Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Телекоммуникационные технологии

Отчет по лабораторной работе №2
Корреляция

Работу выполнил:
Чугунов А.А.
Группа: 33501/4
Преподаватель:
Богач Н.В.

Санкт-Петербург
2017

1. Цель работы

Познакомиться с понятием корреляции и функцией корреляции.

2. Теоретическая информация

В данной лабораторной работе будем рассматривать корреляционный анализ. Его смысл состоит в количественном измерении степени сходства различных сигналов. Для этого будем использовать специальные корреляционные функции. Так, для получения взаимной корреляции двух последовательностей, имеем следующую формулу:

$$r_{12}(j) = \frac{1}{N}\sum_{n=0}^{N-1} x_1(n)x_2(n+j) = r_{12}(-j) = \frac{1}{N}\sum_{n=0}^{N-1} x_2(n)x_1(n-j) \tag{1}$$

Здесь j - это смещение одного сигнала относительно другого.

Так же можно ввести аналогичную формулу для непрерывной временной области:

$$r_{12}(\tau) = \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} x_1(t)x_2(t+\tau)dt \tag{2}$$

Рассчет корреляции можно ускорить, воспользовавшись следующей формулой:

$$r_{12}(j) = \frac{1}{N}F_D^{-1}[X_1^*(k)X_2(k)] \tag{3}$$

Здесь, F_D^{-1} - обратное преобразование Фурье. При различной длине сигналов выполнить расчет корреляции можно путем добавления к двум последовательностям дополняющих нулей. Если последовательность $x_1(n)$ имеет длину N1, а последовательность $x_2(n)$ — N2, то $x_1(n)$ дополняется (N2 − 1) нулями, а $x_2(n)$ — (N1 − 1) нулями. Далее на основе двух расширенных последовательностей рассчитывается взаимная корреляция.

3. Ход выполнения работы

Имеем сигнал, сосотоящий из нолей и единиц - [0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0] и синхропосылку - [1, 0, 1]. Задача - найти положение синхропосылки в сигнале. Изначально преобразуем все ноли в -1. Это необходимо для корректной работы алгоритма быстрого расчета корреляции сигналов, так как сигнал будет дополняться нолями. Корреляцию рассчитаем с помощью встроенной функции correlate библиотеки numpy. Алгоритм для быстрого расчета корреляции напишем самостоятельно используя преобразования Фурье из той же библотеки. Результаты представлены на Рисунке. 3.1

Листинг 1: CorrelLab.py

```
from scipy.fftpack import fft
import numpy as np
import time as time
import random
import matplotlib.pyplot as plt
from scipy import signal

sig = np.asarray([0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0])
syncSig = np.asarray([1, 0, 1])

```

```
sig = sig + sig - 1
syncSig = syncSig + syncSig - 1
print('correlation')
print('the signal: ', sig)
print('the sync signal: ', syncSig)

correlation = np.asarray([])
t = np.arange(1000)
sum = 0
for ti in t:
    beforeTime = time.time()
    correlation = np.correlate(sig, syncSig)
    afterTime = time.time()
    sum = sum + afterTime - beforeTime
print('time needed for correlation: ', sum / len(t))

print('------------------------------------------------------------------------------------')
print('fast correlation')
sig = np.asarray([-1, -1, -1, 1, -1, 1, -1, 1, 1, 1, -1, -1, -1, -1, 1, -1, 0, 0])
syncSig = np.asarray([1, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
print('the signal: ', sig)
print('the sync signal: ', syncSig)
sum = 0
finalCorrel = np.asarray([])
for di in t:
    beforeTime = time.time()
    conjY = np.conjugate(np.fft.fft(syncSig))
    y1Fft = np.fft.fft(sig)
    multiplication = conjY * y1Fft
    finalCorrel = np.fft.ifft(multiplication)
    afterTime = time.time()
    sum = sum + afterTime - beforeTime
print('time needed for fast correlation: ', sum / len(t))
print('correlation result: ', correlation)
print('fast correlation result: ', finalCorrel[:14].real)

print('------------------------------------------------------------------------------------')
print('------------------------------------------------------------------------------------')
print('extended signal')
print('')
print('correlation')
s = []
size = np.arange(10000)
for si in size:
    s.append(random.randint(0, 1))
sig = np.asarray(s)
syncSig = np.asarray([1, 0, 1])

sig = sig + sig - 1
syncSig = syncSig + syncSig - 1

print('the signal: ', sig)
print('the sync signal: ', syncSig)
```

```python

correlation = np.asarray([])
sum = 0
for ti in t:
    beforeTime = time.time()
    correlation = np.correlate(sig, syncSig)
    afterTime = time.time()
    sum = sum + afterTime - beforeTime
print('time needed for correlation in an extended signal: ', sum / len(t))


print('--------------------------------------------------------------------------')
print('fast correlation')
sizeZeroListForSig = np.arange(len(syncSig) - 1)
sizeZeroListForSyncSig = np.arange(len(sig) - 1)
sig = list(sig)
for z in sizeZeroListForSig:
    sig.append(0)
syncSig = list(syncSig)
for zi in sizeZeroListForSyncSig:
    syncSig.append(0)
sig = np.asarray(sig)
syncSig = np.asarray(syncSig)

print('the signal: ', sig)
print('the sync signal: ', syncSig)

sum = 0
finalCorrel = np.asarray([])
for di in t:
    beforeTime = time.time()
    conjY = np.conjugate(np.fft.fft(syncSig))
    y1Fft = np.fft.fft(sig)
    multiplication = conjY * y1Fft
    finalCorrel = np.fft.ifft(multiplication)
    afterTime = time.time()
    sum = sum + afterTime - beforeTime
print('time needed for fast correlation in an extended signal: ', sum / len(t))
print('correlation in an extended signal result: ', correlation)
print(len(sig) - len(syncSig))
print('fast correlation in an extended signal result: ', finalCorrel[:9997 ].real)
```

```
correlation
the signal:  [-1 -1 -1  1 -1  1 -1  1  1  1 -1 -1 -1 -1  1 -1]
the sync signal:  [ 1 -1  1]
time needed for correlation:  1.3542296886444091e-06
--------------------------------------------------------------------------
fast correlation
the signal:  [-1 -1 -1  1 -1  1 -1  1  1  1 -1 -1 -1 -1  1 -1  0  0]
the sync signal:  [ 1 -1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
time needed for fast correlation:  1.6743515253067018e-05
correlation result:  [-1  1 -3  3 -3  3 -1  1 -1  1 -1 -1  1 -3]
fast correlation result:  [-1.  1. -3.  3. -3.  3. -1.  1. -1.  1. -1. -1.  1. -3.]
--------------------------------------------------------------------------
```

Рисунок 3.1. Результаты работы прямого и быстрого расчетов корреляции

Наблюдаем более быструю работу прямого алогоритма, что является очень странным. Обе функции проделывали расчет миллон раз, а затем на основе этого было вычисленно среднее время, затраченное на один рассчет корреляции этой функцией. Можем сделать предположение, что быстрый алгоритм работает эффективней с большим количеством данных. Стоит отметить, что оба алгоритма выполнили свою задачу и определили положение синхропосылки. В данном случае начало синхропосылки отмечено цифрой 3.

Чтобы проверить или опровергнуть наше предположение о том, что быстрый алгоритм работает эффективней с большим количество данных, проведем еще один опыт. Значения последовательности синхропоссылки менять не будем, однако увеличим наш сигнал, сгенерируя случайную последовтельность нолей и единиц большей размерности, например, равной 10000. Цикл, в котором расчитывается корреляция, уменьшим до 1000 итераций. Получаем следующие неутешительные результаты.(Рисунок. 3.2)

```
---------------------------------------------------------------------------
extended signal

correlation
the signal:  [ 1 -1 -1 ...,  1 -1  1]
the sync signal:  [ 1 -1  1]
time needed for correlation in an extended signal:  4.6906232833862306e-05
---------------------------------------------------------------------------
fast correlation
the signal:  [ 1 -1 -1 ...,  1  0  0]
the sync signal:  [ 1 -1  1 ...,  0  0  0]
time needed for fast correlation in an extended signal:  0.03282925748825073
correlation in an extended signal result:  [ 1 -1 -1 ...,  1 -1  3]
fast correlation in an extended signal result:  [ 1. -1. -1. ...,  1.  1. -1.]
```

Рисунок 3.2. Результаты работы прямого и быстрого расчетов корреляции для расширенного сигнала

Наблюдаем колоссальную разницу во времени работы обеих функций.

4. Выводы

Проделав лабораторную работу, рассмотрели понятие корреляции и научились пользоваться двумя алгоритмама её расчета. Установили, что прямой алогритм поиска корреляции более быстро рассчитывает корреляцию, чем быстрый алгоритм. При этом, увеличивая сигнал, время работы быстрого алгоритма начинает значительно отдаляться от времени работы прямого алгоритма. Отсюда можем сделать предположение, что некоторые встроенные функции, использованные при написании быстрого алгоритма, работают неоптимально.