ThinkDSP. Лабораторная 5. Автокорреляция.

Шерепа Никита

8 мая 2021 г.

Содержание

1 Упражнение 5.1 **5**

2 Упражнение 5.2 **8**

3 Упражнение 5.3 **11**

4 Упражнение 5.4 **14**

5 Вывод **19**

Список иллюстраций

1 Высота тона звука 6
2 Высота тона звука 7
3 Спектрограмма звука 8
4 Кривая и спектрограмма 10
5 Таблица цен *BitCoin* 11
6 График цен *BitCoin* за 7 лет 12
7 Результат автокорреляции 13
8 Спектр сигнала с 2 по 2.5 секунду 15
9 Результат автокорреляции 16
10 Удалили частоту 464 Гц 17
11 Убрали пики 18

Листинги

1 Метод serial_corr 5
2 Метод autocorr 5
3 Фрагмент 1 и высота его тона 5
4 Уточнение частоты 6
5 Фрагмент 2 и высота его тона 6
6 Уточнение частоты 7
7 Спектрограмма звука 8
8 Метод *estimate_fundamental* 9
9 Вычисление частоты с помощью метода *estimate_fundamental* 9
10 Вычисление кривой отслеживания тона 9
11 Таблица цен *BitCoin* 11
12 График цен *BitCoin* 11
13 Автокоррелируем 12
14 Работа со звуком 14
15 Автокоррелируем 15
16 Функция *find_frequency* 16
17 Частота первого пика 16
18 Удалим частоту 464 Гц 16
19 Убираем пики 17

1 Упражнение 5.1

1. **Задание**

   Вычислите автокорреляцию для различных *lag*, воспользовавшись материалами из *chap05.ipynb*. Оцените высоты тона вокального чирпа для нескольких времен начала сегмента.

2. **Ход работы**

   Воспользуемся некоторыми методами из *chap05.ipynb*

   serial_ corr - вычисляет корреляцию для разных типов шума

```
def serial_corr(wave, lag=1):
N = len(wave)
y1 = wave.ys[lag:]
y2 = wave.ys[:N-lag]
corr = np.corrcoef(y1, y2)[0, 1]
return corr
```

Листинг 1: Метод serial_corr

   autocorr - вызывает *serial_ corr* с различными значениями *lag*

```
def autocorr(wave):
lags = np.arange(len(wave.ys)//2)
corrs = [serial_corr(wave, lag) for lag in lags]
return lags, corrs
```

Листинг 2: Метод autocorr

   Теперь возьмем звук, выберем из него фрагмент и найдем высоту его тона.

```
wave =
    read_wave('res/28042__bcjordan__voicedownbew.wav')
wave.normalize()
segment = wave.segment(start=0.3, duration=0.01)

lags, corrs = autocorr(segment)
plt.plot(lags, corrs)
decorate(xlabel='Lag (index)', ylabel='Correlation')
```

Листинг 3: Фрагмент 1 и высота его тона

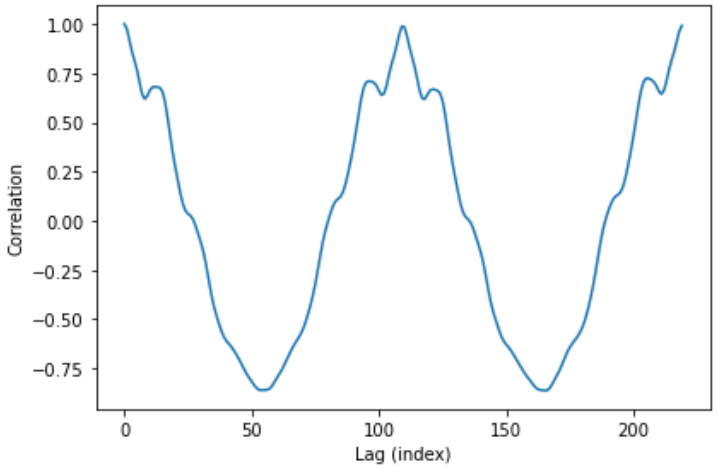

Рис. 1: Высота тона звука

Видим, что пик лежит между 100 и 120. Для уточнения *lag* воспользуемся методом *argmax*

```
low, high = 100, 120
lag = np.array(corrs[low:high]).argmax() + low

period = lag / segment.framerate
frequency = 1 / period
frequency

Output
404.5871559633028
```

Листинг 4: Уточнение частоты

Частота = 404.5871559633028

Теперь возьмем другой фрагмент и вычислим частоту для него

```
wave =
    read_wave('res/28042__bcjordan__voicedownbew.wav')
wave.normalize()
segment = wave.segment(start=0.6, duration=0.01)

lags, corrs = autocorr(segment)
plt.plot(lags, corrs)
decorate(xlabel='Lag (index)', ylabel='Correlation')
```

Листинг 5: Фрагмент 2 и высота его тона

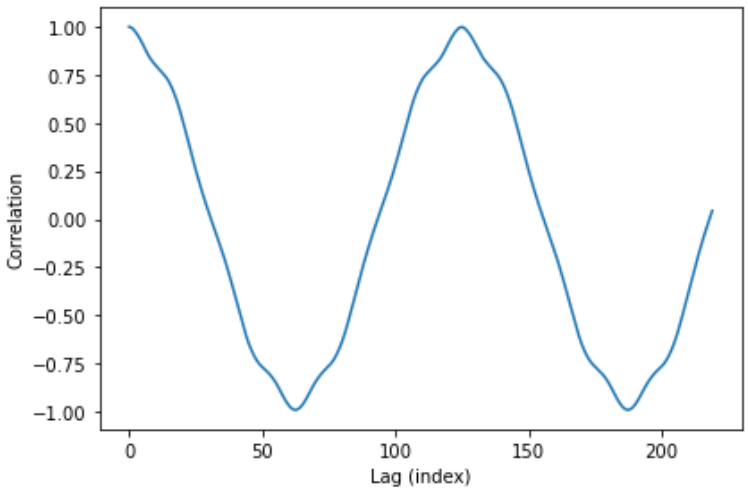

Рис. 2: Высота тона звука

Видим, что пик лежит между 100 и 150. Для уточнения *lag* воспользуемся методом *argmax*

```
low, high = 90, 110
lag = np.array(corrs[low:high]).argmax() + low

period = lag / segment.framerate
frequency = 1 / period
frequency


Output
352.8
```

Листинг 6: Уточнение частоты

Частота = 352.8

Частота уменьшается с увеличением времени начала сегмента.

2 Упражнение 5.2

1. **Задание**

   Инкапсулируйте код, выполняющий автокорееляцию в функцию, названную *estimate_fundamental*, и используйте её для отслеживания высоты тона записанного звука.

   Проверьте, насколько хороша она работает, накладывая оценки высоты тона на спектрограмму записи.

2. **Ход работы**

   Возьмём звук из прошлого пункта и построим его спектрограмму

```
from thinkdsp import read_wave

wave =
    read_wave('res/28042__bcjordan__voicedownbew.wav')
wave.normalize()

wave.make_spectrogram(2048).plot(high=4200)
decorate(xlabel='Time (s)',
ylabel='Frequency (Hz)')
```

Листинг 7: Спектрограмма звука

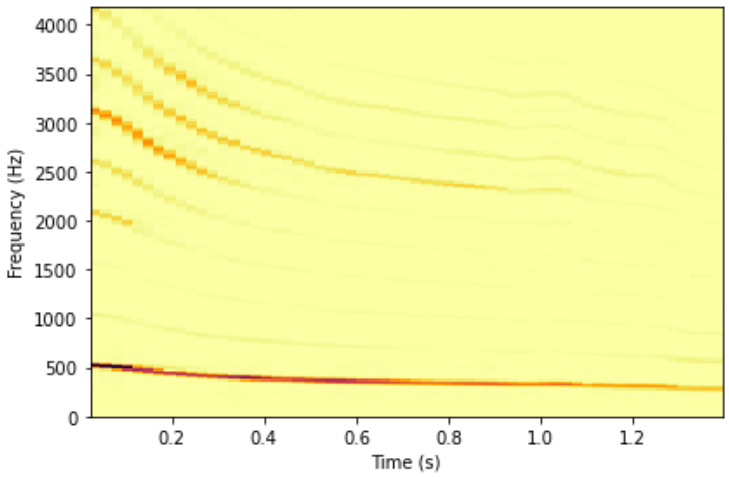

Рис. 3: Спектрограмма звука

Теперь инкапсулируем код в функцию *estimate_fundamental*

```
def estimate_fundamental(segment, low = 70, high =
    150):
lags, corrs = autocorr(segment)
lags = np.array(corrs[low:high]).argmax() + low
period = lag / segment.framerate
frequency = 1 / period
return frequency
```

Листинг 8: Метод *estimate_fundamental*

Проверим его работоспособность

```
segment = wave.segment(start = 0.2, duration = 0.01)
frequency = estimate_fundamental(segment)
frequency

Output
436.63366336633663
```

Листинг 9: Вычисление частоты с помощью метода *estimate_fundamental*

Частота = 436.63366336633663

Теперь вычислим высоту тона записанного звука и построим кривую отслеживания тона, наложенную на спектрограмму

```
step = 0.05
starts = np.arange(0.0, 1.4, step)
duration = 0.01

ts = []
freqs = []

for start in starts:
ts.append(start + step/2)
segment = wave.segment(start, duration)
freq = estimate_fundamental(segment)
freqs.append(freq)

wave.make_spectrogram(2048).plot(high=900)
plt.plot(ts, freqs, color='green')
decorate(xlabel='Time (s)',
ylabel='Frequency (Hz)',
xlim=[0, 1.4],
```

```
19          ylim=[0, 900])
```

Листинг 10: Вычисление кривой отслеживания тона

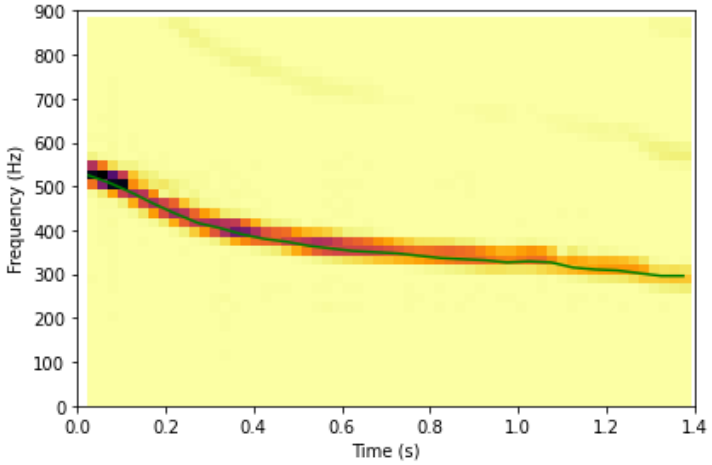

Рис. 4: Кривая и спектрограмма

Видим, что зеленая кривая правильно наложилась на спектрограмму, значит все вычисления верны.

3 Упражнение 5.3

1. **Задание**

   Используя данные про *BitCoin* из прошлой лабораторной работы, вычислите автокорреляции цен в платежной системе *BitCoin*. Быстро ли спадает автокорреляционная функция? Есть ли признаки периодичности процесса?

2. **Ход работы**

   Вспомним данные про *BitCoin* из прошлой лабораторной работы

```
import pandas as pd

df =
    pd.read_csv('res/BTC_USD_2013-10-01_2021-05-05-CoinDesk.csv',
parse_dates=[0])
df
```

Листинг 11: Таблица цен *BitCoin*

	Currency	Date	Closing Price (USD)	24h Open (USD)	24h High (USD)	24h Low (USD)
0	BTC	2013-10-01	123.654990	124.304660	124.751660	122.563490
1	BTC	2013-10-02	125.455000	123.654990	125.758500	123.633830
2	BTC	2013-10-03	108.584830	125.455000	125.665660	83.328330
3	BTC	2013-10-04	118.674660	108.584830	118.675000	107.058160
4	BTC	2013-10-05	121.338660	118.674660	121.936330	118.005660
...	...	...	...	...	...	...
2768	BTC	2021-05-01	57302.646424	53598.879503	57434.933127	53097.762794
2769	BTC	2021-05-02	57677.975222	57741.020910	58511.256049	57062.700071
2770	BTC	2021-05-03	56427.043125	57824.300187	57925.741567	56123.039508
2771	BTC	2021-05-04	57255.306838	56639.439786	59001.359642	56508.240449
2772	BTC	2021-05-05	53658.843121	57218.805329	57246.891191	53613.595218

2773 rows × 6 columns

Рис. 5: Таблица цен *BitCoin*

```
ys = df['Closing Price (USD)']
ts = df.index

from thinkdsp import Wave

wave = Wave(ys, ts, framerate=1)
wave.plot()
```

```
8         decorate(xlabel='Time (days)')
```

Листинг 12: График цен *BitCoin*

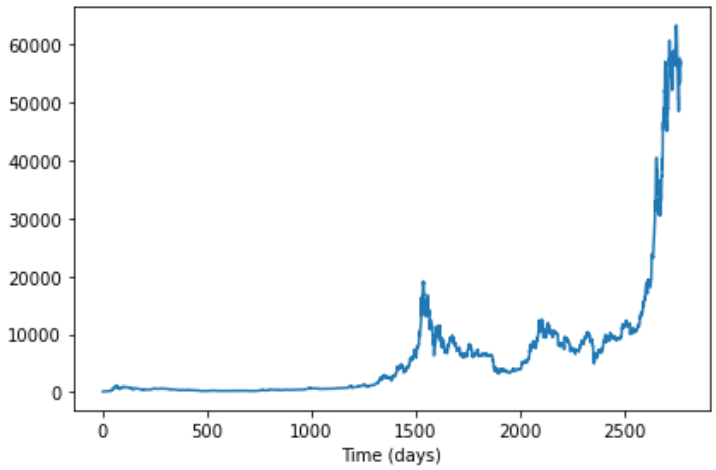

Рис. 6: График цен *BitCoin* за 7 лет

Теперь воспользуемся функцией автокорреляции

```
1         lags, corrs = autocorr(wave)
2         plt.plot(lags, corrs)
3         decorate(xlabel='Lag',
4         ylabel='Correlation')
```

Листинг 13: Автокоррелируем

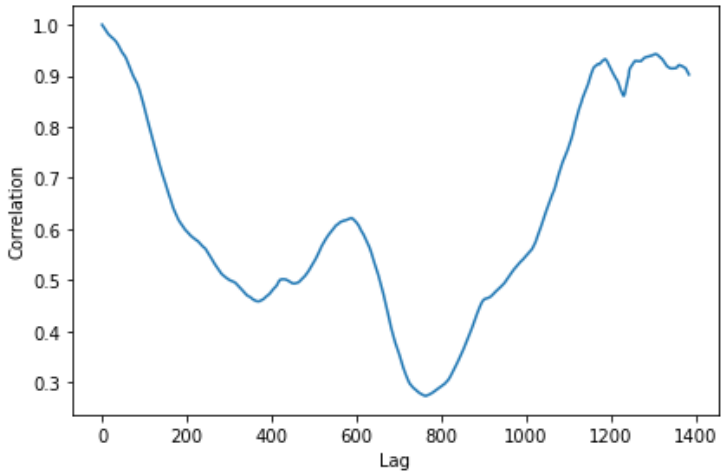

Рис. 7: Результат автокорреляции

Как видим, идет плавное уменьшение значения, как у розового шума. Результат совпадает с результатом прошлом лабораторной.

Увеличение значения происходит из-за того, что на самом графике курс цен сначала увеличился, а затем уменьшился, а затем опять резко увеличился.

4 Упражнение 5.4

1. **Задание**

   Поэкспериментируйте с исследованием автокорреляции, используя материалы файла *saxophone.ipynb*

2. **Ход работы**

   Вопроизведем звук саксофона и посмотрим на спектр сигнала вблизи

```
from thinkdsp import read_wave

wave =
    read_wave('res/100475__iluppai__saxophone-weep.wav')
wave.normalize()
wave.make_audio()

start = 2.0
duration = 0.5
segment = wave.segment(start=start, duration=duration)
segment.make_audio()

spectrum = segment.make_spectrum()
spectrum.plot(high=3000)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 14: Работа со звуком

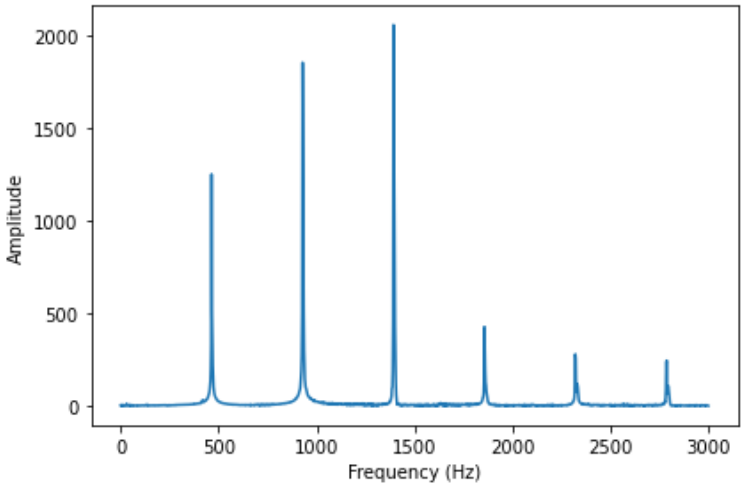

Рис. 8: Спектр сигнала с 2 по 2.5 секунду

Воспринимаем мы основную частоту, равную 464 Гц. Но она не доминирующая. Доминирующая = 1392 Гц.

Воспользуемся автокорреляцией, чтобы понять, почему мы воспринимаем не доминирующую частоту.

```
1          def autocorr(segment):
2             corrs = np.correlate(segment.ys, segment.ys,
                  mode='same')
3             N = len(corrs)
4             lengths = range(N, N//2, -1)
5
6             half = corrs[N//2:].copy()
7             half /= lengths
8             half /= half[0]
9             return half
10
11         corrs = autocorr(segment)
12         plt.plot(corrs[:200])
13         decorate(xlabel='Lag', ylabel='Correlation',
              ylim=[-1.05, 1.05])
```

Листинг 15: Автокоррелируем

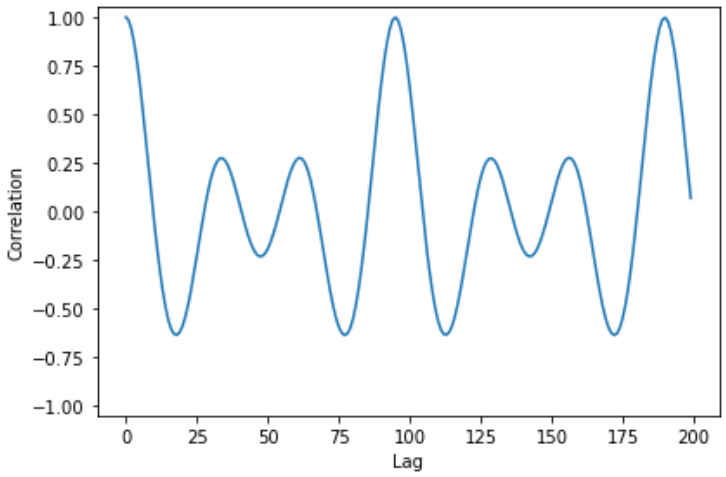

Рис. 9: Результат автокорреляции

Как видим, первый крупный пик находится в районе lag = 100

Воспользуемся фнукцией, которая находит самую высокую корреляцию в заданном диапазоне задержек и возвращает соответствующую частоту - *find_frequency*

```
def find_frequency(corrs, low, high):
lag = np.array(corrs[low:high]).argmax() + low
print(lag)
period = lag / segment.framerate
frequency = 1 / period
return frequency
```

Листинг 16: Функция *find_frequency*

Вычислим частоту 1ого пика

```
find_frequency(corrs, 80, 100)

Output
464.2105263157895
```

Листинг 17: Частота первого пика

Частота = 464.2105263157895

Если попробовать удалить частоту 464 Гц из спектра то восприятие не изменится

```
spectrum2 = segment.make_spectrum()
```

```
spectrum2.high_pass(600)
spectrum2.plot(high=3000)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 18: Удалим частоту 464 Гц

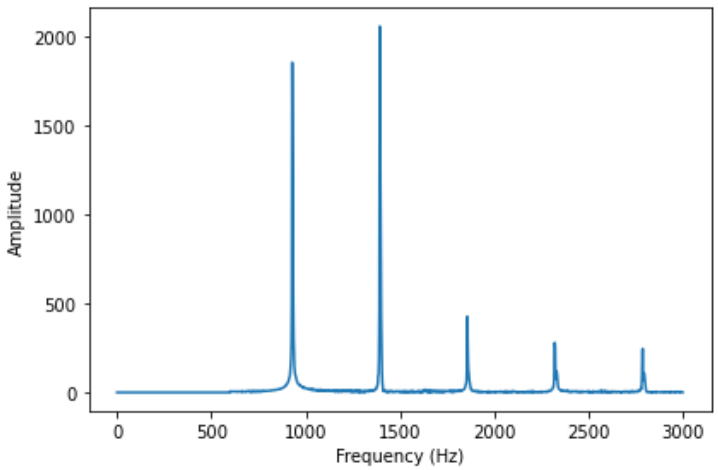

Рис. 10: Удалили частоту 464 Гц

Всопринимается всё еще частота = 464 Гц. Это явление называется *"подавленная"основная частота.*

Мы всё еще воспринимаем частоту = 464 Гц, потому что оставшиеся пики, которые присутствуют в сигнале, представляют собой гармоники 464 Гц.

Наше ухо интерпретирует высокие гармоники как свидетельство того, что «правильная» основная частота = 464 Гц.

Если избавиться от остальных пиков, то весь эффект пропадет и мы будем слышать не саксофон, а звук, похожий на сигнал перед записью сообщения на автоответчик.

```
spectrum4 = segment.make_spectrum()
spectrum4.high_pass(600)
spectrum4.low_pass(1200)
spectrum4.plot(high=3000)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 19: Убираем пики

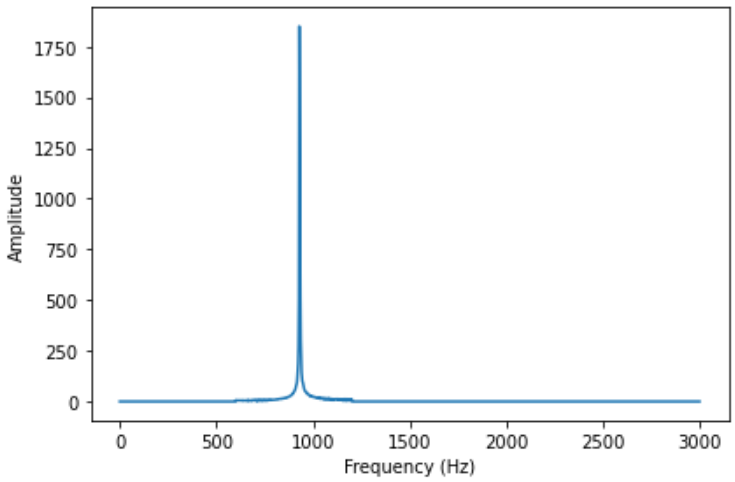

Рис. 11: Убрали пики

5 Вывод

В результате выполнения лабораторной работы получены навыки работы с корреляцией, автокорреляцией. Мы ознакомились с явлением "подавленных"основных частот и функцией автокорреляции.