nabla^2 F(x)|_{x=x_k}$,
$g_k \equiv \nabla F(x)|_{x=x_k}$.

Полагаем, что *F(x)* представляет из себя сумму квадратов функций:

$$F(x) = \sum_{i=1}^{N} v_i^2(x) = v^T(x)v(x)$$

Тогда j-й элемент градиента находится:

$$[\nabla F(x)]_j = \frac{\partial F(x)}{\partial x_j} = 2\sum_{i=1}^{N} v_i(x)\frac{\partial v_i(x)}{\partial x_j}$$

или в матричной форме:

$$\nabla F(x) = 2J^T(x)v(x), \tag{4}$$

где

$$J(x) = \begin{pmatrix} \frac{\partial v_1(x)}{\partial x_1} & \frac{\partial v_1(x)}{\partial x_2} & \cdots & \frac{\partial v_1(x)}{\partial x_n} \\ \frac{\partial v_2(x)}{\partial x_1} & \frac{\partial v_2(x)}{\partial x_2} & \cdots & \frac{\partial v_2(x)}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial v_N(x)}{\partial x_1} & \frac{\partial v_N(x)}{\partial x_2} & \cdots & \frac{\partial v_N(x)}{\partial x_n} \end{pmatrix} \tag{5}$$

Матрица Гессе (k,j) определяется:

$$[\nabla^2 F(x)]_{k,j} = \frac{\partial^2 F(x)}{\partial x_k \partial x_j} = 2\sum_{i=1}^{N}\left\{\frac{\partial v_i(x)}{\partial x_k}\frac{\partial v_i(x)}{\partial x_j} + v_i(x)\frac{\partial^2 v_i(x)}{\partial x_k \partial x_j}\right\}$$

или в матричной форме:

$$\nabla^2 F(x) = 2J^T(x)J(x) + 2S(x)$$

где

$$S(x) = \sum_{i=1}^{N} v_i(x)\nabla^2 v_i(x)$$

Если значение S(x) мало, то можно аппроксимировать Гессиан следующим образом:

$$\nabla^2 F(x) \cong 2J^T(x)J(x) \tag{6}$$

При подстановке уравнения (4) и (6) в уравнение (3) и получаем метод Гаусса-Ньютона:

$$\begin{aligned} x_{k+1} &= x_k - [2J^T(x_k)J(x_k)]^{-1}2J^T(x_k)v(x_k) = \\ &= x_k - [J^T(x_k)J(x_k)]^{-1}J^T(x_k)v(x_k) \end{aligned} \tag{7}$$

Преимущество метода Гаусса-Ньютона над классическим методом Ньютона заключается в том, что здесь не нужно вычислять вторые производные.

Единственная проблема данного метода состоит в том, что матрица $H = J^T J$ может быть необратимой. Для того чтобы избежать такой ситуации, модифицируется приближение матрицы Гессе:

$$G = H + \mu I$$

Полученная матрица G всегда будет обратимой. Пусть $\{ \lambda_1, \lambda_2, \dots, \lambda_n \}$ – собственные значения матрицы H, а $\{ z_1, z_2, \dots, z_n \}$ – собственные вектора. Тогда:

$$Gz_i = [H + \mu I]z_i = Hz_i + \mu z_i = \lambda_i z_i + \mu z_i = (\lambda_i + \mu)z_i$$

Таким образом, собственные вектора матрицы G совпадают с собственными векторами матрицы H, а собственные значения матрицы G равны $(\lambda_i + \mu)$. Матрица G будет положительно определена за счет увеличения μ, пока не будет получено, что $(\lambda_i + \mu)>0$ для всех i, и, следовательно, полученная матрица G будет обратима.

Вышеописанная модификация приводит к алгоритму Левенберга-Марквардта:

$$x_{k+1} = x_k - [J^T(x_k)J(x_k) + \mu_k I]^{-1} J^T(x_k)v(x_k) \quad (8)$$

$$\Delta x_k = -[J^T(x_k)J(x_k) + \mu_k I]^{-1} J^T(x_k)v(x_k) \quad (9)$$

Полученныйалгоритм обладает очень полезным свойством: при увеличении μ_k алгоритм приближается к алгоритму наискорейшего спуска (10), в то время как уменьшение значения μ_k до нуля приводит к алгоритму Гаусса-Ньютона (7).

$$x_{k+1} \cong x_k - \frac{1}{\mu_k} J^T(x_k)J(x_k) = x_k - \frac{1}{2\mu_k}\nabla F(x), \text{при больших } \mu_k \quad (10)$$

Алгоритм начинается с небольшого значения μ_k (например, $\mu_k = 0.01$). Если на очередном шаге не получается уменьшить значение функции F(x), то шаг повторяется с новым значением μ_k, умноженным на некоторый коэффициент α (например, α = 10). В конце концов, значение функции F(x) будет уменьшаться, так как мы будем делать небольшие шаги в направлении наискорейшего спуска. Если на очередной итерации значение функции F(x) действительно уменьшилось, тогда для следующей итерации значение μ_k делится на α, приближая алгоритм к алгоритму Гаусса-Ньютона, который должен обеспечить более быструю сходимость. Таким образом, алгоритм Левенберга-Марквардта обеспечивает компромисс между скоростью сходимости алгоритма Ньютона и гарантией сходимости алгоритма наискорейшего спуска.

Рассмотрим применение алгоритма Левенберга-Марквардта для обучения многослойной нейронной сети. В процессе обучения минимизируется среднеквадратичная ошибка. Если значения выхода равновероятны, то среднеквадратичная ошибка пропорциональна сумме квадратов ошибок всех выходов Q в обучающем множестве:

$$\begin{aligned} F(x) &= \sum_{q=1}^{Q} (t_q - a_q)^T (t_q - a_q) = \\ &= \sum_{q=1}^{Q} e_q^T e_q = \sum_{q=1}^{Q} \sum_{j=1}^{S^M} (e_{j,q})^2 = \sum_{i=1}^{N} (v_i)^2, \end{aligned} \quad (11)$$

где $e_{j,q}$ – j-й элемент ошибки для q-й пары входа/выхода.

Уравнение (11) эквивалентно оценке качества обучения сети, для минимизации которой был разработан алгоритм Левенберга-Марквардта. Её смысл должен быть понятным независимо от способа обучения сети. Оказывается, это верная концепция, но она требует некоторой осторожности в разработке деталей.

Ключевым шагом в алгоритме Левенберга-Марквардта является вычисление якобиана. Для выполнения этого вычисления используется вариация алгоритма обратного распространения, в классическом варианте которого вычисляются производные квадратов ошибок относительно весов и смещений сети. Для вычисления якобиана необходимо вычислить производные от ошибок, а не от их квадратов.

Прежде чем представить саму процедуру вычисления якобиана, необходимо более детально изучить его форму (5). Вектор ошибок представляется в виде:

$$v^T = [v_1\ v_2 \dots v_N] = \left[e_{1,1}\ e_{2,1} \dots e_{S^M,1}\ e_{1,2} \dots e_{S^M,Q}\right], N = Q \times S^M \qquad (12)$$

а вектор параметров:

$$x^T = [x_1\ x_2 \dots x_n] = \left[w^1_{1,1}\ w^1_{1,2} \dots\ w^1_{S^1,R}\ b^1_1 \dots\ b^1_{S^1}\ w^2_{1,1} \dots\ b^M_{S^M}\ \right], \qquad (13)$$

где $n = S^1(R+1) + S^2(S^1+1) + \dots + S^M(S^{M-1}+1)$.

Подставлением (12) и (13) в уравнение (5) получается более подробная запись якобиана:

$$J(x) = \begin{pmatrix} \frac{\partial e_{1,1}}{\partial w^1_{1,1}} & \frac{\partial e_{1,1}}{\partial w^1_{1,2}} & \dots & \frac{\partial e_{1,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,1}}{\partial b^1_1} & \dots \\ \frac{\partial e_{2,1}}{\partial w^1_{1,1}} & \frac{\partial e_{2,1}}{\partial w^1_{1,2}} & \dots & \frac{\partial e_{2,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{2,1}}{\partial b^1_1} & \dots \\ \vdots & \vdots & & \vdots & \vdots & \\ \frac{\partial e_{S^M,1}}{\partial w^1_{1,1}} & \frac{\partial e_{S^M,1}}{\partial w^1_{1,2}} & \dots & \frac{\partial e_{S^M,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{S^M,1}}{\partial b^1_1} & \dots \\ \frac{\partial e_{1,2}}{\partial w^1_{1,1}} & \frac{\partial e_{1,2}}{\partial w^1_{1,2}} & \dots & \frac{\partial e_{1,2}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,2}}{\partial b^1_1} & \dots \\ \vdots & \vdots & & \vdots & \vdots & \end{pmatrix} \qquad (14)$$

Элементы якобиана (14) могут быть вычислены с помощью простой модификации алгоритма обратного распространения по формуле:

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial x_l}$$

В алгоритме обратного распространения используется такое понятие, как чувствительность, которая вычисляется через рекуррентное соотношение от последнего слоя к первому. Та же концепция используется для вычисления условий, необходимых для (14), и чувствительность Марквардта:

$$s_{i,h}^{-m} = \frac{\partial v_h}{\partial n_{i,q}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m},$$

где $h = (q-1)S^M + k$ из (24).

Элемент якобиана вычисляется:

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial w_{i,j}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = s_{i,h}^{-m} \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = s_{i,h}^{-m} \times a_{j,q}^{m-1} \quad (15)$$

Чувствительность Марквардта может быть вычислена через те же рекуррентные соотношения, как и стандартная чувствительность (16) с единственной модификацией на последнем слое. Уравнение для последнего слоя в случае чувствительности Марквардта записано в формуле (17).

$$s^m = F^m(n^m)(W^{m+1})^T s^{m+1} \quad (16)$$

$$\begin{aligned} s_{i,h}^{-M} = \frac{\partial v_h}{\partial n_{i,q}^M} = \frac{\partial e_{k,q}}{\partial n_{i,q}^M} = \frac{\partial (t_{k,q} - a_{k,q}^M)}{\partial n_{i,q}^M} = -\frac{\partial a_{k,q}^M}{\partial n_{i,q}^M} \\ = \begin{cases} -f^M\left(n_{i,q}^M\right), & \text{для } i = k \\ 0, & \text{для } i \neq k \end{cases} \end{aligned} \quad (17)$$

Когда на вход сети подается вектор входных значений p_q и на выходе сети получаются соответствующие вычисленные значения a_q^M, то алгоритм обратного распространения с помощью метода Левенберга-Марквардта инициализируется:

$$S_q^M = -F^M\left(n_q^M\right), \quad (18)$$

$$F^m(n^m) = \begin{vmatrix} f^m(n_1^m) & 0 & \cdots & 0 \\ 0 & f^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & f^m(n_{S^m}^m) \end{vmatrix}$$

Каждый столбец матрицы S_q^M должен пройти по алгоритму обратного распространения, используя формулу (16), чтобы получить одну строку в исходном якобиане. Также столбцы можно распространить обратно с помощью формулы:

$$S_q^m = F^m(n_q^m)(W^{m+1})^T S_q^{m+1} \quad (19)$$

Конечная матрица чувствительностей Марквардта создается путем расширения матриц, вычисленных для каждого входа:

$$S^m = [S_1^m | S_2^m | \ldots | S_Q^m] \quad (20)$$

Вектор чувствительности будем вычислять для каждого входа, так как нам нужно вычислить производные непосредственно от каждой ошибки, а не от суммы квадратов ошибок. Каждому входу, подаваемому нейронной сети, будут

соответствовать свои ошибки (своя для каждого элемента выходного вектора), а каждой из этих ошибок будет соответствовать одна строка в якобиане. После вычисления чувствительности Марквардта, элементы якобиана вычисляются по формуле (15).

Теперь с учетом вышеописанного, можем сформулировать итерационный алгоритм обратного распространения Левенберга-Марквардта.

1) Для каждого входа нейронной сети вычисляются соответствующие выходы и их ошибки $e_q = t_q - a_q^M$. Вычисляется сумма квадратов ошибок F(x) по всем входам (11).

2) Вычисляется матрица Якоби (14). По рекуррентным формулам (19) вычисляется чувствительность, которая инициализируется значением (18). Полученная матрица дополняется значениями чувствительности Марквардта (20). Вычисляются элементы матрицы Якоби (15).

3) Решается уравнение (9), для получения Δx_k.

4) Пересчитывается сумма квадратов ошибок с учетом $x_k + \Delta x_k$. Если полученная сумма меньше той, что была посчитана на первом шаге, то делится μ на α, изменяется значение $x_{k+1} = x_k + \Delta x_k$ и происходит переход к первому шагу. Если же сумма квадратов ошибок не уменьшилась, то μ умножается на α и происходит переход к третьему шагу.

Алгоритм завершает свою работу, когда градиент (15) станет меньше некоторого наперед заданного значения или сумма квадратов ошибок будет в допустимых пределах.

Ключевым недостатком алгоритма является требование большого объема памяти для хранения всех данных, в частности, необходимо запоминать всю матрицу Гессе, тогда как во многих других методах требуется хранить только лишь градиент. Поэтому когда число параметров нейронной сети очень велико, использование алгоритма Леверберга-Марквардта нецелесообразно.

2.3 Байесовская регуляризация

Байесовская регуляризация решает задачу генерализации. [14] Пусть задано множество значений пар входа/выхода:

$$\{p_1, y_1\}, \{p_2, y_2\}, \dots, \{p_Q, y_Q\}$$

В рамках концепции генерализации полагают, что выходы генерируются:

$$y_i = f(p_i) + \varepsilon_i,$$

где f – некоторая неизвестная функция;

ε_i – независимая случайная величина с математическим ожиданием, равным 0 (шум).

Цель – обучить нейронную сеть, игнорируя шум. Качество обучения сети, как и в предыдущем пункте, определяется величиной суммы квадратов ошибок:

$$F(x) = E_D = \sum_{q=1}^{Q} (y_q - a_q)^T (y_q - a_q), \quad (21)$$

где a_q – выход нейронной сети для входа p_q.

Один из методов решения задачи генерализации – это регуляризация. Для этого метода необходимо изменить уравнение (21), добавив к нему штраф (который иногда и называют регуляризацией). Он присутствует в производной аппроксимирующей функции (нейронная сеть в нашем случае), благодаря чему результирующая функция будет гладкой. При определенных условиях можно получить уравнение:

$$F(x) = \beta E_D + \alpha E_W = \beta \sum_{q=1}^{Q} (y_q - a_q)^T (y_q - a_q) + \alpha \sum_{i=1}^{n} x_i^2, \quad (22)$$

где отношение α/β контролирует сложность решения сети: чем больше это отношение, там более гладкой является полученная функция.

Байесовская регуляризация обучает нейронные сети, основываясь на некоторых положениях Байесовской статистики, которые будут описаны ниже. Они полезны во многих аспектах обучения, помимо выбора параметров регуляризации.

Предполагается, что веса сети – случайные величины. Выбираются веса, максимизирующие условные вероятности весов исходных данных. Функция вероятности находится с помощью формулы Байеса:

$$P(x|D, \alpha, \beta, M) = \frac{P(D|x, \beta, M) P(x|\alpha, M)}{P(D|\alpha, \beta, M)}, \quad (23)$$

где x – вектор, содержащий все веса и настройки сети;
D – обучающее множество;
α, β – параметры, связанные с функциями плотности $P(D|x, \beta, M)$ и $P(x|\alpha, M)$;
M – характеризует выбранную модель сети, то есть ее архитектуру (количество слоев и нейронов на каждом слое).

В формуле (23) $P(D|x, \beta, M)$ – это плотность распределения для данных с учетом определения набора весов, параметра и выбранной модели. Если предположить, что шум – независимая случайная величина и имеет нормальное распределение, то получаем формулу:

$$P(D|x, \beta, M) = \frac{1}{Z_D(\beta)} e^{-\beta E_D}, \quad (24)$$

где $\beta = 1/(2\sigma_\varepsilon^2)$;
σ_ε^2 – это дисперсия каждого элемента ε_k;
E_D – квадрат ошибки (21);
$Z_D(\beta)$ – определяется по формуле:

$$Z_D(\beta) = (2\pi\sigma_\varepsilon^2)^{N/2} = (\pi/\beta\,)^{N/2}, \quad (25)$$

где N – это $Q \times S^M$ (11).

Уравнение (24) называется функцией правдоподобия. Это функция весов x сети, показывающая вероятность того, что получится данный набор при установленном значении весов. Метод максимального правдоподобия выбирает весовые коэффициенты таким образом, чтобы максимизировать функцию правдоподобия, что в данном случае означает то же самое, что и минимизация квадрата ошибок E_D. Таким образом, E_D может быть получена статистически с предположением о присутствии нормального шума в обучающем множестве, а стандартный выбор весов определяется оценкой максимального правдоподобия.

Второй множитель числителя в правой части уравнения (23): $P(x|\alpha, M)$ – априорная плотность распределения. Она содержит в себе знания о сети до того, как были собраны какие-либо данные. Байесовская статистика позволяет учесть все априорные знания через данную функцию. Например, если предположить, что весовые коэффициенты являются небольшими, близкими к нулю значениями, то можно выбрать априорную плотность вероятности, определяющуюся:

$$P(x|\alpha, M) = \frac{1}{Z_W(\alpha)} e^{-\alpha E_W},$$

где $\alpha = 1/(2\sigma_W^2)$;
σ_W^2 – это дисперсия весовых коэффициентов;
E_W – квадрат ошибки, определяющийся из (22);
$Z_W(\alpha)$ – определяется:

$$Z_W(\alpha) = (2\pi\sigma_W^2)^{n/2} = (\pi/\alpha\,)^{n/2}, \quad (26)$$

где n – определяется из (13).

Знаменатель в правой части уравнения (23): $P(D|\alpha, \beta, M)$ называется доказательством или коэффициентом нормализации, и он единственный не зависит от x. Если цель – найти веса x, которые максимизируют апостериорную плотность $P(x|D, \alpha, \beta, M)$, тогда нет необходимости концентрировать внимание на $P(D|\alpha, \beta, M)$ (но эта функция будет иметь важное значение в дальнейшем при оценке параметров α и β).

Функция вероятности (23) запишется в виде:

$$P(x|D, \alpha, \beta, M) = \frac{\frac{1}{Z_W(\alpha)}\frac{1}{Z_D(\beta)} e^{-(\beta E_D + \alpha E_W)}}{\text{Коэффициент нормализации}} = \\ = \frac{1}{Z_F(\alpha, \beta)} e^{-F(x)}, \quad (27)$$

где функция $Z_F(\alpha, \beta)$ зависит от α и β и не зависит от x.

Чтобы найти наиболее вероятное значение весовых коэффициентов, необходимо решить задачу максимизации апостериорной плотности

$P(x|D,\alpha,\beta,M)$, что равносильно решению задачи минимизации функции оценки качества регуляризации $F(x) = \beta E_D + \alpha E_W$.

Таким образом, оценка качества регуляризации может быть получена с помощью Байесовской статистики, с предположением о присутствии в обучающем множестве нормально распределенного шума и нормальной априорной плотности распределения весовых коэффициентов сети. Будем определять веса, которые максимизируют апостериорную плотность распределения как x^{MP} (most probable – наиболее вероятные). Они будут противопоставлены весам, максимизирующим функцию правдоподобия – x^{ML}.

Для оценки параметров α и β определяется плотность вероятности $P(\alpha,\beta|D,M)$ по формуле Байеса:

$$P(\alpha,\beta|D,M) = \frac{P(D|\alpha,\beta,M)\,P(\alpha,\beta|M)}{P(D|M)} \tag{28}$$

Формула (28) имеет такой же формат, как и уравнение (23) с функцией правдоподобия и априорной плотностью распределения в числители правой части. Если считать априорную плотность распределения $P(\alpha,\beta|M)$ постоянной (независящей от параметров регуляризации α и β), то максимизация апостериорной плотности распределения будет достигаться за счет максимизации функции правдоподобия $P(D|\alpha,\beta,M)$. Однако эта функция представлена как коэффициент нормализации в уравнении (23). Так как предполагается, что все вероятности имеют нормальное распределение, известно распределение априорной плотности распределения уравнения (23), которое описано в уравнении (27). Поэтому можно выразить коэффициент нормализации из уравнения (27):

$$\begin{aligned} P(D|\alpha,\beta,M) &= \frac{P(D|x,\beta,M)P(x|\alpha,M)}{P(x|D,\alpha,\beta,M)} = \\ &= \frac{\left[\frac{1}{Z_W(\alpha)}e^{-\alpha E_W}\right]\left[\frac{1}{Z_D(\beta)}e^{-\beta E_D}\right]}{\frac{1}{Z_F(\alpha,\beta)}e^{-F(x)}} = \\ &= \frac{Z_F(\alpha,\beta)}{Z_W(\alpha)Z_D(\beta)} \cdot \frac{e^{-(\beta E_D+\alpha E_W)}}{e^{-F(x)}} = \frac{Z_F(\alpha,\beta)}{Z_W(\alpha)Z_D(\beta)} \end{aligned} \tag{29}$$

Константные значения $Z_W(\alpha)$ и $Z_D(\beta)$ известны из уравнений (26) и (25). Неизвестное значение $Z_F(\alpha,\beta)$ можно оценить, используя разложение в ряд Тейлора.

Поскольку целевая функция имеет квадратичную форму в окрестности точки минимума, то разложение в ряд Тейлора в окрестности этой точки, в которой градиент равен нулю, до второго порядка имеет вид:

$$F(x) \approx F(x^{MP}) + \frac{1}{2}(x - x^{MP})^T H^{MP}(x - x^{MP}), \tag{30}$$

где H – матрица Гессе для функции $F(x)$, которая определяется по формуле (31); H^{MP} – значение матрицы Гессе в точке x^{MP}.

$$H = \beta\nabla^2 E_D + \alpha\nabla^2 E_W \tag{31}$$

Полученное приближение (30) подставляется в (27):

$$P(x|D,\alpha,\beta,M) \approx \frac{1}{Z_F} e^{-F(x^{MP})-\frac{1}{2}(x-x^{MP})^T H^{MP}(x-x^{MP})}$$

Или в более удобной форме:

$$P(x|D,\alpha,\beta,M) \approx \left\{\frac{1}{Z_F} e^{-F(x^{MP})}\right\}\left\{e^{-\frac{1}{2}(x-x^{MP})^T H^{MP}(x-x^{MP})}\right\}$$

Приравнивая стандартную формулу нормальной плотности распределения к полученному выражению, имеем:

$$P(x) = \frac{1}{\sqrt{(2\pi)|(H^{MP})^{-1}|}} e^{-\frac{1}{2}(x-x^{MP})^T H^{MP}(x-x^{MP})}, \text{откуда}$$

$$Z_F(\alpha,\beta) \approx (2\pi)^{n/2}\sqrt{\det\left((H^{MP})^{-1}\right)} e^{-F(x^{MP})}$$

Подставляя результат в (29), получаем оптимальные значения параметров α и β в точке минимума:

$$\alpha^{MP} = \frac{\gamma}{2E_W(x^{MP})} \tag{32}$$

$$\beta^{MP} = \frac{N-\gamma}{2E_D(x^{MP})}, \tag{33}$$

где γ – эффективное число параметров:

$$\gamma = n - 2\alpha^{MP}\mathrm{tr}(H^{MP})^{-1}$$

где n – общее число параметров сети.

Байесовская оптимизация параметров регуляризации требует вычисления матрицы Гессе в точке минимума. Можно использовать приближение Гаусса-Ньютона для матрицы Гессе, которое легко получить, если для поиска точки минимума используется алгоритм Левенберга-Марквардта (8). Дополнительные вычисления для оптимизации регуляризации минимальны.

Алгоритм Байесовской регуляризации.

1) Веса инициализируются случайными значениями. Затем вычисляются E_D и E_W. Устанавливается $\gamma = n$ и вычисляются значения параметров α и β по формулам (32–33).
2) Совершается один шаг алгоритма Левенберга-Марквардта минимизации целевой функции $F(x) = \beta E_D + \alpha E_W$.

3) Вычисляется параметр $\gamma = n - 2\alpha \mathrm{tr}(H)^{-1}$. Находится гессиан с помощью приближения Гаусса-Ньютона в алгоритме Левенберга-Марквардта:

$$H = \nabla^2 F(x) \approx 2\beta \mathrm{J}^T \mathrm{J} + 2\alpha \mathrm{I_n},$$

где J – матрица Якоби (14).

4) Вычисляются новые оценки параметров регуляризации по формулам (32–33).

5) Шаги 2–4 повторяются до тех пор, пока алгоритм не сойдется.

Стоит обратить внимание на то, что с каждой новой переоценкой параметров регуляризации α и β, изменяется целевая функция $F(x)$, поэтому точка минимума перемещается. Если перемещение происходит к следующей точке минимума, то новые оценки параметров оказываются более точными. В конце концов, будет достигнута нужная точность и целевая функция перестанет существенно изменяться в последующих итерациях. Таким образом, достигается сходимость.

2.4 Метод масштабируемых сопряженных градиентов

Метод масштабируемых сопряженных градиентов (scaled conjugate gradient) [15] был разработан на основе метода сопряженных градиентов.

Пусть w – матрица весов сети, $F(w)$ – ошибка обучения сети (как и в предыдущих методах – среднеквадратичная ошибка). Пусть $p_1, p_2, \dots, p_k \in R^N$ – множество ненулевых векторов. Говорят, что множество векторов $\{p_i\}_{i=\overline{1,k}}$ образуют сопряженную систему векторов относительно симметричной матрицы A размером NxN, если выполнено условие:

$$p_i^T A p_j = 0, \qquad i \neq j, i = \overline{1,k}, j = \overline{1,k}$$

Таким образом, множество точек $w \in R^N$, удовлетворяющих уравнению (34), называют k-плоскостью или π_k.

$$w = w_i + \alpha_1 p_1 + \cdots + \alpha_k p_k, \ \ \alpha_i \in R, \tag{34}$$

где w_i – точка в пространстве весов;

$p_1, p_2, \dots, p_k$ – подмножество сопряженной системы векторов.

Опишем алгоритм минимизации $F(w)$.

1) Выбирается первоначальное приближение w_1 и установим k=1.

2) Определяется направление поиска p_k и величина шага α_k так, чтобы выполнялось неравенство:

$$F(w_k + \alpha_k p_k) < F(w_k)$$

3) Изменяется вектор $w_{k+1} = w_k + \alpha_k p_k$.

4) Если $F'(w_k) \neq 0$, тогда $k = k + 1$ и осуществляется переход к шагу 2; иначе w_{k+1} – искомый минимум.

Определение следующей точки в этом итерационном процессе состоит из двух независимых действий. Сначала определяется, в каком направлении текущего

весового пространства необходимо двигаться в поисках нового приближения. После того, как это направление поиска будет найдено, нужно определить размер шага.

Метод сопряженных градиентов в своей основе содержит вышеописанный алгоритм минимизации, изменяется лишь способ выбора величины шага. Этот метод предполагает, что задана система сопряженных векторов. Но нет необходимости знать вектора $p_1, p_2, \dots, p_k$ заранее, так как они могут быть определены рекурсивно. Изначально, p_1 равен вектору направления наискорейшего спуска $-F'_{qw}(y_1)$. Затем p_{k+1} рекурсивно определяется как линейная комбинация текущего направления наискорейшего спуска $-F'_{qw}(y_{k+1})$ и предыдущего направления p_k. Если говорить более точно, то p_{k+1} выбирается как ортогональная проекция вектора $-F'_{qw}(y_{k+1})$ на $(N-k)-$ плоскость π_{N-k}, сопряженной с π_k. Для того, чтобы показать, как это работает, приведем формулировки теорем.

Теорема 1. Пусть $p_1, p_2, \dots, p_k$ – сопряженная система векторов и y_1 – точка в пространстве весов, а точки $y_2, y_3, \dots, y_{N+1}$ определяются рекурсивно:

$$y_{k+1} = y_k + \alpha_k p_k,$$

где $\alpha_k = \mu_k / \delta_k$;
$\mu_k = -p_k^T F'_{qw}(y_k)$;
$\delta_k = -p_k^T F''_{qw}(w) p_k$.

Тогда y_{k+1} минимизирует функцию F_{qw}, ограниченную $k-$ плоскостью π_k, заданной для y_1 и $p_1, p_2, \dots, p_k$.

Теорема 2. Пусть y_1 – точка в весовом пространстве, p_1 и r_1 равны вектору наискорейшего спуска $-F'_{qw}(y_1)$. Определим рекурсивно p_{k+1}:

$$p_{k+1} = r_{k+1} + \beta_k p_k,$$

где $r_{k+1} = F'_{qw}(y_{k+1})$,
$\beta_k = (|r_{k+1}|^2 - r_{k+1}^T r_k) / p_k^T r_k$,
y_{k+1} – точка, получаемая из теоремы 1.

Тогда p_{k+1} – это вектор наискорейшего спуска F_{qw}, на $(N-k)-$ плоскость π_{N-k}, сопряженный с π_k, заданный для y_1 и $p_1, p_2, \dots, p_k$.

Сопряженные векторы, полученные с помощью теоремы 2, обычно называют направлениями сопряженных градиентов. В совокупности 1 и 2 теоремы дают нам алгоритм сопряженных градиентов. На каждой итерации происходит квадратичная аппроксимация F_{qw} глобальной функции ошибок F в текущей точке w пространства весов. Если функция ошибок F не квадратичная, то алгоритм может не сойтись за N шагов, в этом случае он перезапускается, т.е. p_{k+1} инициализируется значением текущего направления наискорейшего спуска r_{k+1}. Это означает, что теоремы 1 и 2 справедливы лишь в случае, если функция F является квадратичной. Так же считается, что чем ближе начальное приближение к искомому минимуму, тем точнее будет получено квадратичное приближение

F_{qw} функции ошибок F. На практике, это свойство является адекватным и дает быструю сходимость. Опишем алгоритм сопряженных градиентов, так как метод масштабируемых сопряженных градиентов будет основываться на нем.

1) Выбирается начальное приближение w_1 . Устанавливается $p_1 = r_1 = -F'(w_1),\ k = 1$.
2) Вычисляются s_k и δ_k по формулам:

$$s_k = F''(w_k)p_k$$

$$\delta_k = p_k^T s_k$$

3) Вычисляется величина шага α_k с помощью формул:

$$\mu_k = p_k^T r_k$$

$$\alpha_k = \frac{\mu_k}{\delta_k}$$

4) Совершается шаг в направлении p_k:

$$w_{k+1} = w_k + \alpha_k p_k \tag{35}$$

$$r_{k+1} = -F'(w_{k+1})$$

5) Если $k\ mod\ N = 0$, то алгоритм перезапускается (36), иначе по формулам (37–38) находится новое сопряженное направление.

$$p_{k+1} = r_{k+1} \tag{36}$$

$$\beta_k = \frac{|r_{k+1}|^2 - r_{k+1}^T r_k}{\mu_k} \tag{37}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{38}$$

6) Если направление наискорейшего спуска $r_k \neq 0$, тогда $k = k + 1$ и осуществляется переход к шагу 2. Иначе возвращается w_{k+1} как достигнутый минимум.

Коэффициент β_k может быть получен и с помощью других формул. Вычислительная сложность вышеописанного алгоритма составляет $O(N^3)$ и требуется $O(N^2)$ памяти. Для решения этой проблемы обычно ищут очередное приближение величины шага с помощью линейного поиска. Используя факт, что значение w_{k+1}, полученное по формуле (35) является минимумом для k-плоскости $p_1, p_2, \dots, p_k$ (по теореме 1), можно показать, что выполняется равенство (39), из которого следует, что α_k является решением (40).

$$F'(w_{k+1})p_k = 0 \tag{39}$$

$$\min F(w_k + \alpha p_k) \tag{40}$$

Таким образом, α_k – это минимум функции F вдоль прямой $w_k + \alpha p_k$.

Помимо линейного поиска, существует иной подход приближения шага. Идея заключается в приближении s_k в методе сопряженных градиентов с помощью несимметричной аппроксимации:

$$s_k = F''(w_k)p_k \approx \frac{F'(w_k + \sigma_k p_k) - F'(w_k)}{\sigma_k}, 0 < \sigma_k \leq 1$$

В пределе данное приближение стремится к истинному значению $F''(w_k)p_k$. Вычислительная сложность составляет $O(3N^2)$, а памяти требуется $O(N)$. Если использовать этот подход в сочетании с методом градиентного спуска, то получается алгоритм, применимый непосредственно к нейронной сети прямого распространения. Эта слегка модифицированная версия метода градиентного спуска так же может называться градиентным спуском.

Проблема алгоритма заключается в том, что он работает только для функций с положительно определенной матрицей Гессе, а так же результат работы алгоритма может быть плохим, если начальное приближение очень далеко от желаемого минимума.

Идея решения данной проблемы заключается в объединении методов Левенберга-Марквардта и сопряженных градиентов. Это делается путем введения скаляра λ_k в метод сопряженных градиентов, который должен регулировать неопределенность $F''(w_k)$. Такой результат получим, используя формулу:

$$s_k = \frac{F'(w_k+\sigma_k p_k)-F'(w_k)}{\sigma_k} + \lambda_k p_k$$

Корректировка λ_k на каждой итерации зависит от знака σ_k, который непосредственно показывает, если $F''(w_k)$ не положительно определена. Если $\sigma_k \leq 0$, тогда матрица Гессе не является положительно определенной и λ_k увеличивается и происходит перерасчет s_k. Вводятся обозначения для нового значения s_k как $\hat{s}_k$, а λ_k как $\hat{\lambda}_k$. Тогда справедлива формула:

$$\hat{s}_k = s_k + (\hat{\lambda}_k - \lambda_k)p_k$$

Пусть на данной итерации $\sigma_k \leq 0$.Тогда определяется, насколько нужно увеличить λ_k, чтобы получить $\sigma_k > 0$. Для этого вводится обозначение для σ_k как $\hat{\sigma}_k$ и оценивается значение $\hat{\lambda}_k$:

$$\hat{\sigma}_k = p_k^T \hat{s}_k = p_k^T(s_k(\hat{\lambda}_k - \lambda_k)p_k) = \sigma_k + (\hat{\lambda}_k - \lambda_k)|p_k|^2 > 0 =>$$
$$=> \hat{\lambda}_k > \lambda_k - \frac{\sigma_k}{|p_k|^2}$$

Получается, что если λ_k увеличится более чем на $-(\sigma_k/|p_k|^2)$, то $\hat{\sigma}_k$ станет больше 0. Но до сих пор нет ответа на вопрос, насколько нужно увеличить λ_k, чтобы получить оптимальное решение. Тем не менее, ясно, от каких величин $\hat{\lambda}_k$ зависит. Таким образом, разумно считать $\hat{\lambda}_k$ по формуле:

$$\hat{\lambda}_k = 2(\lambda_k - \frac{\delta_k}{|p_k|^2}) \quad (41)$$

Это приводит к положительному значению $\hat{\sigma}_k$:

$$\hat{\sigma}_k = \sigma_k + (\hat{\lambda}_k - \lambda_k)|p_k|^2 = \sigma_k + (2\lambda_k - 2\frac{\sigma_k}{|p_k|^2} - \lambda_k)|p_k|^2 =$$

$$= -\sigma_k + \lambda_k|p_k|^2 > 0$$

Величина шага считается по формуле:

$$\alpha_k = \frac{\mu_k}{\sigma_k} = \frac{\mu_k}{p_k^T \hat{s}_k + \lambda_k|p_k|^2}$$

Значение λ_k напрямую масштабирует величину шага: чем больше λ_k, тем меньше шаг, что вполне логично.

Аппроксимация квадратичной функции, для которой запускается алгоритм, не всегда может показывать хорошие результаты, так как λ_k масштабирует матрицу Гессе искусственным путем. А механизм изменения значения λ_k позволяет добиться хорошей аппроксимации даже тогда, когда матрица Гессе положительно определена.

Получается алгоритм масштабирования сопряженных градиентов.

1) Выбирается вектор весов w_1, величины $0 < \sigma \leq 10^{-4}$, $0 < \lambda_1 \leq 10^{-6}$, $\hat{\lambda}_k = 0$. Устанавливаются значения $p_1 = r_1 = -F'(w_1)$, $k = 1$ и success = true.
2) Если success = true, то вычисляются необходимые значения:

$$\sigma_k = \frac{\sigma}{|p_k|}$$

$$s_k = \frac{F'(w_k + \sigma_k p_k) - F'(w_k)}{\sigma_k}$$

$$\delta_k = p_k^T s_k$$

3) δ_k масштабируется по формуле:

$$\delta_k = \delta_k + (\hat{\lambda}_k - \lambda_k)|p_k|^2$$

4) Если $\delta_k \leq 0$, то матрица Гессе преобразуется к положительно определенной с помощью формул (41), (42–43).

$$\delta_k = -\delta_k + \lambda_k|p_k|^2 \quad (42)$$

$$\lambda_k = \hat{\lambda}_k \quad (43)$$

5) Вычисляется величина шага по формулам:

$$\mu_k = p_k^T r_k$$

$$\alpha_k = \frac{\mu_k}{\delta_k}$$

6) Вычисляется параметр сравнения результатов Δ_k:

$$\Delta_k = 2\delta_k \frac{F(w_k) - F(w_k + \alpha_k p_k)}{\mu_k^2}$$

7) Если $\Delta_k \geq 0$, тогда можно уменьшить величину ошибки и продолжить алгоритм:

$$w_{k+1} = w_k + \alpha_k p_k$$

$$r_{k+1} = -F'(w_{k+1})$$

$$\hat{\lambda}_k = 0$$

Значение success = true. Если $k \; mod \; N = 0$, то алгоритм перезапускается (44), иначе вычисляются значения по формулам (45–46).

$$p_{k+1} = r_{k+1} \tag{44}$$

$$\beta_k = \frac{|r_{k+1}|^2 - r_{k+1}^T r_k}{\mu_k} \tag{45}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{46}$$

Если $\Delta_k \geq 0.75$, то изменяется значение параметра λ_k:

$$\lambda_k = \frac{1}{4}\lambda_k$$

Если $\Delta_k < 0$, то success = false и $\hat{\lambda}_k = \lambda_k$.

8) Если $\Delta_k < 0.25$, тогда значение λ_k увеличивается:

$$\lambda_k = \lambda_k + \delta_k \frac{(1 - \Delta_k)}{|p_k|^2}$$

9) Если направление наискорейшего спуска $r_k \neq 0$, тогда $k = k + 1$ и переходим к шагу 2. Иначе возвращаем w_{k+1} как достигнутый минимум.

Величина σ должна быть настолько маленькой, насколько это возможно, принимая во внимание точность вычислительной машины. При сохранении σ маленьких значений ($\leq 10^{-4}$) экспериментально показано, что значение σ не оказывает существенного влияния на результаты работы алгоритма масштабируемых сопряженных градиентов. Таким образом, одно из преимуществ алгоритма заключается в том, что он не содержит параметров, от выбора которых зависит успешность работы алгоритма.

Если установить значение $\lambda_k = 0$, то получим обычный метод сопряженных градиентов.

Выводы по разделу

Для решения поставленной задачи была выбрана нелинейная авторегресионная модель нейронной сети с внешними входами и определены некоторые параметры настройки сети. В частности было зафиксировано разбиение данных тестовой выборки на три подмножества: 80% будут составлять обучающее подмножество, 10% – проверочное подмножество и оставшиеся 10% будут тестировать обученную сеть. Помимо этого, была определена величина задержки, равная 2.

Для обучения выбранной модели нейронной сети были выбраны три алгоритма обучения. Главным преимуществом алгоритма Левенберга-Марквардта является его быстрая сходимость. К недостаткам можно отнести необходимость в большом объеме памяти для хранения всех данных. Байесовская регуляризация хотя и сильно уступает по времени работы предыдущему алгоритму, тем не менее, может показать более точные результаты за счет игнорирования шума. Из достоинств метода масштабируемых сопряженных градиентов стоит выделить отсутствие параметров, влияющих на успешность работы алгоритма.

3 ОЦЕНКА ЭФФЕКТИВНОСТИ ПРОГНОЗИРОВАНИЯ ВРЕМЕННОГО РЯДА С ПОМОЩЬЮ НЕЙРОННОЙ СЕТИ NARX

3.1 Получение прогнозов

Целью данной работы является оценка эффективности прогнозирования временного ряда посредством нейронной сети. Для достижения цели необходимо написать программу, моделирующую нелинейную авторегрессионную модель нейронной сети с внешними входами, описанную в п. 2.1 данной работы, которая обучается с помощью алгоритмов Левенберга-Марквардта, Байесовской регуляризации и метода масштабируемых сопряженных градиентов, которые были описаны в п. 2.2–2.4 данной работы. Результат работы программы должен предоставлять нам прогнозируемые значения.

Нейронная сеть была реализована в среде MATLAB с использованием встроенного программного пакета Neural Network Toolbox. Код программы представлен в приложении 1. Стоит отметить, что полученная программа строит прогноз для конкретного алгоритма обучения и конкретного количества нейронов в скрытом слое. Хотя, для решения задачи, код был модифицирован так, чтобы вычислять наилучший прогноз для нескольких вариантов архитектур нейронной сети (то есть для разного числа нейронов в скрытом слое). Но, по сути, достаточно привести пример программы для получения одного прогноза в одном конкретном случае, поэтому был предоставлен именно такой вариант. Схема алгоритма программы приведена на рисунке 5. Для функционирования программы после ее запуска нужно определить количество нейронов в скрытом слое и выбрать один из предложенных алгоритмов обучения. После нажатия кнопки «Показать прогноз» будут представлены графики полученных прогнозов.

В качестве временного ряда взяты данные о цене на недвижимость в Челябинской области с 2000 по 2015 гг. с сайта Федеральной службы государственной статистики [1] (см. приложение 2). В нашем распоряжении средняя цена за 1 кв.м первичного и вторичного жилья за четыре квартала каждого года из вышеуказанного диапазона. С помощью нейронной сети необходимо получить предполагаемые значения цены в кварталы 2015 года на первичное и вторичное жилье и сравнить их с исходными данными, чтобы оценить качество прогноза. Таким образом, в качестве обучающей выборки использовались данные за 2000-2014 года. Так как процесс обучения традиционно можно разделить на три этапа, то исходные данные поделятся следующим образом: 80% отвечают за обучение нейронной сети, 10% служат для проверки нейронной сети и оставшиеся 10% – тестируют полученную сеть. Полученный прогноз на 2015-2020 гг. содержит известные данные за 2015 год, они дают возможность оценить качество полученного прогноза.

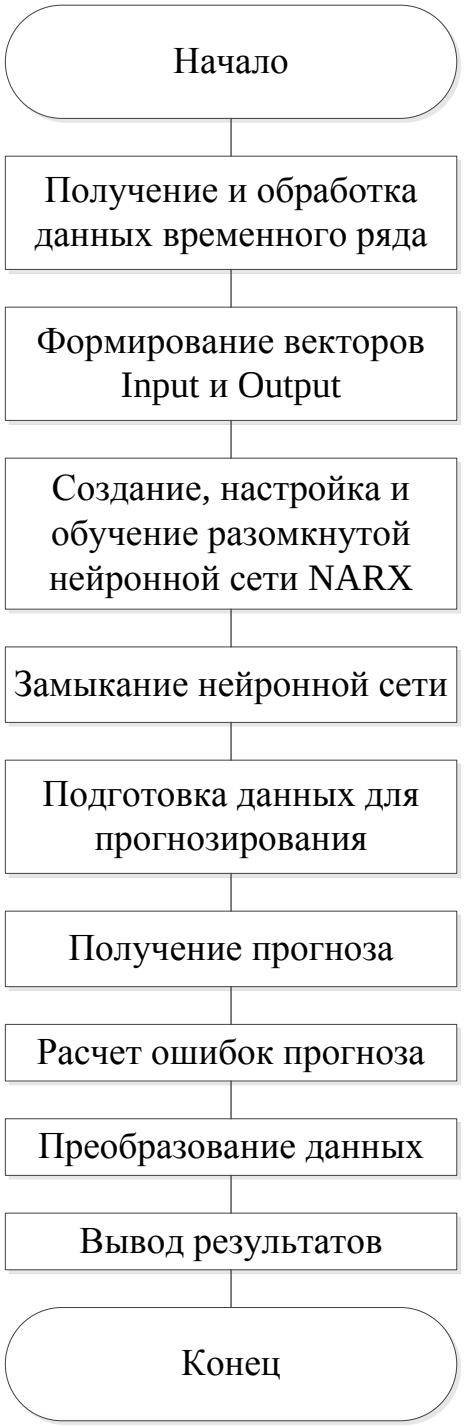

Рисунок 5 – Схема алгоритма программы

Прежде, чем приступить к обучению, необходимо нормировать все данные:

$$\tilde{x} = \frac{x}{M + const},$$

где $\tilde{x}$ – нормированный элемент временного ряда;
x – элемент исходного ряда;
M – значение максимального элемента временного ряда;
$const$ – константа, в нашем случае равная 500.

На вход нейронной сети подаются номера по порядку (1, 2, …), обозначающие порядковый номер момента времени, для которого известны значения цен. На выходе получаются значения цен на первичную и вторичную недвижимость, соответствующие поданному на вход номеру момента времени.

Так как нам необходимо сравнить несколько нейронных сетей, точнее, сравнить прогнозы, полученные с помощью них, то опишем подробнее различия сетей и принцип выбора наилучшего прогноза.

Первое различие нейронных сетей заключается в алгоритме обучения. Это три вышеуказанных метода (Левенберга-Марквардта, Байесовской регуляризации и масштабируемых сопряженных градиентов). Каждый из них имеет свои преимущества и недостатки, но мы сравним их с точки зрения качества полученного прогноза.

Второе различие уже теперь трех нейронных сетей будет заключаться в разном количестве нейронов в скрытом слое. В данной работе были взяты следующие значения: 3, 6, 9, 12, 15, 18, 21 и 24. Итого, получаем, что для каждой из трех сетей будет получено 8 прогнозов. Из них выберем наиболее точный для каждого алгоритма обучения, чтобы потом сравнить их между собой. Критерий точности определим чуть позже.

В конечном счете будет получено 24 различных прогноза, из которых целесообразно выбрать один самый лучший.

Опишем критерий выбора наиболее точного прогноза в рамках нейронных сетей с разным числом нейронов в скрытом слое и одинаковым алгоритмом обучения. Для каждой нейронной сети считался средний квадрат ошибки результатов (далее - ошибка) обучения сети p и полученного прогноза:

$$p = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{47}$$

Среди 8 полученных прогнозов для каждого из алгоритмов обучения выберем тот, для которого ошибка p минимальна. И три полученных прогноза будем сравнивать между собой. Для этого будем сравнивать уже средние абсолютные ошибки P:

$$P = \frac{1}{n}\sum_{i=1}^{n}\frac{|x_i - \bar{x}|}{x_i}\cdot 100\% \tag{48}$$

И среди трех прогнозов выберем тот, у которого абсолютная ошибка прогноза будет минимальна. Будем считать выбранный прогноз наилучшим. А характеристики сети (алгоритм обучения и количество нейронов в скрытом слое) можно будет считать оптимальными для данного временного ряда.

Необходимо обратить внимание на еще один важный нюанс. Как говорилось ранее, нейронная сеть каждый раз инициализируется от разных значений весов и по-разному разделяет данные обучающей выборки на три подгруппы, поэтому на выходе всякий раз будут получаться разные прогнозы. Для решения этой

проблемы целесообразно получать несколько прогнозов для каждой из 24-х сетей и среди них выбрать наиболее точный, то есть сравнивать ошибки p (можно сравнивать как ошибки, полученные на тестовой выборке, так и ошибки, полученные при получении прогнозируемых значений) и оставить для дальнейших исследований только те прогнозы, где были достигнуты минимальные ошибки. Такой подход позволит избежать различного рода выбросов и получить более точные данные о качестве прогнозирования с использованием сетей с различным количеством нейронов в скрытом слое и обучающихся с помощью разных алгоритмов обучения.

Для каждой сети было получено по 50 прогнозов, среди которых сохранялись только те результаты, где ошибка прогноза достигала минимального значения.

3.2 Сравнение полученных результатов

Нейронная сеть была протестирована для разного количества нейронов в скрытом слое: 3, 6, 9, 12, 15, 18, 21 и 24, а также с использованием различных алгоритмов обучения – Левенберга-Марквардта, с помощью Байесовской регуляризации и методом масштабируемых сопряженных градиентов. Для удобства, представим полученные результаты в виде таблиц 1-3.

Напомним, что на выходе из сети получаются нормированные данные, поэтому необходимо с помощью формулы (49) преобразовать выходные значения цен для дальнейшего сравнения. В таблицах представлены уже преобразованные значения цен.

$$x = \tilde{x} \cdot (M + const) \tag{49}$$

Таблица 1

Результаты обучения нейронной сети с помощью алгоритма Левенберга-Марквардта

Количество нейронов в скрытом слое	3	6	9	12	15	18	21	24	Фактическая цена
Ошибка обучения (47)	6.75E-04	4.56E-04	3.22E-04	3.98E-04	2.84E-04	3.30E-04	3.32E-04	3.19E-04	
Ошибка прогноза (47)	0.0015	2.23E-04	3.77E-04	4.27E-04	1.97E-04	1.42E-04	2.54E-04	2.22E-04	
Первичное жилье									
1 квартал	38780	39052	38667	37759	39793	39499	39897	39870	39416.1
2 квартал	38500	38486	38579	37755	37605	38322	37992	39349	38770.71
3 квартал	37995	37972	37639	37220	37117	38182	38979	38222	38055.95
4 квартал	37361	37302	37460	37331	37313	36674	37288	37662	37541.88

Окончание таблицы 1

Количество нейронов в скрытом слое	3	6	9	12	15	18	21	24	Фактическая цена
Вторичное жилье									
1 квартал	34937	35659	35218	35905	35975	35905	36155	35954	36618.35
2 квартал	35314	36455	36327	36045	37407	36761	37206	37620	37860.26
3 квартал	35625	37261	36887	37219	38639	38130	38024	38430	38926.31
4 квартал	35861	37919	37909	37705	39177	38839	37975	38049	38667.62
Ошибка прогноза (48), в %	3.7547	1.8803	2.3446	2.9158	0.9256	1.1591	0.7654	0.2292	

Таблица 2

Результаты обучения нейронной сети с помощью алгоритма с использованием Байесовской регуляризации

Количество нейронов в скрытом слое	3	6	9	12	15	18	21	24	Фактическая цена
Ошибка обучения (47)	5.96E-04	4.53E-04	3.18E-04	2.41E-04	2.73E-04	2.81E-04	2.50E-04	2.73E-04	
Ошибка прогноза (47)	0.004	0.001	0.0033	5.92E-04	2.16E-04	7.81E-04	0.0012	5.38E-04	
Первичное жилье									
1 квартал	39797	39665	40050	38916	38090	39459	39719	38561	39416.1
2 квартал	39762	39539	40650	37973	38307	39251	39340	38110	38770.71
3 квартал	39659	39365	40904	38203	37382	38874	39195	38297	38055.95
4 квартал	39565	39250	41319	38481	37214	38821	39341	38379	37541.88
Вторичное жилье									
1 квартал	34637	35606	35400	35561	35824	35565	35458	35522	36618.35
2 квартал	34639	36516	36110	36275	36850	36523	36160	36497	37860.26
3 квартал	34597	37161	36684	36796	37256	36962	36775	36907	38926.31
4 квартал	34552	37629	37272	37281	37767	37280	37352	37209	38667.62
Ошибка прогноза (48), в %	2.8278	0.3682	0.8277	2.0830	2.3433	1.0208	0.8229	2.0843	

Таблица 3

Результаты обучения нейронной сети с помощью метода масштабируемых сопряженных градиентов

Количество нейронов в скрытом слое	3	6	9	12	15	18	21	24	Фактическая цена
Ошибка обучения (47)	0.0016	8.14E-04	0.0025	4.62E-04	5.26E-04	6.93E-04	0.0011	6.73E-04	
Ошибка прогноза (47)	0.0037	0.0015	0.0015	4.56E-04	0.0013	3.33E-04	0.0012	1.00E-03	
Первичное жилье									
1 квартал	39250	39112	38925	38503	38646	39362	39965	39674	39416.1
2 квартал	39150	38553	38840	38343	38586	38871	39539	39110	38770.71
3 квартал	39394	38571	38075	38175	38598	38172	39285	38924	38055.95
4 квартал	39696	38851	37845	38061	38641	37238	39616	39255	37541.88
Вторичное жилье									
1 квартал	34553	35315	35534	35274	35160	35432	35671	35575	36618.35
2 квартал	34475	35441	35638	36271	35569	36280	36931	35892	37860.26
3 квартал	34658	35968	35728	37003	36053	37182	38188	36927	38926.31
4 квартал	34890	35951	35606	37718	36427	38038	40060	37740	38667.62
Ошибка прогноза (48), в %	3.2012	2.6467	3.1603	2.1281	2.6735	1.7270	1.1109	0.9024	

Критерием для выбора структуры нейронной сети выступила ошибка полученного прогноза. Как видно из таблиц (1-3), в случае обучения сети алгоритмом Левенберга-Марквардта минимальная ошибка достигается при 18 нейронах в скрытом слое (p = 0.000142), с использованием Байесовской регуляризации – при 15 нейронах (p = 0.000216) и методом масштабируемых сопряженных градиентов – при 18 нейронах (p = 0.000333). Таким образом, последующие прогнозы строились с помощью трех отобранных нейронных сетей.

Стоит сказать о трудоёмкости трех выбранных алгоритмов обучения. В данных примерах нейронная сеть с помощью алгоритма Левенберга-Марквардта обучилась за 13 итераций, с использованием Байесовской регуляризации – за 739 итераций, и при обучении методом масштабируемых сопряженных градиентов – за 34 итерации. Эти результаты вполне предсказуемы. Таким образом, очевиден недостаток второго алгоритма обучения, так как он обучает нейронную сеть на порядок дольше в сравнении с двумя оставшимися алгоритмами обучения.

Для выбора наилучшего прогноза были рассчитаны средние абсолютные ошибки для каждой выбранной нейронной сети. Результаты представлены в таблице 4.

Таблица 4

Значение средних абсолютных ошибок полученного прогноза

Алгоритм обучения	Средняя абсолютная ошибка, %
Левенберга-Марквардта	1.418885
С использованием Байесовской регуляризации	2.332807
Метод масштабируемых сопряженных градиентов	1.87917

Таким образом, ошибки прогнозов, полученных в каждом из трех случаев, не превышают 2,4%. А наименьшая ошибка прогноза оказалась у нейронной сети с 18 нейронами в скрытом слое, обученной с помощью алгоритма Левенберга-Марквардта, поэтому в качестве итогового прогноза целесообразно взять значения, полученные на выходе именно этой нейронной сети (таблицы 5–6). На рисунках 5–6 можно наблюдать динамику цен с 2000 по 2020 гг. для первичного и вторичного жилья соответственно. Все данные, представленные на графиках, были получены на выходе нейронной сети.

Таблица 5

Прогноз цен на первичное жилье

Год / Квартал	2015	2016	2017	2018	2019	2020
I	39499	37386	37290	39244	42104	45525
II	38322	35392	35916	38704	41495	44909
III	38182	37157	38004	40635	43738	47396
IV	36674	35091	37272	40065	43099	46941

Таблица 6

Прогноз цен на вторичное жилье

Год / Квартал	2015	2016	2017	2018	2019	2020
I	35905	40499	43539	45000	47106	49370
II	36761	40787	43328	45088	47214	49412
III	38130	42451	44210	46011	48233	50424
IV	38839	42310	44145	46139	48315	50436

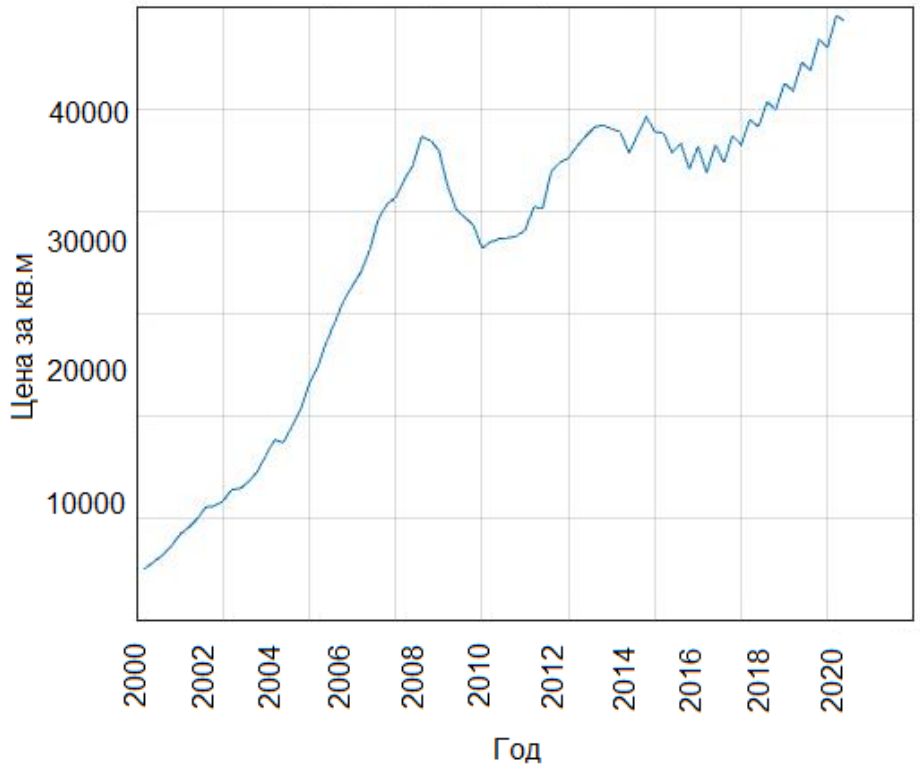

Рисунок 6 – Прогноз цен на первичное жилье, полученный на выходе нейронной сети при обучении алгоритмом Левенберга-Марквардта

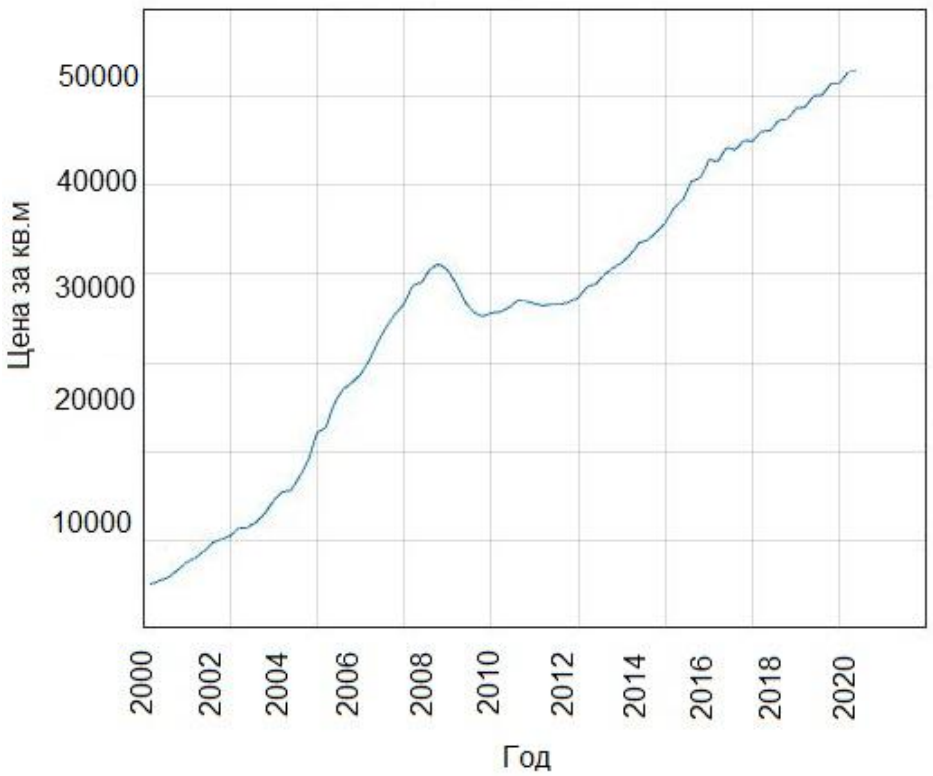

Рисунок 7 – Прогноз цен на вторичное жилье, полученный на выходе нейронной сети при обучении алгоритмом Левенберга-Марквардта

Выводы по разделу

Анализ различного числа нейронов в скрытом слое и алгоритмов обучения показал, что наилучший прогноз получается при 18 нейронах в скрытом слое в случае обучения алгоритмом Левенберга-Марквардта, при 15 нейронах – с использованием Байесовской регуляризации и при 18 нейронах при обучении методом масштабируемых сопряженных градиентов. В результате сравнения трех вышеуказанных сетей и их прогнозов, было выявлено, что нейронная сеть, обученная алгоритмом Левенберга-Марквардта, оказывается менее трудоемкой и показывает наименьшую ошибку прогноза. Таким образом, был получен прогноз цен на первичное и вторичное жилье в Челябинской области на 2015-2020 гг. Ошибка прогноза составила 1,4%, что свидетельствует об эффективности применения нелинейной авторегрессионной модели нейронной сети с внешними входами для прогнозирования временных рядов, так как полученный прогноз получается довольно точным.

Была разработана программа в среде Matlab с использованием встроенного программного пакета Neural Network Toolbox, моделирующая нелинейную авторегрессионную модель нейронной сети с внешними входами. Результатами работы программы являются графики спрогнозированных временных рядов.

ЗАКЛЮЧЕНИЕ

Для решения конкретной задачи прогнозирования временного ряда с помощью нелинейной авторегресионной модели нейронной сети в ходе работы была выполнена настройка сети для получения наилучших результатов. Были выбраны алгоритмы обучения и значения количества нейронов в скрытом слое. Разработана программа в среде Matlab, моделирующая нелинейную авторегрессионную модель нейронной сети с внешними входами. Результатами работы программы являются графики спрогнозированных временных рядов. С помощью нее был получен прогноз цен на первичную и вторичную недвижимость в Челябинской области на 2015-2020 гг. с помощью нейронной сети. Ошибка прогноза составила 1,4%, что говорит о высокой точности полученного прогноза. Это показывает эффективность применения нелинейной авторегрессионной модели нейронной сети открытыми входами для прогнозирования временного ряда.

Самый лучший прогноз получен на выходе нейронной сети с 18 нейронами в скрытом слое и обученной по алгоритму Левенберга-Марквардта. Две другие нейронные сети показали результаты не намного хуже: ошибка прогноза не превышает 2,5%. Но, если оценивать вычислительную сложность алгоритмов обучения, то очевидно стоит отдать предпочтение алгоритму Левенберга-Марквардта. Поэтому и с точки зрения качества прогноза, и с точки зрения эффективности, для рассматриваемой задачи нейронная сеть, обучаемая с помощью вышеуказанного алгоритма, имеет явное преимущество.

Полученную ошибку прогноза (1,4%) можно считать приемлемой, если сравнивать с ошибками, полученными в исследованиях, результаты которых приведены в обзоре. Высокая точность объясняется удачным выбором нейронной сети. Во многих работах использовался простой многослойный перцептрон. Поэтому можно утверждать, что выбранная нами нелинейная авторегресионная модель нейронной сети с внешними входами больше подходит для прогнозирования временного ряда. Это объясняется тем, что многослойный перцептрон может хорошо решать множество различных задач, а сеть NARX предназначена именно для получения прогноза. Поэтому узкая специализация сети позволяет получать высококачественный прогноз.

Алгоритм Левенберга-Марквардта является одним из наилучших алгоритмов, который используется при обучении нейронной сети для решения задачи прогнозирования с использованием минимума входной информации.

Еще одним важным заключением данной работы является то, что удалось получить довольно точный прогноз, используя минимум входной информации. Имеется в виду то, что как описывалось ранее, на вход сети подается лишь время, и все особенности ряда (тренд, циклическая и случайная составляющая), а так же внешние факторы, влияющие на поведение временного ряда, не учитываются. И, тем не менее, полученный прогноз говорит о том, что нейронная сеть способна восстанавливать функцию без дополнительной информации, которая зачастую

может быть вовсе неизвестна. В этом заключается несомненное преимущество нейронной сети по сравнению с другими способами прогнозирования. Но в данном случае следует говорить о краткосрочном прогнозировании. Это обусловлено первым делом отсутствием какой-либо вспомогательной информации, помимо времени, которая, безусловно, повысила бы точность полученного прогноза. А во-вторых, сеть получает прогнозируемые значения последовательно, то есть вновь полученный прогноз зависит, в том числе, и от предыдущих полученных значений, которые не могут быть гарантированно точными. Поэтому с каждой итерацией, ошибка получаемого прогноза увеличивается. А первые прогнозируемые значения получаются с высокой точностью.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Федеральная служба государственной статистики [Электронный ресурс]: офиц. сайт. – М., 1999 – 2016. – Режим доступа: http://www.gks.ru/, свободный. – Загл. с экрана (дата обращения: 05.04.2016).

2 Pany, K.P. Dynamic electricity price forecasting using local linear wavelet neural network / K.P. Pany, S.P. Ghoshal // Neural Comput & Applic. – 2015. – №26. – P. 2039–2047.

3 Dmitriev, A.V. Time series prediction of morbidity using artificial neural networks / A.V. Dmitriev, V.V. Kotin // Biomedical Engineering. – 2013. – V. 47, №1. – P. 43–45.

4 Strnad, D. Neural network models for group behavior prediction: a case of soccer match attendance / D. Strnad, A. Nerat, S. Kohek // Neural Comput & Applic. – 2015.

5 Erdil, A. The prediction of meteorological variables using artificial neural network / A. Erdil, E. Arcaklioglu // Neural Comput & Applic. – 2013. – №22. – P. 1677–1683.

6 Hodge, V.J. Short-term prediction of traffic flow using a binary neural network / V.J. Hodge, R. Krishnan, J. Austin, J. Polak, T. Jackson // Neural Comput & Applic. – 2014. – №25. – P. 1639–1655.

7 Stojadinovic, S. Prediction of flyrock launch velocity using artificial neural networks / S. Stojadinovic, N. Lilic, I. Obradovic, R. Pantovic, M. Denic // Neural Comput & Applic. – 2016. – №27. – P. 515–524.

8 Эконометрика: учебник / И.И. Елисеева [и др]; под. ред. И.И. Елисеевой. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2005. – 576 с.

9 Чучуева, И.А. Модель прогнозирования временных рядов по выборке максимального правдоподобия: дис. … канд. техн. наук: 05.13.18 / Чучуева Ирина Александровна. – М., 2012. – 155 с.

10 Haykin, S. Neural networks and learning machines / S. Haykin. – 3rd ed. – Pearson Education, 2009. – 937 p.

11 Кувшинов, Б.М. Нейронные сети: учебное пособие / Б.М. Кувшинов. – Челябинск: Издательский центр ЮУрГУ, 2015. – 66 с.

12 Заенцев, И.В. Нейронные сети. Основные модели: учебное пособие / И.В. Заенцев. – Воронеж: ВГУ, 1999. – 76 с.

13 The MathWorks. Documentation [Электронный ресурс]: офиц. сайт. – 1994 – 2016. – Режим доступа: http://www.mathworks.com/help/, свободный. – Загл. с экрана (дата обращения: 21.03.2016).

14 Neural Network Design / M.T. Hagan. – 2nd ed. – MA: PWS, 1996. – 1012 p.

15 Moller, M.F. A scaled conjugate gradient algorithm for fast supervised / M.F. Moller // Neural Networks. – 1993. – V. 6. – P. 525-533.

ПРИЛОЖЕНИЕ 1
Текст программы

```
%Функция time_ser_pred
%   На вход подается количество нейронов в скрытом слое
%   и информация об алгоритме обучения.
%   Функция возвращает спрогнозированный временной ряд
function My = time_ser_pred (nwr_numb,alg)
% Временной ряд
TS1=[4222.13    3748.41
4411.66     3784.05
4637.5      3883.22
5073.7      3950.98
5097.36     4757.15
5552.22     5193.09
6159.37     5480.41
6874.68     6129.81
8473.76     7072.18
9300.57     8083.51
9513.42     8603.75
10124.87    9008.09
10270.67    9008.43
10477.23    9215.28
10729.8     9875.76
11556.45    10902.75
13768.46    11125.29
13987.95    11907.67
14767.61    12673.86
15592.05    13839.2
17066.32    16095.23
17629.71    16403.69
20655.39    19041.4
22885.7     21779.45
23014.65    21138.43
24899.08    21939.62
26276.05    23340.24
28174.65    25221.2
30166.05    25890.41
31393.94    27183.96
32053.17    28477.76
33419.83    30484.72
34704.53    30805.48
37376.9     31995.24
38350.23    33491.23
38320.71    33323.25
36579.19    32936.49
34630.39    30833.52
33327.18    29036.58
```

```
32279.71    28320.58
31004.54    29309.24
30165.74    28807.47
30565.26    29477.99
31374.57    30515.89
28508.45    29024.19
28923.22    29458.71
31025.23    29118.12
32332.21    29879.83
35186.31    29300.73
35096.29    29119.52
35765.17    29663.04
33957.09    29644.41
39183.61    31194.06
39498.76    31588.99
39114.31    31996.09
39493.1     32747.62
39349.35    33491.97
37636.11    34718.18
39155.85    34821.74
40099.67    35055.86
39416.1     36618.35
38770.71    37860.26
38055.95    38926.31
37541.88    38667.62];

%Нормируем временной ряд
MM = max(max(TS1))+500;
TS1 = TS1 ./MM;

%Выделяем number - мощность обучающего множества
pred  = 4;
number = size(TS1,1)-pred;

%Формируем векторы входных и выходных значений
%Input - входные значения, подаваемые на вход нейронной сети
%Output - выходные значения, которые должны получиться на
выходе
Input = zeros(1, number);
Output = zeros(2, number);
for i = 1:(number)
    Input(:,i) = i;
    Output(:,i) = TS1(i,:)';
end
```

```
%Устанавливаем максимальные величины ошибок обучения и
тестирования
%для получения наилучшего прогноза
Mperformance1=1000;
Mperformance=1000;
for ttt=1:10
    X = tonndata(Input,true,false);
    T = tonndata(Output,true,false);
    % Выбираем алгоритм обучения
    % 'trainlm' - алгоритм Левенберга-Марквардта
    % 'trainbr' Байесовская регуляризация
    % 'trainscg' - метод масштабируемых сопряженных градиентов
    if alg == 1
        trainFcn = 'trainlm';
    end
    if alg == 2
        trainFcn = 'trainbr';
    end
    if alg == 3
        trainFcn = 'trainscg';
    end

    %Создаем NARX сеть
    inputDelays = 1:2;
    feedbackDelays = 1:2;
    hiddenLayerSize = nwr_numb;
    net =
narxnet(inputDelays,feedbackDelays,hiddenLayerSize,'open',train
Fcn);

    %Подготавливаем данные временного ряда так,
    %чтобы они удовлетворяли особенностям
    %построенной нейронной сети,
    %с помощью функции 'preparets',
    %которая сдвигает время на минимальную величину
    %и заполняет входные состояния и слои нейронной сети
    [x,xi,ai,t] = preparets(net,X,{},T);

    %Определяем долю обучающего,
    % проверочного и тестового множеств
    net.divideParam.trainRatio = 80/100;
    net.divideParam.valRatio = 10/100;
    net.divideParam.testRatio = 10/100;

    %Обучаем нейронную сеть
    [net,tr] = train(net,x,t,xi,ai);
```

```
    %Тестируем нейронную сеть
    y = net(x,xi,ai);
    e = gsubtract(t,y);
    performance = perform(net,t,y);

    %Замыкаем нейронную сеть
    netc = closeloop(net);
    netc.name = [net.name ' - Closed Loop'];

    %Получаем прогноз
    [xc,xic,aic,tc] = preparets(netc,X,{},T);
    yc = netc(xc,xic,aic);
    x1 = X;
    t1 = T(1:number);

    %val_pred - количество прогнозируемых значений
    val_pred = 24;
    x2 = num2cell([number+1:number+val_pred]);
    t2 = squeeze(t1(1:pred));
    for i=1:pred
       trr=TS1(number+i,:)';
       t2(i)={[trr]};
    end
    [x,xi,ai,t] = preparets(net,x1,{},t1);
    [y1,xf,af] = net(x,xi,ai);
    [netc,xi,ai] = closeloop(net,xf,af);
    [y2,xf,af] = netc(x2,xi,ai);
    YNew = zeros(2,size(y1,2)+size(y2,2));
    YNew(:,1:size(y1,2))=cell2mat(y1);
    YNew(:,size(y1,2)+1:size(y1,2)+size(y2,2))=cell2mat(y2);
    performance1 = perform(netc,y2(1:pred),t2);
    if performance1<Mperformance1
        Mperformance1=performance1;
        Mperformance=performance;
        My=YNew;
    end
end

%Выводим полученные результаты
figure(1);
plot  (1:size(My,2),My(1,:),1:number+2,TS1(3:number+pred,1),'o'),
grid on;
figure(2);
plot (1:size(My,2),My(1,:)), grid on;
title('Прогноз цен на первичное жилье');
xlabel('Квартал по порядку');
```

```
ylabel('Относительное значение цены');
figure(3);
plot  (1:size(My,2),My(2,:),1:number+2,TS1(3:number+pred,2),'o'),
grid on;
figure(4);
plot (1:size(My,2),My(2,:)), grid on;
title('Прогноз цен на вторичное жилье');
xlabel('Квартал по порядку');
ylabel('Относительное значение цены');

%Преобразование полученных значений к нормальному виду
My=My*MM;

end
```

ПРИЛОЖЕНИЕ 2
Исходные данные цен на недвижимость

Таблица 2.1

Год	Квартал	Цены на первичное жилье	Цены на вторичное жилье
2000	1	4222,13	3748,41
	2	4411,66	3784,05
	3	4637,5	3883,22
	4	5073,7	3950,98
2001	1	5097,36	4757,15
	2	5552,22	5193,09
	3	6159,37	5480,41
	4	6874,68	6129,81
2002	1	8473,76	7072,18
	2	9300,57	8083,51
	3	9513,42	8603,75
	4	10124,87	9008,09
2003	1	10270,67	9008,43
	2	10477,23	9215,28
	3	10729,8	9875,76
	4	11556,45	10902,75
2004	1	13768,46	11125,29
	2	13987,95	11907,67
	3	14767,61	12673,86
	4	15592,05	13839,2
2005	1	17066,32	16095,23
	2	17629,71	16403,69
	3	20655,39	19041,4
	4	22885,7	21779,45
2006	1	23014,65	21138,43
	2	24899,08	21939,62
	3	26276,05	23340,24
	4	28174,65	25221,2
2007	1	30166,05	25890,41
	2	31393,94	27183,96
	3	32053,17	28477,76
	4	33419,83	30484,72
2008	1	34704,53	30805,48
	2	37376,9	31995,24
	3	38350,23	33491,23
	4	38320,71	33323,25
2009	1	36579,19	32936,49
	2	34630,39	30833,52

Продолжение приложения 2

Окончание таблицы 2.1

Год	Квартал	Цены на первичное жилье	Цены на вторичное жилье
	3	33327,18	29036,58
	4	32279,71	28320,58
2010	1	31004,54	29309,24
	2	30165,74	28807,47
	3	30565,26	29477,99
	4	31374,57	30515,89
2011	1	28508,45	29024,19
	2	28923,22	29458,71
	3	31025,23	29118,12
	4	32332,21	29879,83
2012	1	35186,31	29300,73
	2	35096,29	29119,52
	3	35765,17	29663,04
	4	33957,09	29644,41
2013	1	39183,61	31194,06
	2	39498,76	31588,99
	3	39114,31	31996,09
	4	39493,1	32747,62
2014	1	39349,35	33491,97
	2	37636,11	34718,18
	3	39155,85	34821,74
	4	40099,67	35055,86
2015	1	39416,1	41618,35
	2	38770,71	40860,26
	3	38055,95	40926,31
	4	37541,88	38667,62

Все цены указаны за 1 кв.м.