Лабораторная работа №8

Рекуррентные нейронные сети для анализа временных рядов

Набор данных для прогнозирования временных рядов, который состоит из среднемесячного числа пятен на солнце, наблюдаемых с января 1749 по август 2017.

Данные в виде csv-файла можно скачать на сайте *Kaggle*: https://www.kaggle.com/robervalt/sunspots/ (https://www.kaggle.com/robervalt/sunspots/)

Задание 1

Загрузите данные. Изобразите ряд в виде графика. Вычислите основные характеристики временного ряда (сезонность, тренд, автокорреляцию).

In [0]:

```
import warnings

warnings.filterwarnings('ignore')
```

In [0]:

```
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

In [0]:

```
from google.colab import drive

drive.mount('/content/drive', force_remount = True)
```

```
Mounted at /content/drive
```

In [0]:

```
BASE_DIR = '/content/drive/My Drive/Colab Files/mo-2'

import sys

sys.path.append(BASE_DIR)

import os
```

In [0]:

```
DATA_ARCHIVE_NAME = 'sunspots.zip'

LOCAL_DIR_NAME = 'sunspots'
```

In [0]:

```
from zipfile import ZipFile

with ZipFile(os.path.join(BASE_DIR, DATA_ARCHIVE_NAME), 'r') as zip_:
    zip_.extractall(LOCAL_DIR_NAME)
```

In [0]:

```
DATA_FILE_PATH = 'sunspots/Sunspots.csv'
```

In [0]:

```
import pandas as pd

all_df = pd.read_csv(DATA_FILE_PATH, parse_dates = ['Date'], index_col = 'Date')
```

In [0]:

```
print(all_df.shape)
```

```
(3252, 2)
```

In [0]:

```
all_df.keys()
```

Out[10]:

```
Index(['Unnamed: 0', 'Monthly Mean Total Sunspot Number'], dtype='object')
```

In [0]:

```
from statsmodels.tsa.seasonal import seasonal_decompose

additive = seasonal_decompose(all_df['Monthly Mean Total Sunspot Number'],
                              model = 'additive', extrapolate_trend = 'freq')
```

In [0]:

```
%matplotlib inline

import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

rcParams['figure.figsize'] = 12, 8

sns.set()
sns.set_palette(sns.color_palette('hls'))

def plot_accuracy(_history,
                  _train_acc_name = 'accuracy',
                  _val_acc_name = 'val_accuracy'):

    plt.plot(_history.history[_train_acc_name])
    plt.plot(_history.history[_val_acc_name])

    plt.title('Model accuracy')

    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')

    plt.legend(['Train', 'Validation'], loc = 'right')

    plt.show()

def plot_loss(_history):

    plt.plot(_history.history['loss'])
    plt.plot(_history.history['val_loss'])

    plt.title('Model loss')

    plt.ylabel('Loss')
    plt.xlabel('Epoch')

    plt.legend(['Train', 'Validation'], loc = 'right')

    plt.show()
```

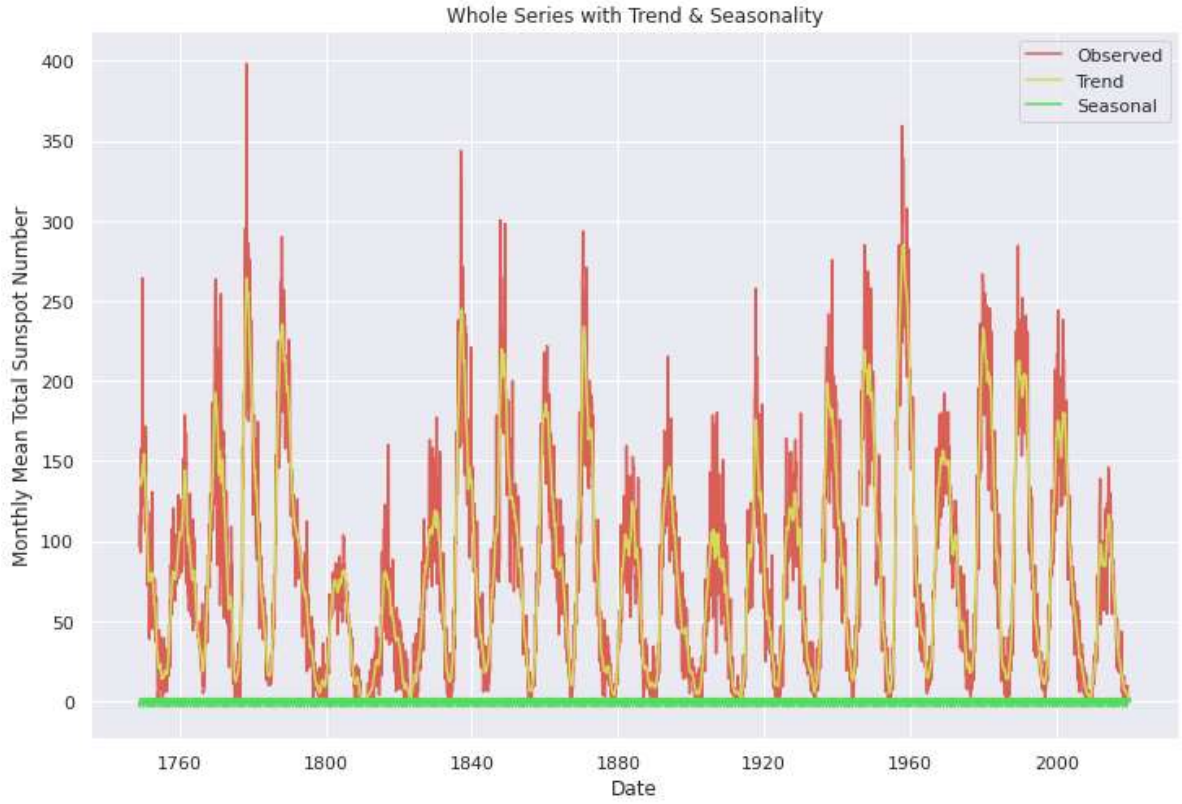
In [0]:

```
sns.lineplot(data = additive.observed, label = 'Observed')
sns.lineplot(data = additive.trend, label = 'Trend')
sns.lineplot(data = additive.seasonal, label = 'Seasonal')

plt.xlabel('Date')
plt.ylabel('Monthly Mean Total Sunspot Number')

plt.title('Whole Series with Trend & Seasonality')

plt.show()
```



Рассмотрим подробнее на небольшом промежутке:

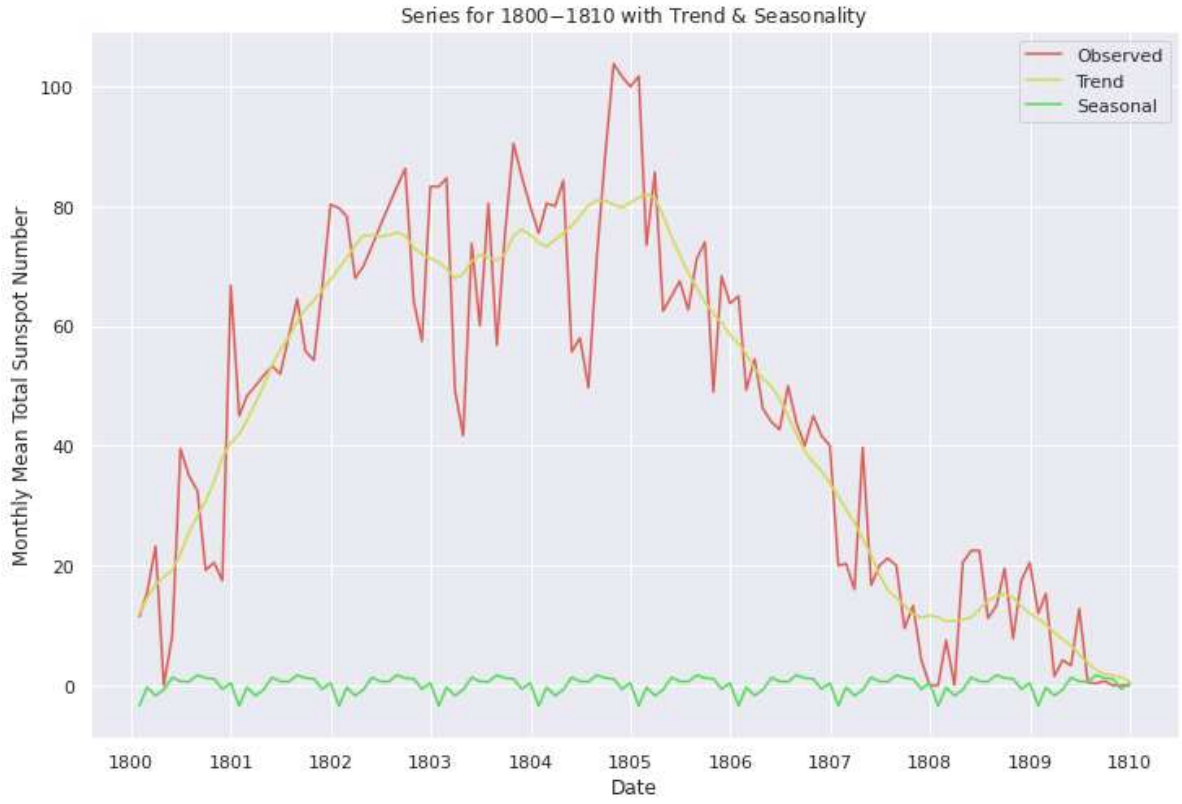
In [0]:

```
sns.lineplot(data = additive.observed['1800-01-01':'1810-01-01'], label = 'Observed')
sns.lineplot(data = additive.trend['1800-01-01':'1810-01-01'], label = 'Trend')
sns.lineplot(data = additive.seasonal['1800-01-01':'1810-01-01'], label = 'Seasonal')

plt.xlabel('Date')
plt.ylabel('Monthly Mean Total Sunspot Number')

plt.title('Series for 1800$-$1810 with Trend & Seasonality')

plt.show()
```



In [0]:

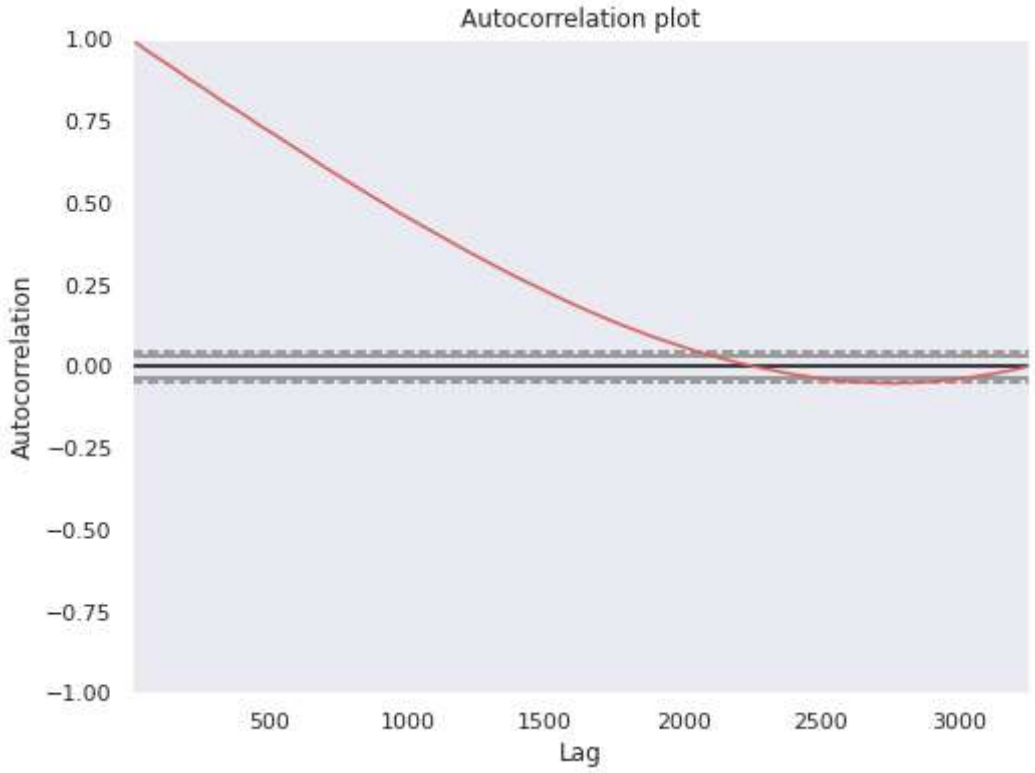
```
from pandas.plotting import autocorrelation_plot

rcParams['figure.figsize'] = 8, 6

autocorrelation_plot(all_df.values.tolist())

plt.title('Autocorrelation plot')

plt.show()
```



Задание 2

Для прогнозирования разделите временной ряд на обучающую, валидационную и контрольную выборки.

Этот шаг будет применён автоматически с помощью индексации массива данных и как параметр `validation_split` метода `model.fit()`.

Задание 3

Примените модель *ARIMA* для прогнозирования значений данного временного ряда.

In [0]:

```
! pip install pmdarima --quiet
```

In [0]:

```
TEST_PERIOD = 600
```

In [0]:

```
OBSERVATIONS_PER_CYCLE = 11 * 12
```

In [0]:

```
from pmdarima.arima import auto_arima

arima = auto_arima(all_df['Monthly Mean Total Sunspot Number'][:-TEST_PERIOD],
                   trace = True, error_action = 'ignore',
                   suppress_warnings = True, seasonal = True,
                   max_p = 1, max_q = 2,
                   m = OBSERVATIONS_PER_CYCLE)
```

```
Performing stepwise search to minimize aic
Fit ARIMA: (1, 0, 2)x(1, 0, 1, 132) (constant=True); AIC=24713.600, BIC=2475
4.781, Time=1202.505 seconds
Fit ARIMA: (0, 0, 0)x(0, 0, 0, 132) (constant=True); AIC=29864.324, BIC=2987
6.090, Time=0.098 seconds
Fit ARIMA: (1, 0, 0)x(1, 0, 0, 132) (constant=True); AIC=25075.627, BIC=2509
9.159, Time=282.392 seconds
Fit ARIMA: (0, 0, 1)x(0, 0, 1, 132) (constant=True); AIC=27390.911, BIC=2741
4.444, Time=427.550 seconds
Near non-invertible roots for order (0, 0, 1)(0, 0, 1, 132); setting score t
o inf (at least one inverse root too close to the border of the unit circle:
0.992)
Fit ARIMA: (0, 0, 0)x(0, 0, 0, 132) (constant=False); AIC=32247.275, BIC=322
53.158, Time=0.058 seconds
Fit ARIMA: (1, 0, 2)x(0, 0, 1, 132) (constant=True); AIC=24716.156, BIC=2475
1.454, Time=598.015 seconds
Fit ARIMA: (1, 0, 2)x(1, 0, 0, 132) (constant=True); AIC=24715.701, BIC=2475
1.000, Time=640.621 seconds
```

In [0]:

```
arima_forecast = arima.predict(n_periods = TEST_PERIOD)
```

In [0]:

```
from sklearn.metrics import mean_squared_error

mean_squared_error(all_df['Monthly Mean Total Sunspot Number'][-TEST_PERIOD:],
                   arima_forecast)
```

Задание 4

Повторите эксперимент по прогнозированию, реализовав рекуррентную нейронную сеть (с как минимум 2 рекуррентными слоями).

Сначала нужно создать датасет из данных.

In [0]:

```
! pip install tensorflow-gpu --pre --quiet
```

In [0]:

```
TIME_STEPS = OBSERVATIONS_PER_CYCLE
```

In [0]:

```
import numpy as np
from datetime import timezone

def timeseries_to_dataset(_X_ts, _time_steps):

    samples_n_ = len(_X_ts) - _time_steps

    print( len(_X_ts))

    X_ = np.zeros((samples_n_, _time_steps))
    y_ = np.zeros((samples_n_, ))

    for i in range(samples_n_):

        X_[i] = _X_ts[i:(i + _time_steps)]

        y_[i] = _X_ts[(i + _time_steps)]

    return X_[..., np.newaxis], y_
```

In [0]:

```
X, y = timeseries_to_dataset(
    all_df['Monthly Mean Total Sunspot Number'][:-TEST_PERIOD].values,
    TIME_STEPS)

X_test, y_test = timeseries_to_dataset(
    all_df['Monthly Mean Total Sunspot Number'][-TEST_PERIOD:].values,
    TIME_STEPS)
```

In [0]:

```
import tensorflow as tf
from tensorflow import keras
```

In [0]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = tf.keras.Sequential()

model.add(LSTM(8, activation = 'relu', return_sequences = True,
               input_shape = X.shape[-2:]))
model.add(LSTM(8, activation = 'relu'))
model.add(Dense(1))
```

In [0]:

```
model.compile(optimizer = 'adam',
              loss = 'mse',
              metrics = ['accuracy'])

model.summary()
```

In [0]:

```
history = model.fit(x = X, y = y, epochs = 20, validation_split = 0.15,
                    verbose = 0)
```

In [0]:

```
plot_accuracy(history)
```

In [0]:

```
plot_loss(history)
```

In [0]:

```
results = model.evaluate(X_test, y_test)

print('Test mse, test accuracy:', results)
```

Задание 5

Сравните качество прогноза моделей.

Какой максимальный результат удалось получить на контрольной выборке?

Нейронная сеть дала среднеквадратичную ошибку в 4 раза больше, чем *ARIMA*, а точность предсказания вообще равна нулю. Можно сделать вывод, что предсказание временных рядов требует более тонкой настройки архитектуры сетей.