ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования. В последнее время большое распространение получили различные сервисы-агрегаторы, объединяющие данные из нескольких источников определённой тематики в один, что уменьшает затрачиваемое пользователем время на поиск необходимой информации.

Для обработки запроса от одного пользователя, агрегатору может потребоваться совершить десятки или даже сотни запросов к сторонним ресурсам. В силу того, что пользователи могут запрашивать одну и ту же информацию, самым распространённым способ снижения времени ответа является использование промежуточных буферов, так называемых кэшей. Но данный способ приводит к снижению актуальности предоставляемых агрегатором данных и может быть использован только для предупреждения избыточной нагрузки системы. Следовательно, разработка алгоритма распределения нестационарной нагрузки, определяющего источник данных агрегатора, является актуальной задачей. Однако для её решения необходим анализ или прогноз нагрузки системы.

Степень теоретической разработанности темы. В открытом доступе существует множество научных работ, описывающих методы прогнозирования временных рядов. Но материалы, описывающие проблему выбора метода прогнозирования для разработки алгоритма распределения нестационарной нагрузки, найти не удалось. Из сказанного выше можно сделать вывод о том, что рассматриваемая тема имеет низкую степень теоретической проработанности.

Объектом исследования является прогнозирование нестационарной нагрузки сервиса-агрегатора.

Предметом исследования являются методы прогнозирования временных рядов.

Область исследования. Проведённое исследование методов прогнозирования нагрузки вычислительной системы в пределах разработки алгоритма распределения запросов пользователей полностью соответствует специальности «Вычислительные машины, комплексы, системы и сети», а содержание выпускной квалификационной работы – техническому заданию.

Цель и задачи исследования. Целью работы является улучшение каче-

ства обслуживания пользователей сервиса-агрегатора за счет снижения среднего времени ответа.

Для достижения данной цели были поставлены следующие задачи:

1. Выполнить сравнительный анализ методов прогнозирования временных рядов.
2. Выбрать наиболее адекватный метод прогнозирования.
3. Разработать алгоритм распределения нестационарной нагрузки на основе выбранного метода прогнозирования.
4. Разработать программную реализацию алгоритма.
5. Выполнить сравнение времени ответа исходной и модернизированной систем.

Теоретическую основу исследования составляют научные труды отечественных и зарубежных авторов в области компьютерных технологий и математической статистики.

Методологическую основу исследования составляет эксперимент.

Научная новизна работы заключается в следующем:

1. В настоящее время не существует структурированных рекомендаций по выбору метода прогнозирования при разработке алгоритмов распределения нагрузки, представленных в данной работе.
2. Разработанный алгоритм распределения нестационарной нагрузки может обеспечить улучшение качества обслуживания пользователей сервиса-агрегатора.

Практическая значимость данной работы заключается в том, что разработанный алгоритм распределения нестационарной нагрузки будет интегрирован в разрабатываемый сервис-агрегатор. Кроме того, данный алгоритм может быть интегрирован существующими сторонними сервисами, а сформулированные рекомендации могут быть использованы для осуществления выбора метода

прогнозирования при разработке собственного алгоритма распределения запросов пользователей.

Апробация результатов исследования. Сформулированные рекомендации по выбору метода прогнозирования обсуждались на VII Конгрессе молодых учёных.

Объем и структура работы.

1 СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ПРОГНОЗИРОВАНИЯ

1.1 Модели прогнозирования AR, MA, ARMA, ARIMA

Временной ряд - это ряд наблюдений, произведенных последовательно во времени [1]. Временным рядом можно назвать огромное количество последовательностей, они являются следствием измерений некоторого показателя. Например, показатели (характеристики) экономических, природных, промышленных, информационных и других систем.

Существуют модели прогнозирования, позволяющие на основе доступных к моменту времени t наблюдений спрогнозировать значение временного ряда в определенный момент времени $t + 1$ в будущем. Наибольшее же распространение получила модель ARIMA (autoregressive integrated moving average: интегрированная модель авторегрессии – скользящего среднего, или модель Бокса – Дженкинса). Она удобна еще и тем, что сочетает в себе модели ARMA, AR и MA, их описание представлено в разделе далее.

1.1.1 Модель скользящего среднего MA(q)

В моделях скользящего среднего текущее значение ряда представляется в виде линейной комбинации текущего и прошедших значений ошибки ε_t, ε_{t-1}, ..., ε_{t-q}, по своим свойствам соответствующей «белому шуму». Модель скользящего среднего порядка q может быть выражена следующим уравнением [2]:

$$y_t = \varepsilon_t - \gamma_1\varepsilon_{t-1} - \gamma_2\varepsilon_{t-2} - ... - \gamma_q\varepsilon_{t-q}, \tag{1.1}$$

где:

y_t – значение ряда в момент времени t

$\gamma_1, \gamma_2, ..., \gamma_q$ – параметры модели,

ε_t – случайные ошибки образующие «белый шум».

В эквивалентной форме:

$$y_t = (1 - \gamma_1 B - \gamma_2 B^2 - ... - \gamma_q B^q)\varepsilon_t, \tag{1.2}$$

или:

$$y_t = \gamma(B)\varepsilon_t, \tag{1.3}$$

где B - оператор сдвига назад:

$$By_t = y_{t-1}.$$

То есть, процесс скользящего среднего можно трактовать как выход y_t линейного фильтра с передаточной функцией $\gamma(B)$, на вход которого подается процесс белого шума ε_t [1].

1.1.2 Модель авторегрессии AR(p)

Модель авторегрессии порядка p может быть представлена в следующем виде [2]:

$$y_t = \beta_1 y_{t-1} + \beta_2 y_{t-2} + ... + \beta_p y_{t-p} + \varepsilon_t, \tag{1.4}$$

где $\beta_1, \beta_2, ..., \beta_p$ – параметры модели. Таким образом прогнозируемое значение представляется в виде линейной зависимости от p известных значений временного ряда.

В эквивалентной форме:

$$y_t = \frac{\varepsilon_t}{1 - \beta_1 B - \beta_2 B^2 - ... - \beta_p B^p}, \tag{1.5}$$

или:

$$y_t = \beta^{-1}(B)\varepsilon_t, \tag{1.6}$$

отсюда процесс авторегрессии можно трактовать как выход y_t линейного фильтра с передаточной функцией $\beta^{-1}(B)$, на вход которого подается процесс белого шума ε_t [1].

1.1.3 Модель авторегрессии – скользящего среднего ARMA(p, q)

Конечный процесс авторегрессии может быть представлен как бесконечный процесс скользящего среднего MA(∞) [3], однако с увеличением порядка модели её расчет значительно усложняется и потому если процесс действительно типа AR, то его представление в виде скользящего среднего не может быть экономичным. Точно так же процесс MA не может быть экономично представлен с помощью процесса авторегресии. Чтобы параметризация была более экономичной в модель могут быть включены как члены описывающие скользящее среднее, так и члены моделирующие авторегрессию.

Общий вид авторегрессии – скользящего среднего ARMA(p, q) определяется следующим уравнением [2]:

$$y_t = \beta_1 y_{t-1} + \beta_2 y_{t-2} + ... + \beta_p y_{t-p} + \varepsilon_t - \gamma_1 \varepsilon_{t-1} - \gamma_2 \varepsilon_{t-2} - ... - \gamma_p \varepsilon_{t-p}, \quad (1.7)$$

Как легко заметить, при порядке p равном нулю будет получен процесс MA, а при обнулении порядка q - процесс AR.

Процесс может быть записан в эквивалентной форме:

$$y_t = \frac{1 - \gamma_1 B - \gamma_2 B^2 - ... - \gamma_q B^q}{1 - \beta_1 B - \beta_2 B^2 - ... - \beta_p B^p} \varepsilon_t, \quad (1.8)$$

таким образом смешанный процесс авторегрессии – скользящего среднего можно интерпретировать как выход y_t линейного фильтра, его передаточная функция есть отношение двух полиномов, на вход которого подается белый шум ε_t [1].

1.1.4 Модель авторегрессии – проинтегрированного скользящего среднего ARIMA(p, d, q)

Перечисленные выше модели используются для прогнозирования стационарных процессов. Это такие процессы, свойства которых не зависят от изменения начала отсчета времени. Однако, существуют временные ряды, которые ведут себя так, словно они не имеют фиксированного среднего значения и даже в этом случае они проявляют однородность, и если не учитывать локальный

уровень или, возможно, локальный уровень и тренд, то любая часть временного ряда по своему поведению подобна любой другой части. Описывающие такое однородное нестационарное поведение модели можно получить, сделав предположение, что некая подходящая разность процесса стационарна. Модели, в которых d-я разность есть стационарный смешанный процесс авторегрессии – скользящего среднего называются процессами авторегрессии – проинтегрированного скользящего среднего ARIMA(p, d, q) [1]. Данный процесс может быть представлен уравнением вида:

$$\Delta^d y_t = \beta_1 \Delta^d y_{t-1} + \beta_2 \Delta^d y_{t-2} + ... + \beta_p \Delta^d y_{t-p} + \varepsilon_t - \gamma_1 \varepsilon_{t-1} - \gamma_2 \varepsilon_{t-2} - ... - \gamma_p \varepsilon_{t-p}, \quad (1.9)$$

где Δ^d – оператор разности временного ряда порядка d, например:

$$\Delta y_t = y_t - y_{t-1},$$

для первого порядка;

$$\Delta^2 y_t = \Delta y_t - \Delta y_{t-1} = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2y_{t-1} + y_{t-2},$$

для второго.

1.2 Прогнозирование при помощи нейросетевых методов

Искусственные нейронные сети (далее - нейронные сети) возникли на основе знаний о функционировании нервной системы живых существ. Они представляют собой попытку использования процессов, происходящих в нервных системах для выработки новых технологических решений [4].

Основным элементом обработки информации в нейронной сети является модель нейрона. В его основе лежат [5]:

1. Набор связей (connecting link) или синапсов (synapse), при этом каждый синапс имеет свой вес (weight). Например, если на вход синапса j, который связан с нейроном k, поступает сигнал x_j, то этот сигнал умножается на вес w_{kj}. При этом веса могут иметь как положительное, так и отрицательное значение.

2. Сумматор, который складывает все входные сигналы перемноженные на соответсвующие им веса.

3. Функция активация (activation function), ограничивающая амплитуду выходного сигнала нейрона. Как правило, значение на выходе нейрона лежит в интервале $[0, 1]$ или $[-1, 1]$. Обычно выделяют 3 основных типа активационных функций: единичного скачка, кусочко-линейные и сигмоидальные. Наибольшее распространение получили последние.

Одним из важнейших преимуществ нейронных сетей является возможность обучения. Обучение можно рассмотреть как корректировку весов сети, выполняемую по обучающим примерам (или обучающим данным), в следствие которой сеть меняет свою реакцию на входные воздействия.

Следует отметить, что нейронные сети позволяют получить результат на ранее не виденных примерах данных. Поэтому нейросетевые методы хорошо себя зарекомендовали как средство моделирования динамических систем при неизвестной априори математической модели динамической системы. Существует два базовых метода наделения нейронных сетей свойствами, необходимых для прогнозирования поведения динамических систем: добавление линий задержек и добавление рекуррентных связей. [6]. Оба метода представлены в разделе далее.

1.2.1 Многослойный персептрон с линией задержек

В многослойном персептроне (Multilayer Perceptron, MLP) с линией задержек все нейроны расположены слоями, при этом имеется один входной слой, один выходной и как минимум один скрытый слой. Пример схемы данной сети порядка N и одним скрытым слоем изображен на рисунке 1.1. Сеть содержит нейроны с линейной функцией активации во входном слое и нейроны с сигмоидальной функцией активации в скрытом и выходном слоях. Весовые коэффициенты задаются матрицами $W^{(1)}$ и $W^{(2)}$. На вход нейронная сеть получает текущее значение временного ряда $y(k)$, а так же задержанные значения $y(k-1)$, $y(k-2)$, ... $y(k-N)$, полученные при помощи элементов запаздывания Z^{-1}, Z^{-2},

... Z^{-N}. По полученным данным сеть обучается делать прогноз следующего значения временного ряда $y(k+1)$.

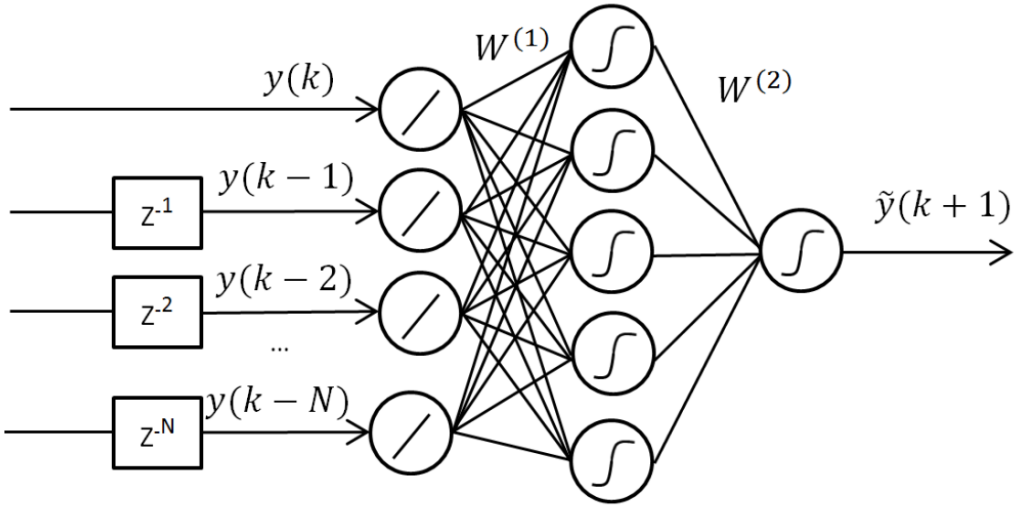

Рисунок 1.1 — Схема многослойного персептрона с линией задержек порядка N.

1.2.2 Рекуррентный многослойный персептрон

Рекуррентный многослойный персептрон (Recurrent Multilayer Perceptron, RMLP) отличается от многослойного персептрона тем, что его выходные значения зависят не только от значений на входе в данный момент времени, но и от предыдущих входных значений или состояния сети. Поэтому данный вид нейронных сетей получил широкое распространение в управляющих приложениях и приложениях направленных на обработку сигналов [7]. Схема персептрона, используемого для прогнозирования временного ряда изображена на рисунке 1.2. Как можно заметить, в матрице весов скрытого слоя $W^{(1)}$ теперь хранятся еще и веса рекуррентных связей.

1.3 Подготовка данных для сравнения

Для построения и тестирования моделей прогнозирования были использованы статистические данные, полученные с разрабатываемого сервиса-агрегатора. Так как сервис разрабатывается на Java, сбор информации по нагрузке процессора осуществлялся при помощи компонента Java платформы com.sun.management.

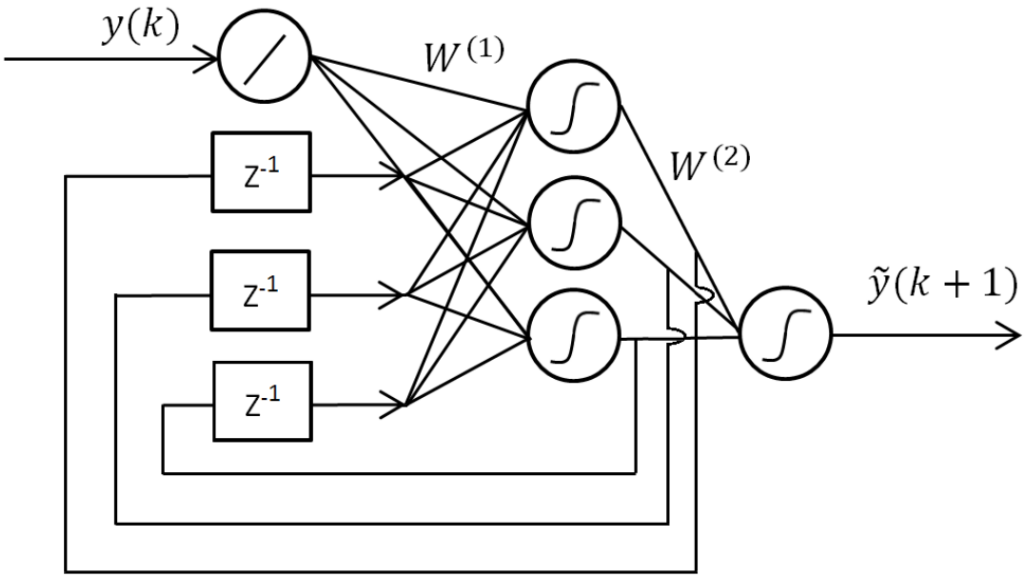

Рисунок 1.2 — Схема рекуррентного многослойного персептрона.

OperatingSystemMXBean. Данный компонент позволяет получить информацию по нагрузке процессора от операционной системы [8].

Каждые 300 мс производился сбор шестнадцати статистических характеристик, которые возможно разделить на четыре типа:

1. **Количество пришедших запросов с момента предыдущего замера.**
   У разрабатываемого сервиса насчитывается шесть видов запросов пользователей, для каждого выделено по одному отдельному счетчику.

2. **Количество загрузчиков внешних ресурсов.**
   При поисковом запросе производится запуск процесса загрузки для необходимых внешних ресурсов. По одной количественной характеристике выделено для каждого внешнего ресурса, всего две.

3. **Среднее время обработки запроса с предыдущего замера по текущий момент времени.**
   Аналогично первому типу – шесть видов запросов.

4. **Нагрузка процессора.**
   Две отдельные характеристики: нагрузка процессом и общая нагрузка процессора.

Графики со статистическими данными изображены на рисунке 1.3.

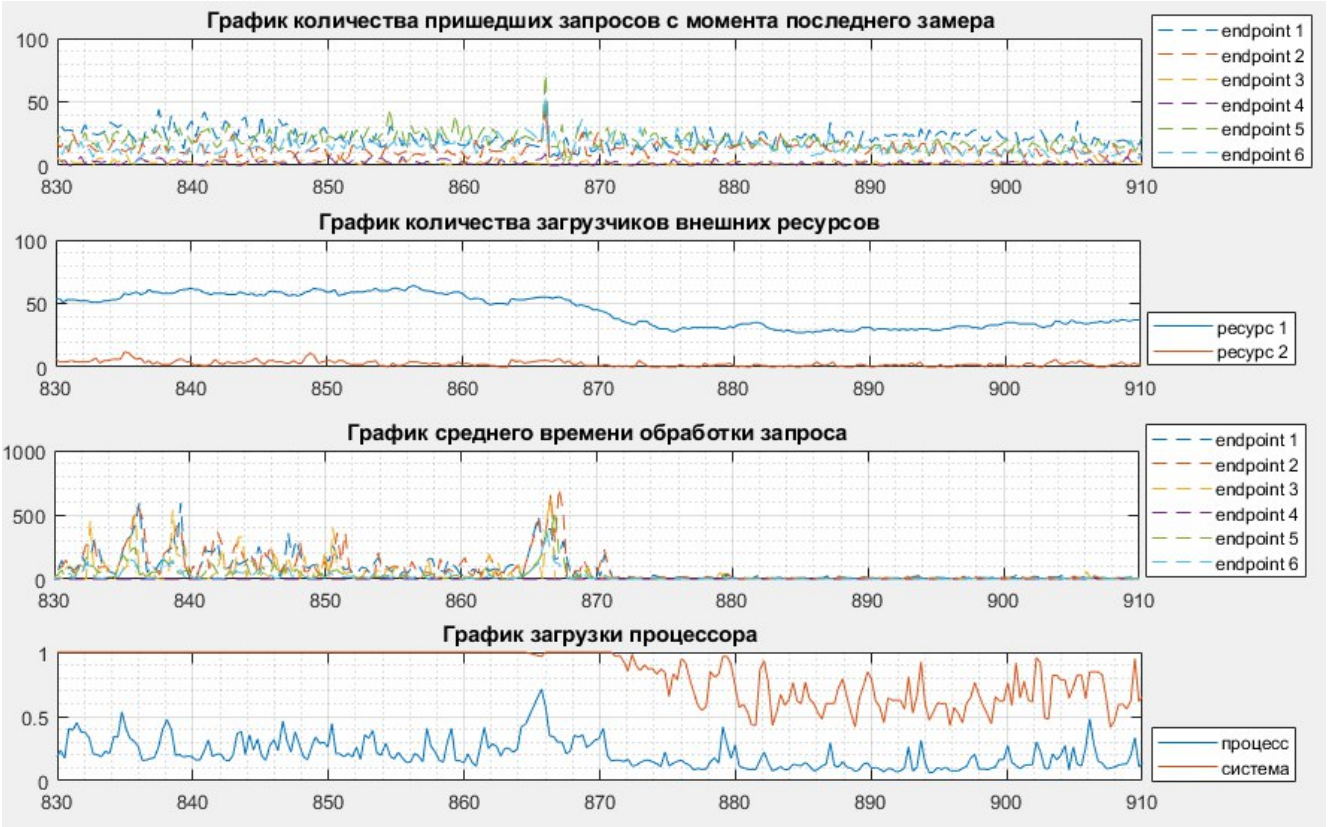

Рисунок 1.3 — Статистические данные сервиса-агрегатора.

1.4 Сравнение моделей прогнозирования ARIMA

Ранее в главе был дан краткий обзор моделей прогнозирования ARIMA, и как можно отметить, один из основных недостатков данных моделей – это необходимость повторного расчета коэффициентов при получении новых данных, что может негативно сказаться на производительности системы (так как значимая доля её процессорного времени будет уделяться данной проблеме). Поэтому порядок моделей и количество обучающих данных были ограничены.

Для расчета ARIMA-моделей была использована Java-библиотека с открытым исходным кодом – Workday/timeseries-forecast. На одних и тех же входных данных по нагрузке процессора, собранных за 50 минут работы системы, были построены модели с порядками $p \leq 11$, $d \leq 3$ и $q \leq 11$. Количество обучающих данных n варьировалось от 10 до 4000.

Оценка точности модели производилась по следующему алгоритму:

- рассчитывалась ARIMA модель с порядками p, d и q на последовательных значениях нагрузки с i-го по $i + n - 1$

- рассчитывался прогноз f следующего шага – $i + n$
- происходила нормализация спрогнозированного значения по следующему правилу: если $f < 0$, то $f_n = 0$; если $f > 1$, то $f_n = 1$
- фиксировалась ошибка – разница между фактическим значением временного ряда и нормализованным значением f_n
- выполненные действия повторялись при $i = i + 1$

После завершения работы данного алгоритма вычислялась средняя квадратическая ошибка (root mean square error RMSE, данный способ оценки точности часто используется для сравнения моделей пронозирования [9; 10]) модели порядков p, d, q с количеством обучающих данных равным n. Результаты, отсортированные в порядке возрастания ошибки, представлены в таблицах 1.1–1.4. Результаты прогнозов наиболее точных моделей AR, MA, ARMA и ARIMA изображены на рисунке 1.4.

Таблица 1.1 — Сравнение моделей MA.

Порядок модели MA (q)	Количество обучающих данных (n)	Средняя квадратическая ошибка
1	15	0.116628
1	16	0.116784
1	11	0.116977
1	13	0.117005
1	18	0.117108
1	20	0.117462
1	19	0.117549
1	14	0.117555
1	17	0.117649
1	22	0.117656

Таблица 1.2 — Сравнение моделей AR.

Порядок модели AR (p)	Количество обучающих данных (n)	Средняя квадратическая ошибка
9	800	0.103868
8	800	0.103885
8	700	0.103948
9	700	0.103956
10	800	0.103992
10	700	0.104136
7	800	0.104277
9	900	0.104283
8	900	0.104333
7	700	0.104349

Таблица 1.3 — Сравнение моделей ARMA.

Порядки модели ARMA (p, q)	Количество обучающих данных (n)	Средняя квадратическая ошибка
8, 1	800	0.104518
9, 1	800	0.104555
10, 1	700	0.104639
9, 1	700	0.104718
8, 1	700	0.104766
10, 1	900	0.104776
10, 1	800	0.104785
9, 2	700	0.104795
9, 3	800	0.104845
9, 1	900	0.104894

Таблица 1.4 — Сравнение моделей ARIMA.

Порядки модели ARIMA (p, d, q)	Количество обучающих данных (n)	Средняя квадратическая ошибка
0, 1, 3	700	0.103599
7, 1, 0	700	0.103604
8, 1, 0	700	0.103613
8, 1, 0	800	0.103665
7, 1, 0	800	0.103714
7, 1, 0	600	0.103752
0, 1, 3	800	0.103774
8, 1, 0	600	0.103777
10, 1, 0	800	0.103798
9, 1, 0	700	0.103804

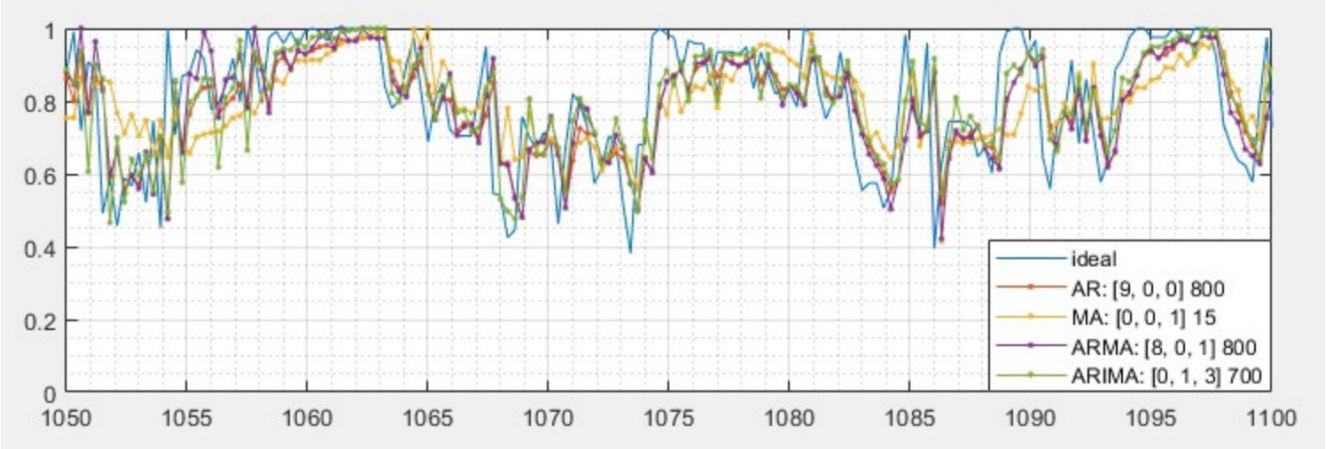

Рисунок 1.4 — Прогноз нагрузки процессора.

На рисунке 1.4 можно отметить, что резкие скачки нагрузки не прогнозируются ни одной из моделей. Кроме того наиболее точные модели вычисляют повторение исходного ряда, но смещенного по времени. По сути значения рассчитанные любой из вышеперечисленных моделей нельзя считать прогнозом.

1.5 Сравнение нейросетевых моделей

Ранее в главе были представлены два типа нейронных сетей, используемых для прогнозирования временных рядов. Следует отметить, что основными их отличиями являются:

1. При использовании многослойного персептрона, в отличие от рекуррентного, необходимо заранее знать порядок линии задержек на входе (в пределах данной работы это не является проблемой, так как порядок может быть выбран в ходе экспериментов, и больше не меняться в ходе работы системы).

2. Рекуррентные сети имеют большую точность при многошаговом прогнозировании [6], однако это не имеет значения в данной работе.

3. Обучение рекуррентных сетей является более трудоемкой задачей, в следствие наличия у них дополнительных степеней свободы [6; 11].

На вход нейронной сети подается шестнадцать динамических характеристик (а также $n - 1$ предыдущих значений каждой из характеристик для сети с линией задержек порядка n). На выходе ожидается бинарное значение: 1 – до следующего замера данные должны браться из кэша; 0 – данные выгружаются непосредственно с внешних ресурсов. Так как при слишком высокой нагрузке повышается среднее время обработки запросов, то в качестве прогнозируемого значения возьмем факт превышения данной характеристикой у любого из шести видов запросов какого-то порогового значения (в пределах данной работы – 90 мс.).

Для составления и обучения нейронной сети был использован Java-фреймворк с открытым исходным кодом – Encog. В качестве обучающего был взят тот же участок данных, что использовался при построении моделей ARIMA в предыдущем разделе.

1.6 Выводы по результатам сравнительного анализа

1. Модели AR, MA, ARMA, ARIMA не подходят для прогнозирования нагрузки процессора, в силу недостаточной точности на данном виде временного ряда.

2. Модель прогнозирования, составленная на базе многослойного персептрона, имеет небольшой процент ошибки на обучающей выборке и будет использоваться при разработке алгоритма распределения нагрузки.

ЗАКЛЮЧЕНИЕ

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Box G. E. P.* Time series analysis: forecasting and control. — Hoboken, New Jersey : John Wiley & Sons, 2008. — ISBN 978-0-470-27284-8.
2. *Рунова Л. П.* Модель авторегрессии и скользящего среднего (ARMA). — Ростов-на-Дону : Издательство ЮФУ, 2013. — 59 с. — ISBN 978-5-699-22086-1.
3. *Hamilton J. D.* Time series analysis. — Princeton, New Jersey : Princeton University Press, 1994. — ISBN 0-691-04289-6.
4. *Осовский С.* Нейронные сети для обработки информации / пер. с пол. И. Д. Рудинский. — М. : Финансы и статистика, 2002. — 344 с. — ISBN 5-279-02567-4.
5. *Haykin S.* Neural networks, a comprehensive foundation. — 2-е изд. — Upper Saddle River, New Jersey : Prentice Hall, 1999. — ISBN 0-13-908385-5.
6. *Чернодуб А. М.* Обучение рекуррентных нейронных сетей методом псевдорегуляризации для многошагового прогнозирования на примере хаотического процесса Маккея-Гласса // Problems of Computer Intellectualization. — Jusautor, Sofia : ITHEA, 2012. — С. 141—151. — ISBN 978-966-02-6529-5.
7. *Medsker L. R.*, *Jain L. C.* Recurrent neural networks: design and applications. — Boca Raton, Florida : CRC Press LLC, 2000. — ISBN 0-8493-7181-3.
8. Официальная документация Oracle [Электронный ресурс] // Oracle Docs: Interface OperatingSystemMXBean. URL: — Режим доступа: `https://docs.oracle.com/javase/7/docs/jre/api/management/extension/com/sun/management/OperatingSystemMXBean.html`, свободный. Дата запроса 14.02.2018.
9. *Liu L. M.* Forecasting and Time Series Analysis Using the SCA Statistical System. Т. 1. — Chicago, Illinois : Scientific Computing Associates Corp., 1994. — ISBN 978-0890355558.
10. *Mills T. C.* The Foundations of Modern Time Series Analysis. — Basingstoke, Hampshire : Palgrave Macmillan, 2011. — ISBN 978-0-230-29018-1.

11. A Comparative Study on CPU Load Predictions in a Computational Grid using Artificial Neural Network Algorithms / S. Naseera [и др.] // Indian Journal of Science and Technology. — 2015. — Т. 8, № 35. — ISSN 0974-5645.