Санкт-Петербургский государственный технический университет
Институт прикладной математики и механики
Кафедра "Телематика (при ЦНИИ РТК)”

# ЛАБОРАТОРНАЯ РАБОТА

Дисциплина: Архитектура систем управления суперкомпьютерных баз данных

Тема: Тройное экспоненциальное сглаживание временных рядов

Выполнила: Стриганова А.В.

Проверил: Попов С.Г.

Санкт-Петербург
2018

# 1. СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ……...3
2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ……...3
2.1 Построение прогнозов временных рядов ..... 3
2.1.1 Интуитивные методы ..... 3
2.1.2 Экспоненциальное сглаживание ..... 4
2.1.3 Двойное экспоненциальное сглаживание ..... 4
2.1.4 Тройное экспоненциальное сглаживание ..... 5
2.2 Обзор Axibase Time Series Database (ATSD) ..... 6
3. РЕЗУЛЬТАТЫ……8
ЗАКЛЮЧЕНИЕ……10
ПРИЛОЖЕНИЕ А. Листинг модуля hw.py…….....11
ПРИЛОЖЕНИЕ Б. Листинг модуля atsd.py……...13

## 2. ЦЕЛЬ РАБОТЫ

В данной работе необходимо рассмотреть алгоритм тройного экспоненциального сглаживания и применить его к метрике, полученной из базы данных временных рядов ATSD. Параметры алгоритма должны быть получены путём минимизации функции ошибки.

## 3. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

### 2.1 Построение прогнозов временных рядов

Тройное экспоненциальное сглаживание, также называемое методом Холта-Винтерса, – это один из способов построения прогнозов временных рядов. Прежде чем приступить к подробному изучению схемы работы этого алгоритма, рассмотрим другие (более простые) методы прогнозирования.

#### 2.1.1 Интуитивные методы

Введём следующие обозначения: y – исходное значение, ŷ – спрогнозированное значение. Для прогноза одного значения можно использовать методы:

1) арифметическое среднее

$$\hat{y}_{i+1} = \frac{1}{N} \sum_{i=1}^{N} y_i$$

2) скользящее среднее

$$\hat{y}_{i+1} = \frac{1}{N} \sum_{i=N-k}^{N} y_i$$

3) взвешенное скользящее среднее

$$\hat{y}_{i+1} = \sum_{i=N-k}^{N} w_i * y_i$$

Взвешенное скользящее среднее кажется наиболее подходящим вари-

антом, но хотелось бы учитывать не только последние N элементов выборки, а все исторические значения ряда, при этом чем старше значение, тем ближе его вес должен быть к нулю. Экспоненциальное сглаживание - решение этой задачи.

2.1.2 Экспоненциальное сглаживание

$$\hat{y}_i = \alpha * y_i + (\alpha - 1) * \hat{y}_{i-1}$$

$\alpha$ — сглаживающий коэффициент

$$\begin{aligned}\hat{y}_i &= \alpha * y_i + (\alpha - 1) * \hat{y}_{i-1} \\ &= \alpha * y_i + \alpha(1-\alpha) * y_{i-1} + (1-\alpha)^2 * y_{i-2} \\ &= \alpha * [y_i + \alpha(1-\alpha) * y_{i-1} + (1-\alpha)^2 * y_{i-2} + (1-\alpha)^3 * y_{i-3} + \ldots + (1-\alpha)^{i-1} * y_1] + (1-\alpha)^i\end{aligned}$$

Веса экспоненциально убывают с течением времени. Чем выше значение α, тем быстрее «забываются» исторические данные. Для каждого ряда нужно подбирать свой сглаживающий коэффициент, этот процесс называется fitting.

Проблема описанных ранее методов заключается в том, что они способны спрогнозировать только одно единственное значение.

2.1.3 Двойное экспоненциальное сглаживание

Введём следующие обозначения:

- *l – level* – одно спрогнозированное значение
- *b – trend* – разница между соседними элементами

$$b = y_i - y_{i-1}$$

Двойное экспоненциально сглаживание – это экспоненциальное сглаживание, применённое и к тренду, и к уровню:

$$\begin{aligned} l_i &= \alpha * y_i + (1-\alpha) * (l_{i-1} + b_{i-1}) && \textit{level} \\ b_i &= \beta * (l_i - l_{i-1}) + (1-\beta) * b_{i-1} && \textit{trend} \\ \hat{y}_{i+1} &= l_i + b_i && \textit{forecast} \end{aligned}$$

$\beta$ – это тренд-фактор; так же, как и для простого экспоненциального сглаживания, $\alpha$ и $\beta$ – отличаются для разных рядов. Данный метод позволяет спрогнозировать два значения.

### 2.1.4 Тройное экспоненциальное сглаживание

Тройное экспоненциальное сглаживание является одним из многих методов, которые могут быть использованы для прогнозирования точек данных в серии, при условии, что серия является “сезонной”, т. е. повторяющейся в течение некоторого периода.

- $L$ – длина сезона (периода)
- $S$ – *seasonal component* – значение, повторяющееся на том же месте в сезонах

В дополнение к уровню и тренду сглаживание применяется так же и к сезонным компонентам:

$$l_i = \alpha * (y_i - s_{i-L}) + (1 - \alpha) * (l_{i-1} + b_{i-1}) \quad level$$
$$b_i = \beta * (l_i - l_{i-1}) + (1 - \beta) * b_{i-1} \quad trend$$
$$s_i = \gamma * (y_i - l_i) + (1 - \gamma) * s_{i-L} \quad seasonal$$
$$\hat{y}_{i+m} = l_i + mb_i + s_{i-L+1+(m-1) mod L} \quad forecast$$

Важно то, что сезонные компоненты сглаживаются по сезонам, т.е. 3-я компонента в данном сезоне, будет сглажена с 3-й компонентой в предыдущем сезоне и т.д. В формуле $\gamma$ – это сглаживающий фактор для сезонной компоненты, $m$ – произвольное целое число. Для реализации нужно знать формулу для подсчёта начального тренда:

$$b_0 = \frac{1}{L}\left(\frac{y_{L+1} - y_1}{L} + \frac{y_{L+2} - y_2}{L} + \ldots + \frac{y_{L+L} - y_L}{L}\right)$$

Это среднее средних трендов по сезонам.

## 2.2 Обзор Axibase Time Series Database (ATSD)

ATSD – это не реляционная база данных, оптимизированная для сбора, хранения и анализа временных данных из ИТ-инфраструктуры, промышленного оборудования, интеллектуальных счетчиков и устройств Интернета вещей. На рисунке 2.2.1 изображена упрощённая архитектура ATSD:

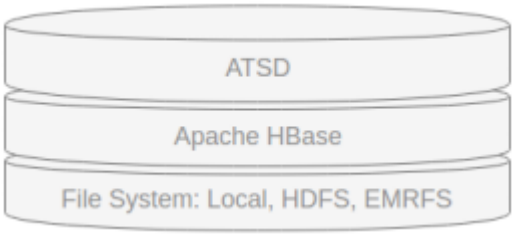


Рис. 2.2.1 Архитектура ATSD.

ATSD состоит из следующих модулей:

- Rule Engine
- SQL Engine
- Portal Server
- Charts Library
- Search Service
- REST API Server
- Network API Server
- CSV Processor

REST API Server предоставляет возможность получения прогнозов, рассчитанных одним из следующих способов: Holt-Winters, ARIMA, SAS, пример запроса представлен на рисунке 2.2.2:

## Request

| Method | Path | `Content-Type` Header |
|---|---|---|
| `POST` | `/api/v1/series/query` | `application/json` |

# Series Query: Named Forecast

## Request

```
[
    {
        "entity": "duckduckgo",
        "metric": "direct.queries",
        "forecastName": "DuckDuckGo1",
        "type": "FORECAST",
        "startDate": "2015-05-01T00:00:00Z",
        "endDate": "2015-07-30T00:00:00Z"
    }
]
```

Рис. 2.2.2 Пример forecast-запроса.

Charts Library позволяет визуализировать как исторические данные, так и спрогнозированные:

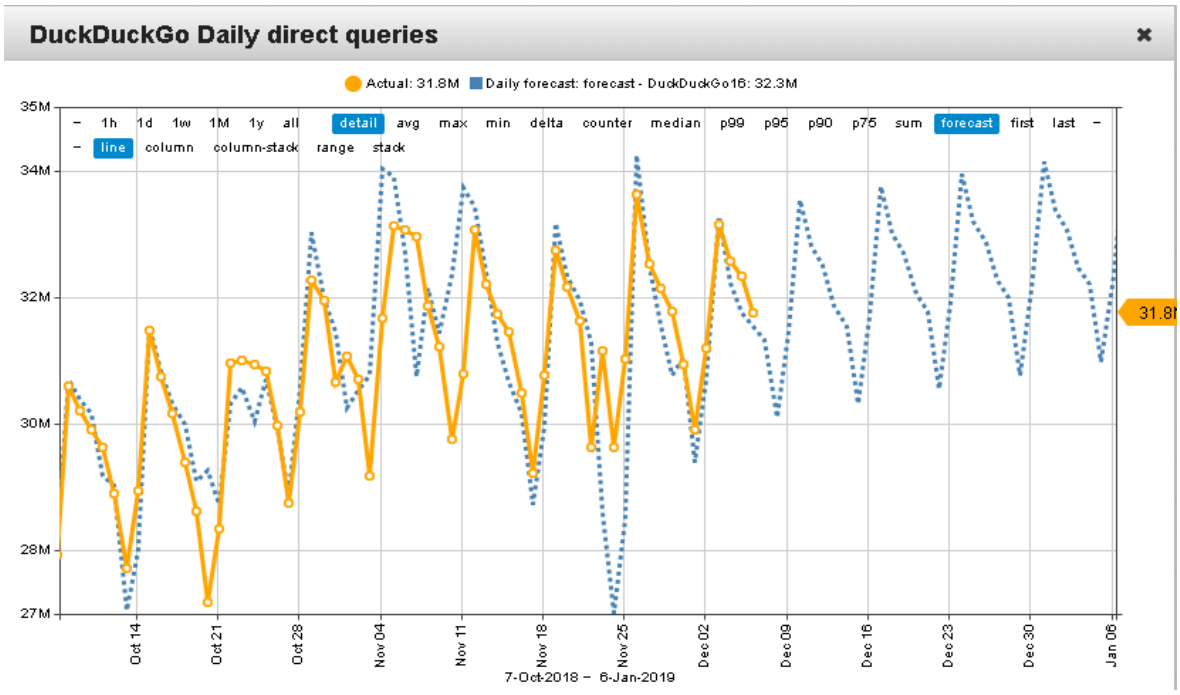


Рис. 2.2.3 Визуализация прогноза с помощью Charts.

# 4. РЕЗУЛЬТАТЫ

В ходе выполнения данной работы была реализована программа на языке Python 3.6, с использованием библиотек Matplotlib 3.0.0, NumPy 1.15.1, SciPy 1.1.0 и ATSD Python Client 3.0.2. Программа состоит из двух модулей:

1. построение прогноза, минимизация суммы квадратов разностей (функции ошибки) и поиск оптимальных параметров $\alpha, \beta, \gamma$;
2. соединение с ATSD, загрузка метрики, вызов метода прогнозирования из 1-го модуля, построение графиков.

В качестве модели используется сумма квадратов разностей:

```
1 # SSE - sum of squared residuals - сумма квадратов разностей
2 def SSE(coefficients, *arr):
3     series = arr[0]
4     season_length = arr[1]
5     n_forecast = arr[2]
6     alpha, beta, gamma = coefficients
7     forecast = holt_winters(series, season_length,
8                             alpha=alpha, beta=beta,
9                             gamma=gamma, n_forecast=n_forecast)
10    sse = 0
11    for i in range(0, len(series)):
12        sse += (forecast[i] - series[i]) ** 2
13    return sse1
```

Листинг 3.1 Функция SSE.

Для поиска оптимальных параметров используется минимизация суммы квадратов разностей с помощью алгоритма L-BFGS-B:

```
1 def forecast(series, season_length, n_forecast):
2     Y = series[:]
3     initial_values = np.array([0.3, 0.1, 0.1])
4     boundaries = [(0, 1), (0, 1), (0, 1)]
5     # минимизируем ошибку с помощью алгоритма L-BFGS-B
6     parameters = fmin_l_bfgs_b(SSE, x0=initial_values,
7                                args=(Y, season_length, n_forecast),
8                                bounds=boundaries,
9                                approx_grad=True)
10    alpha, beta, gamma = parameters[0]
11    print(parameters[0])
12    return holt_winters(series, season_length, alpha, beta, gamma, n_fore-
cast)
```

Листинг 3.2 Функция forecast.

В результате запуска программы получены следующие значения коэффициентов:

$\alpha = 0.01366847$
$\beta = 0.9942451$
$\gamma = 1.0$

На рисунке 3.1 представлены исторические и спрогнозированные с помощью реализованного алгоритма данные:

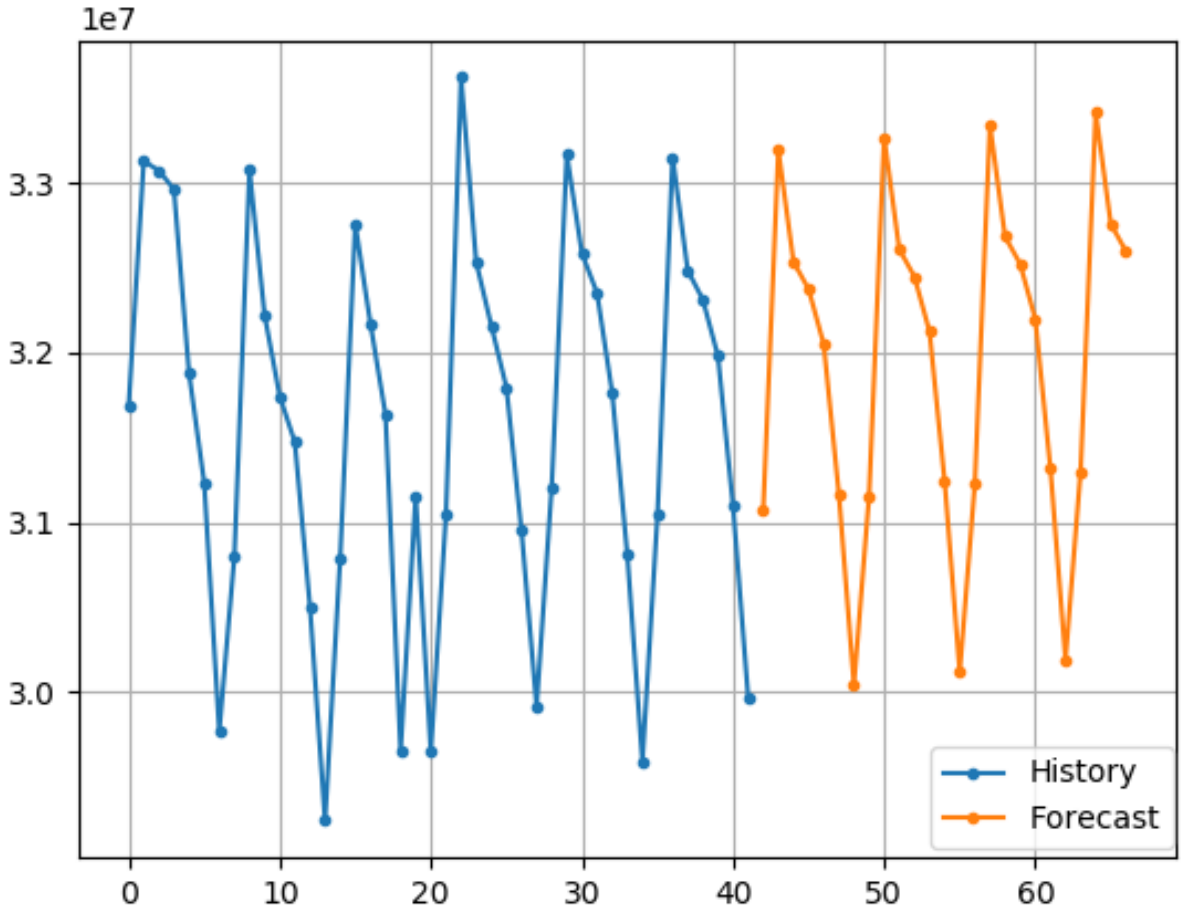


Рис. 3.1 Тестирование программы.

# 5. ЗАКЛЮЧЕНИЕ

В данной работе было изучено и реализовано тройное экспоненциальное сглаживание. Также были рассмотрены простое и двойное экспоненциальное сглаживание. В ходе тестирования полученной программы было выяснено, что метод Холта-Винтерса очень чувствителен к неравномерностям сезонов, а на непериодических данных выдаёт полностью некорректные результаты. Реализованная программа позволяет не задумываться о коэффициентах, т.к. они высчитываются автоматически, однако требуется вручную задавать размер сезона.

# 6. ПРИЛОЖЕНИЕ А. Листинг модуля hw.py

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import fmin_l_bfgs_b


def initial_trend(series, L):
    """
    Начальное значение для тренда.

    :param series: значения ряда
    :param L: длина сезона
    :return: начальное значение для тренда
    """
    sum = 0.0
    for i in range(L):
        sum += float(series[i + L] - series[i]) / L  # среднее средних трендов среди сезонов
    return sum / L


def initial_seasonal_components(series, L):
    """
    Начальные значения для seasonal components - Sx

    :param series: исходные значения ряда
    :param L: длина сезона
    :return: начальные значения для сезонов
    """
    seasonals = {}
    season_averages = []  # средние знячения в сезоне
    n = int(len(series) / L)  # количество сезонов
    for j in range(n):
        season_averages.append(sum(series[L * j:L * j + L]) / float(L))
    # каждое значение в сезоне делится на среднее по сезону,
    # затем полученный результат суммируется с соответствующим значением из следующего сезона,
    # полученная сумма делится на количество сезонов
    for i in range(L):
        sum_of_vals_over_avg = 0.0
        for j in range(n):
            sum_of_vals_over_avg += series[L * j + i] / season_averages[j]
        seasonals[i] = sum_of_vals_over_avg / n
    return seasonals


def holt_winters(series, L, alpha, beta, gamma, n_forecast):
    """
    Тройное экспоненциальное сглаживание.

    :param series: исходный ряд
    :param L: длина сезона
    :param alpha: параметр сглаживания для значения ряда
    :param beta: параметр сглаживания тренда
    :param gamma: сезонный параметр сглаживания
    :param n_forecast: количество точек, которые нужно предсказать
    :return: начальные значения + спрогнозированные
    """
    result = []
    for i in range(len(series) + n_forecast):
        if i == 0:  # начальные значения
            m = 1
            s = initial_seasonal_components(series, L)
```

```
61              level = series[0]
62              trend = initial_trend(series, L)
63              result.append(series[0])
64              continue
65          if i >= len(series):  # прогноз
66              m = i - len(series) + 1  # смещение в прогнозе
67          else:  # HW на известных данных
68              Yi = series[i]
69              last_level = level  # baseline
70              level = alpha * (Yi - s[i % L]) + (1 - alpha) * (level +
trend)
71              trend = beta * (level - last_level) + (1 - beta) * trend
72              s[i % L] = gamma * (Yi - level) + (1 - gamma) * s[i % L]
73          result.append(level + m * trend + s[i % L])
74      return result
75
76
77  # SSE - sum of squared residuals - сумма квадратов разностей
78  def SSE(coefficients, *arr):
79      series = arr[0]
80      season_length = arr[1]
81      n_forecast = arr[2]
82      alpha, beta, gamma = coefficients
83      forecast = holt_winters(series, season_length,
84                              alpha=alpha, beta=beta,
85                              gamma=gamma, n_forecast=n_forecast)
86      sse = 0
87      for i in range(0, len(series)):
88          sse += (forecast[i] - series[i]) ** 2
89      return sse
90
91
92  def forecast(series, season_length, n_forecast):
93      Y = series[:]
94      initial_values = np.array([0.3, 0.1, 0.1])
95      boundaries = [(0, 1), (0, 1), (0, 1)]
96      # минимизируем ошибку с помощью алгоритма L-BFGS-B
97      parameters = fmin_l_bfgs_b(SSE, x0=initial_values,
98                                 args=(Y, season_length, n_forecast),
99                                 bounds=boundaries,
100                                approx_grad=True)
101     alpha, beta, gamma = parameters[0]
102     print(parameters[0])
103     return holt_winters(series, season_length, alpha, beta, gamma,
n_forecast)
```

# 7. ПРИЛОЖЕНИЕ Б. Листинг модуля atsd.py

```
1  from atsd_client import connect
2  from atsd_client.models import *
3  from atsd_client.services import SeriesService
4  import matplotlib.pyplot as plt
5
6  from TDB.hw import forecast
7
8  # В файле connection.properties нужно указать хост, имя и пароль от ATSD
9  connection = connect();
10 # Указываем интересующие нас метрику и сущность, для которых будет стро-
иться прогноз
11 sf = SeriesFilter(metric="direct.queries")
12 ef = EntityFilter(entity="duckduckgo")
13 # Выбираем начальную дату для запроса
14 start_date = "2018-11-03T02:59:00Z"
15 # Инициализируем временной фильтр, конечная дата = начальная дата + 3 ме-
сяца
16 df = DateFilter(interval={"count": 3, "unit": "MONTH"},
start_date=start_date)
17 # Формируем series-запрос
18 query_data = SeriesQuery(series_filter=sf, entity_filter=ef, date_fil-
ter=df)
19 svc = SeriesService(connection)
20 # Загружаем указанный ряд
21 series, = svc.query(query_data)
22 """
23 Визуализируем исторические и спрогнозированные данные.
24 """
25 print(series)
26 start = len(series.values())
27 end = start + 25
28 plt.subplot(111)
29 plt.plot(series.values(), label="History", marker=".")
30 plt.plot(range(start, end), forecast(series.values(), 7, 25)[-25:], la-
bel="Forecast", marker=".")
31 plt.legend(bbox_to_anchor=(1, 0.14), loc=1, borderaxespad=0.1)
32 plt.grid(True)
33 plt.show()
```