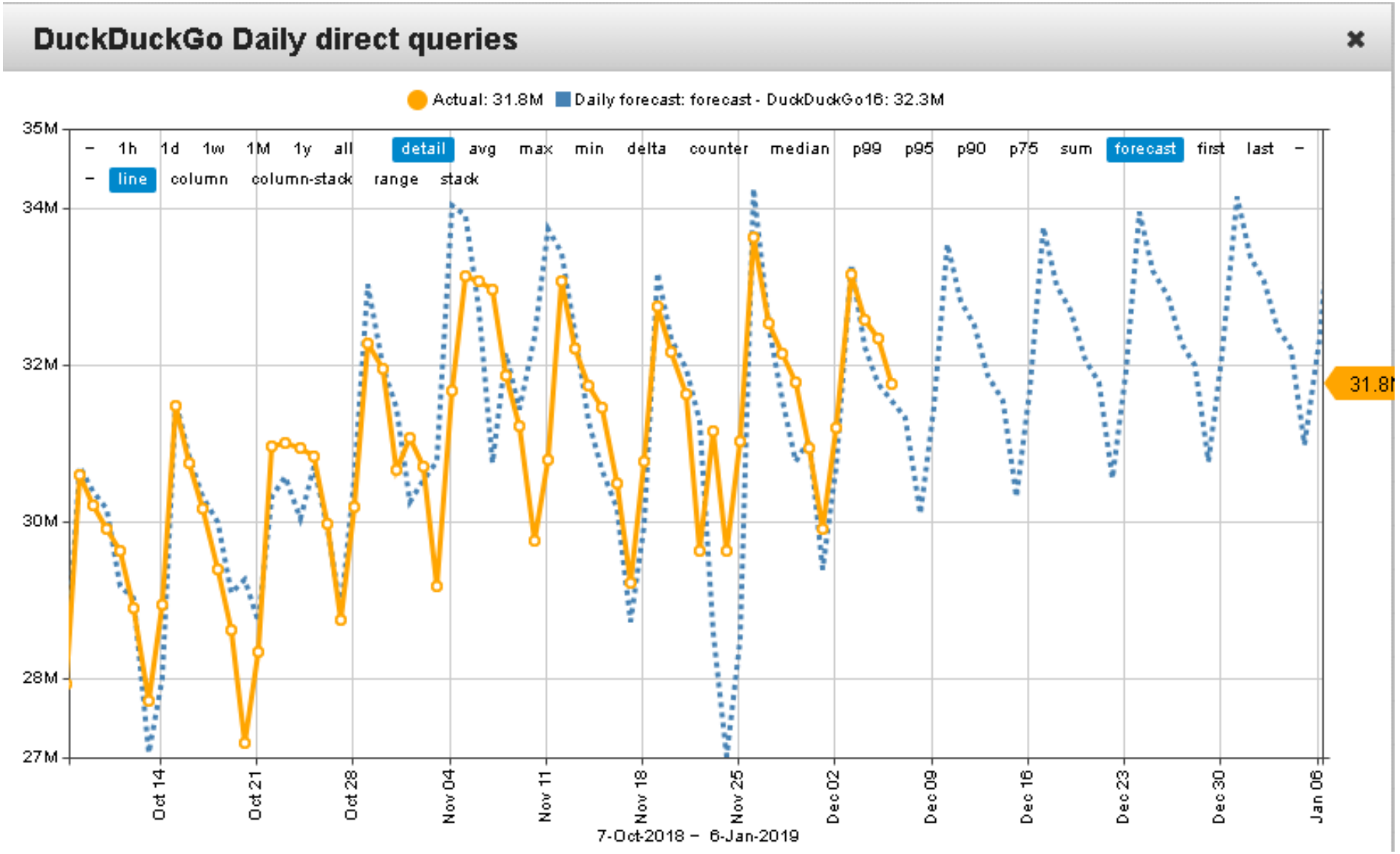
Тройное экспоненциальное сглаживание

А.В. Стриганова, 13643/1
преподаватель: С. Г. Попов
Институт прикладной математики и механики
Санкт-Петербургский политехнический университет им. Петра Великого

Прогнозирование временных рядов



VIEW IN CHARTLAB

Простые методы

y – исходное значение
ŷ – спрогнозированное значение

- Арифметическое среднее

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

- Скользящее среднее

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

- Взвешенное скользящее среднее

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

Экспоненциальное сглаживание

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

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

Двойное экспоненциальное сглаживание

l – *level* – одно спрогнозированное значение

b – *trend* – разница между соседними элементами

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

$$l_i = \alpha * y_i + (1 - \alpha) * (l_{i-1} + b_{i-1}) \quad \textit{level}$$

$$b_i = \beta * (l_i - l_{i-1}) + (1 - \beta) * b_{i-1} \quad \textit{trend}$$

$$\hat{y}_{i+1} = l_i + b_i \quad \textit{forecast}$$

Holt-Winters

L – длина сезона (периода)

S – *seasonal component* – значение, повторяющееся на том же месте в сезонах

$$l_i = \alpha * (y_i - s_{i-L}) + (1 - \alpha) * (l_{i-1} + b_{i-1})$$ *level*

$$b_i = \beta * (l_i - l_{i-1}) + (1 - \beta) * b_{i-1}$$ *trend*

$$s_i = \gamma * (y_i - l_i) + (1 - \gamma) * s_{i-L}$$ *seasonal*

$$\hat{y}_{i+m} = l_i + mb_i + s_{i-L+1+(m-1) mod L}$$ *forecast*

Начальные значения: trend

Среднее средних трендов по сезонам:

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

Начальные значения: seasonal

```
20  def initial_seasonal_components(series, L):
21      """
22      Начальные значения для seasonal components - Sx
23
24      :param series: исходные значения ряда
25      :param L: длина сезона
26      :return: начальные значения для сезонов
27      """
28      seasonals = {}
29      season_averages = []  # средние знячения в сезоне
30      n = int(len(series) / L)  # количество сезонов
31      for j in range(n):
32          season_averages.append(sum(series[L * j:L * j + L]) / float(L))
33      # каждое значение в сезоне делится на среднее по сезону,
34      # затем полученный результат суммируется с соответствующим значением из следующего сезона,
35      # полученная сумма делится на количество сезонов
36      for i in range(L):
37          sum_of_vals_over_avg = 0.0
38          for j in range(n):
39              sum_of_vals_over_avg += series[L * j + i] / season_averages[j]
40          seasonals[i] = sum_of_vals_over_avg / n
41      return seasonals
```

POLYTECH

Реализация: fitting

```
79  # SSE - sum of squared residuals - сумма квадратов разностей
80  def SSE(coefficients, *arr):
81      series = arr[0]
82      season_length = arr[1]
83      n_forecast = arr[2]
84      alpha, beta, gamma = coefficients
85      forecast = holt_winters(series, season_length,
86                              alpha=alpha, beta=beta,
87                              gamma=gamma, n_forecast=n_forecast)
88      sse = 0
89      for i in range(0, len(series)):
90          sse += (forecast[i] - series[i]) ** 2
91      return sse
```

Реализация: минимизация

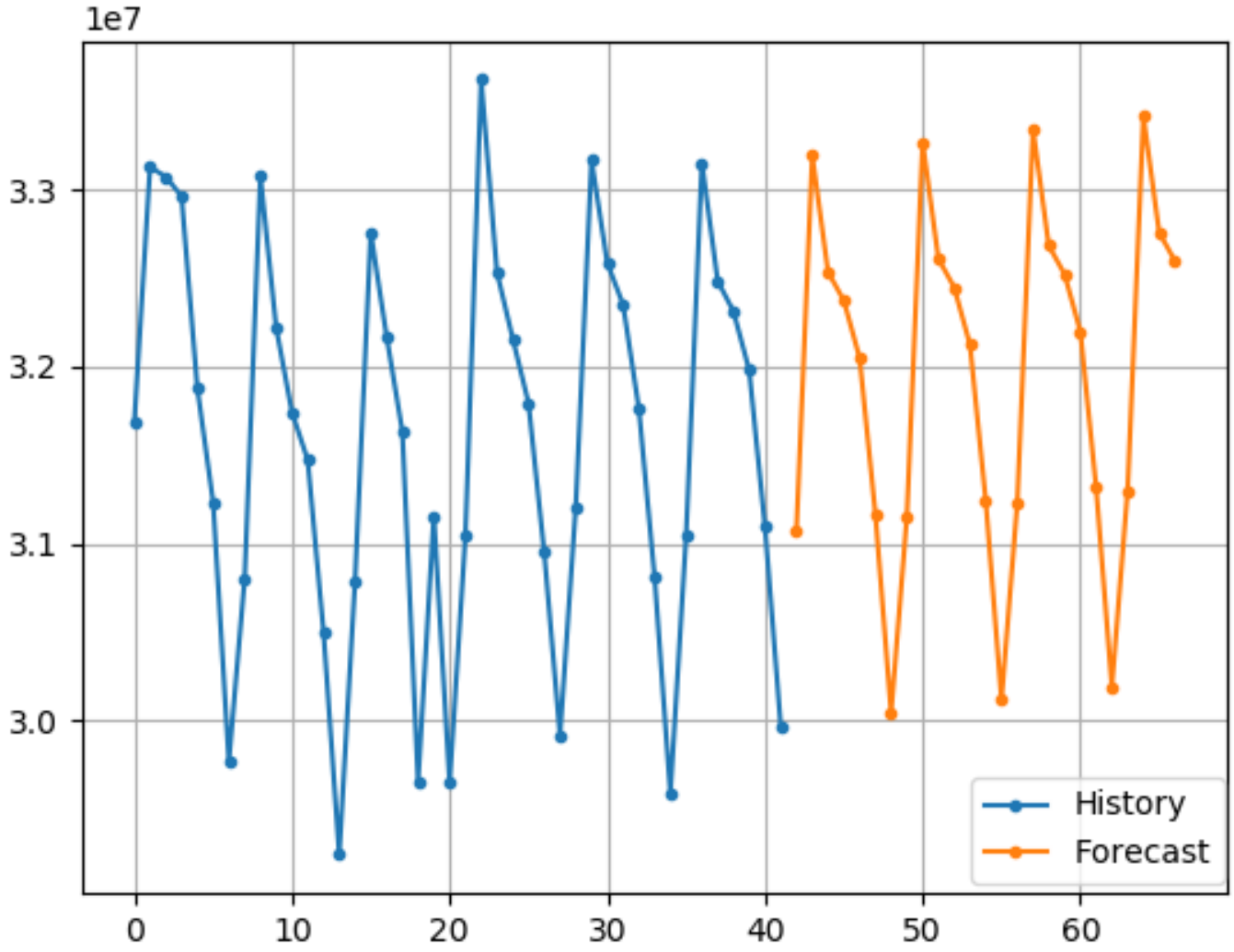
```
94  def forecast(series, season_length, n_forecast):
95      Y = series[:]
96      initial_values = np.array([0.3, 0.1, 0.1])
97      boundaries = [(0, 1), (0, 1), (0, 1)]
98      # минимизируем ошибку с помощью алгоритма L-BFGS-B
99      parameters = fmin_l_bfgs_b(SSE, x0=initial_values,
100                                 args=(Y, season_length, n_forecast),
101                                 bounds=boundaries,
102                                 approx_grad=True)
103     alpha, beta, gamma = parameters[0]
104     print(parameters[0])
105     return holt_winters(series, season_length, alpha, beta, gamma, n_forecast)
```

POLYTECH

Соединение с ATSD

```
from TDB.hw import forecast
connection = connect_url("https://atsd_host:8443", "username", "password")
sf = SeriesFilter(metric="direct.queries")
ef = EntityFilter(entity="duckduckgo")
start_date = "2018-11-03T02:59:00Z"
df = DateFilter(interval={"count": 3, "unit": "MONTH"}, start_date=start_date)
query_data = SeriesQuery(series_filter=sf, entity_filter=ef, date_filter=df)
svc = SeriesService(connection)
series, = svc.query(query_data)
print(series)
plt.plot(series.values(), 'b', forecast(series.values(), 7, 25), 'b--')
plt.grid(True)
plt.show()
```

Результаты



$\alpha = 0.01366847$

$\beta = 0.9942451$

$\gamma = 1.0$

Прогнозы в ATSD

- Holt-Winters
- ARIMA
- SAS

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"
    }
]
```

Как пользоваться

1) git clone https://github.com/annstriganova/HW.git
2) из папки TDB скопировать файл hw.py в директорию с проектом
3) добавить импорт: from hw import forecast
4) forecast(series, m, L)

Спасибо за внимание!