МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени
первого Президента России Б. Н. Ельцина»

ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ РЯДОВ

Методические указания к выполнению
лабораторной работы № 8

Екатеринбург
2019

Содержание

Введение

Целью данной лабораторной работы является изучение студентами методов прогнозирования временных рядов на основе минимизации среднеквадратичной ошибки. В ходе выполнения работы студентами приобретаются навыки и умения по применению методик экстраполяции трендов и прогнозирования на основе авторегрессионных моделей.

1. Задание на лабораторную работу

Результатом выполнения лабораторной работы является оформленный отчет в виде *Jupyter*-тетради, в котором должны быть представлены и отражены все нижеперечисленные пункты:

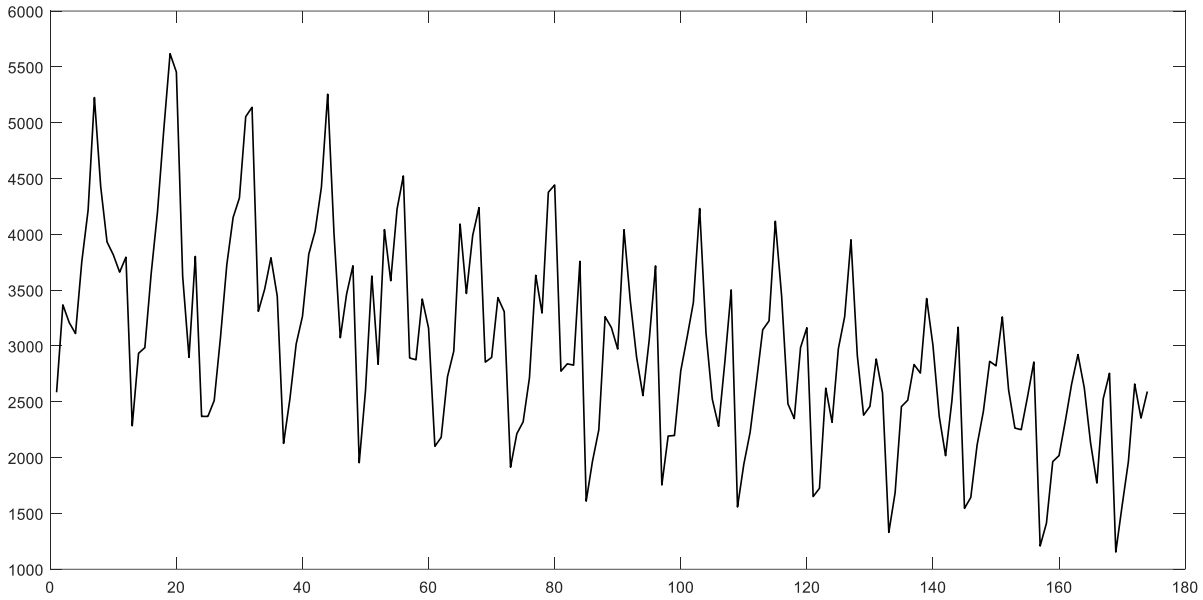
1) Сначала импортируйте в свой код нужные библиотеки, функции и т.д.

```
import numpy as np
import matplotlib.pyplot as plt
import h5py
import scipy.stats as stats
from statsmodels.tsa import api as tsa
from statsmodels.tsa.arima_model import ARIMA
%matplotlib inline
```

2) Загрузите из mat-файла **Fort.mat** ряд, содержащий отсчеты некоторого реального ВР, всего 174 отсчета в вектор-строке.

```
file = h5py.File('Fort.mat', 'r')
data = file.get('Fort')
Fort = np.array(data)
plt.figure(figsize = (10, 5))
plt.plot(Fort, 'k')
plt.show()
```

3) Вид ВР представлен на рисунке ниже. На глаз, в ряде видна явная сезонность, присутствует некоторый убывающий тренд. Был выбран короткий ряд, чтобы повысить скорость расчетов.



4) Мы будем производить **ретроспективный прогноз**, то есть у нас всегда будут точки, по которым можно будет сравнить, правильным получился прогноз, или нет. Для этого отрежем от данного ряда последние 24 точки (которые мы и будем прогнозировать):

```
Z = Fort[:len(Fort)-24+1] # отрезаем последние 24 точки
t=np.arange(0, len(Z), 1) # временная шкала для регрессии
t=t.reshape(-1,1)
plt.figure(figsize = (10, 5))
plt.plot(Fort, 'k') # исходный ВР
plt.plot(t, Z, 'b') # урезанный ряд
plt.show()
```

5) Начнем с простейших методов прогноза. Построим регрессионную модель тренда первого, второго и третьего порядка на основе методики, которая изучалась в лабораторной работе №3, а затем используем полученные регрессионные кривые для прогноза тренда.

6) Сначала для прогноза по методу регрессии используем готовые библиотеки. Например, с помощью **sklearn**:

```
t=np.arange(0, len(Z), 1) # диапазон урезанного ряда
t=t.reshape(-1,1)
t0=np.arange(0, len(Fort), 1) # диапазон полного ряда
t0=t0.reshape(-1,1)
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(t, Z) # модель регрессии
plt.figure(figsize = (10, 5))
plt.plot(t, Z, 'k')
plt.plot(t0, reg.predict(t0), 'r') # прогноз на весь диапазон
plt.plot(t0[-24:], Fort[-24:], 'b') # реальные прогнозные значения
plt.show()
```

7) Напрямую через МНК библиотеки **statsmodels**:

```
import statsmodels.api as sm
x_ = sm.add_constant(t)
smm = sm.OLS(Z, x_)
res = smm.fit() # строим модель регрессии
print(res.params)
plt.figure(figsize = (10, 5))
plt.plot(t, Z, 'k')
plt.plot(t0, res.predict(sm.add_constant(t0)), 'r') # строим прогноз
plt.plot(t0[-24:], Fort[-24:], 'b') # реальные прогнозные значения
plt.show()
```

8) Через полиномиальные кривые **polyfit**:

```
bb = np.polyfit(t.reshape(1,-1)[0], Z.reshape(1,-1)[0], 1)
p = np.poly1d(bb) # создаем экземпляр полинома
plt.figure(figsize = (10, 5))
plt.plot(t, Z, 'k')
plt.plot(t0, p(t0), 'r') # полином на прогнозной временной сетке
plt.plot(t0[-24:], Fort[-24:], 'b')
plt.show()
```

9) Через подгонку функций библиотеки **scipy.optimize**:

```
def func(t, b0, b1):
    return b0 + b1 * t

from scipy.optimize import curve_fit
popt, pcov = curve_fit(func, t.reshape(1,-1)[0], Z.reshape(1,-1)[0])
plt.figure(figsize = (10, 5))
plt.plot(t, Z, 'k')
plt.plot(t0, t0*popt[1]+popt[0], 'r')
plt.plot(t0[-24:], Fort[-24:], 'b')
plt.show()
```

10) Теперь аналогичным образом постройте прогнозы трендов данного ряда для регрессионной кривой **второго и третьего** порядка, строя регрессионные модели тренда подобно тому, как это происходило в лабораторной работе №3.

11) Оцените точность каждого из получившихся прогнозов с помощью следующих оценок (где M – число прогнозируемых точек):

Средняя ошибка прогноза: $\overline{\Delta}^* = \frac{\sum_{i=1}^{M} \Delta_i^*}{M} = \frac{\sum_{i=1}^{M} \left| y(t_i) - y_i \right|}{M}$

СКВО прогноза: $\sigma_e = \sqrt{\frac{\sum_{i=1}^{M} \left(y(t_i) - y_i \right)^2}{M}}$

Средняя ошибка аппроксимации: $\overline{\varepsilon} = \frac{1}{M} \sum_{i=1}^{M} \frac{\left| y(t_i) - y_i \right|}{y(t_i)} \cdot 100\%$

Коэффициент несоответствия 1: $KH_1 = \sqrt{\frac{\sum_{i=1}^{M} \left(y_i - y(t_i) \right)^2}{\sum_{i=1}^{M} y(t_i)^2}}$

Коэффициент несоответствия 2: $KH_2 = \sqrt{\frac{\sum_{i=1}^{M} \left(y_i - y(t_i) \right)^2}{\sum_{i=1}^{M} \left(\overline{y} - y(t_i) \right)^2}}$

12) Постройте доверительные интервалы для тренда **первого** порядка на рисунке вместе с прогнозом и самим рядом по следующей методике:

$$\tau_B(t) = \tau(t) + \delta(t),$$
$$\tau_H(t) = \tau(t) - \delta(t),$$

Нам требуется оценить величину $\delta(t)$. Для **тренда первого порядка** эта величина равняется:

$$\delta_{p=1}(t_l) = 1.96 \cdot S \cdot \sqrt{1 + \frac{1}{N} + \frac{\left(\tau(t_l) - \overline{\tau}\right)^2}{\sum_{i=1}^{N}\left(\tau_i - \overline{\tau}\right)^2}}$$

где $S = \sqrt{\frac{\sum_{i=1}^{N} e_i^2}{N-2}}$, e_i – остаточный ряд или ряд ошибок, то есть разница между исходным ВР и его прогнозом.

13) Постройте доверительные интервалы для тренда **второго и третьего** порядка на рисунке вместе с прогнозом и самим рядом, используя более простые эмпирические оценки:

$$\delta(t) = 1.96 \cdot \sqrt{\frac{\sum_{i=1}^{M}\left(y(t_i) - y_i\right)^2}{M}}$$

14) Метод построения регрессионной кривой через подгонку функций библиотеки **scipy.optimize** позволяет по МНК задавать любую форму кривой. Попробуйте самостоятельно задать некоторую параметрическую кривую, которая давала бы более высокую точность, нежели простые линейные регрессионные кривые. Оцените ее точность аналогично и постройте эмпирические оценки доверительных интервалов.

15) Теперь обратимся к прогнозированию на основе АРПСС моделей. Но прежде, чем строить такую модель, **обратите внимание**: модели АРПСС строятся для рядов с около-нулевым средним, что неверно для заданного временного ряда. Поэтому – **сначала постройте линейный тренд прогнозируемого ряда** (см. линейную регрессию первого порядка выше)**, а затем вычтите его из исходного ряда**, приведя его к нулевому среднему значению (к так называемой **тренд-стационарной** форме).

16) Подберите для данного приведенного к нулю ВР модель АРПСС (*p*, *d*, *q*) некоторого порядка (все параметры целиком и полностью определяются самим студентом) подобно методике из лабораторной работы № 4. Например, была найдена некоторая наилучшая модель:

```
arimaz = ARIMA(Z_minus_trend, order = (p, d, q))
model_fit = arimaz.fit(disp = False) # подгоняем под ВР
print(model_fit.summary())
```

17) Тогда график прогноза по данной модели вместе с доверительными интервалами строится очень легко:

```
model_fit.plot_predict(0, len(Fort))
```

18) Но хотелось бы все же увидеть – как же этот прогноз по АРПСС модели соотносится с исходными известными 24 прогнозными точками (ведь прогноз все-таки ретроспективный). Для этого нужно из исходного ряда ***Fort*** тоже вычесть линейный тренд и соотнести их на одном изображении:

```
plt.figure(figsize = (10, 5))
model_fit.plot_predict(0, len(Fort)) # прогноз по АРПСС
plt.plot(t0, Fort-(trend_as_func_of_t0), 'r') # исходный ВР минус тренд
plt.show()
```

19) Сами прогнозные значения по модели АРПСС можно получить с помощью функции **predict**:

model_fit.predict(len(Z), len(Fort))

20) Используйте эти значения для оценки точности прогноза на основе оценок из пункта 11 выше.

21) Наконец, попробуйте построить АРПСС модель для прогнозирования данного ряда, **но без исходного вычитания из него линейного тренда**. Отметьте получившиеся отличия в работе функций *Python* и точности конечных результатов.

22) Аналогично, для данной модели постройте графики прогноза с доверительными интервалами относительно оригинального ряда ***Fort***, а также оцените точность прогноза на основе оценок из пункта 11 выше.

23) Сформируйте итоговый отчет.

2. Требования к оформлению отчета

Отчет в Jupyter-тетради должен обязательно содержать: номер лабораторной работы, ФИО студента, номер варианта (либо студенческий номер), номер группы, результаты выполнения работы с комментариями студента (комментарии пишутся после #) и изображениями.