Experimental Design in Computer Science
week1 report

Rintaro Sato
ID: 202120597

May 4, 2021

GitHub repo is available at
https://github.com/rintarooo/time_cpp_python

1 Objective

My research interest is Computer Vision and Image Processing, so I use C++ as a one of the my main language. Also, I work on the research in the field where Machine Learning is applied to Computer Vision so Python is also the language I often use for coding.

One thing I am curious about these coding languages when I code. Which one is faster? And how fast? I would like to compare them in terms of runtime speed; C++ vs Python.

Since Python is an interpreted language while C++ is a compiled language, everyone assumes that C++ would be much faster. However, No one has conducted an experiment comparing program's execution time between these languages in details.

My hypothesis is C++ runs 5 times faster than running in Python on average. This report aims to compare their running time.

2 Data Collection

2.1 Method

In this report, I implemented the following algorithm related to basic mathematical operation and Computer Vision/Machine Learning in both Python and C++ in order to compare them.

- Fibonacci sequence with recursive function
- Support Vector Machine(SVM) with OpenCV library

OpenCV is a popular library for Computer Vision. I executed the exact same program 30 times in each language, measured the time of each program's execution and then took the average of 30 execution times to compare their running speed.

2.2 Precaution

I exactly implemented the same program and the same algorithm order in both languages. I make sure outputs and return values of functions do not differ between Python and C++ programs.

In order to get precise execution time, I used the command **time** to time a program's execution. Python and C++ has its own module to time running program but I applied the same method to time for fairness. This **time** command has 2 types, shell built-in command and GNU command. I used former type, zsh built-in command. I modified the environmental variable TIMEFMT to change the output format. Time command outputs 3 values on the terminal, *real*, *sys* and *usr* under my environment. These values are defined by the following equation.

$$real = sys + usr + \underline{I/Owait} \tag{1}$$

real is the total time actually spent on a process. *usr* is generally called as User CPU time which is the time spent on the processor running program's codes while *sys* is the time spent on system calls for the kernel on the program's behalf.

I took the values of *usr* to get User CPU time.

3 Execute the experiment

3.1 Development Environment

My computer which is used as a test environment has the processor "1.6 GHz Dual-Core Intel Core i5", the memory "4 GB 1600 MHz DDR3", running in the OS "MacOS Big Sur version 11.2.2" system. I tested it out on only this computer since I have only this computer at the moment.

3.2 Technical Details

I made a shell script to automatically run program 30 times and export a CSV file. Each CSV file contains the string "execution time" in the 1st line

and measured execution time in the following 30 lines.

3.3 Fibonacci sequence

I wrote down the programs from scratch. I didn't use any specific external libraries. I implemented the algorithm which seeks for the digit in the 33rd position in Fibonacci sequence. I run the program using the shell script and measure running time using the command time. I left out the explanation of the algorithm in this report.

3.4 SVM with OpenCV

I run the program using the shell script and measure running time using the command time in the same way as I did in Fibonacci sequence. The codes are pasted from OpenCV tutorial. Since I run the programs multiple times in a low, I did comment out unnecessary parts in the original codes so that the output image(Figure 1) do not display on the screen and save it. I exploited some external libraries, Numpy and OpenCV. I left out the explanation of the algorithm in this report.

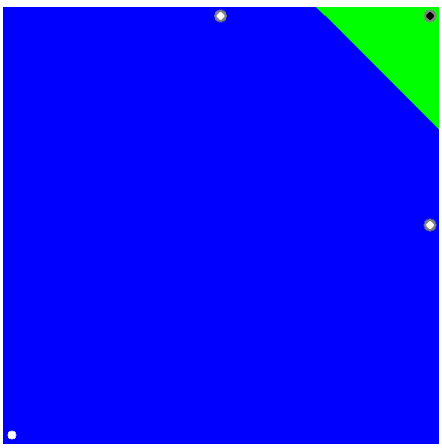

Figure 1: SVM hyperplane

4 Data Analysis

4.1 calculate point and interval estimators

I wrote down the R Markdown file . I did Confidence interval estimate. Python is distributed in a wide range around the average, while C++ is distributed in a very little range. This result indicates that C++ runs more stably and faster.

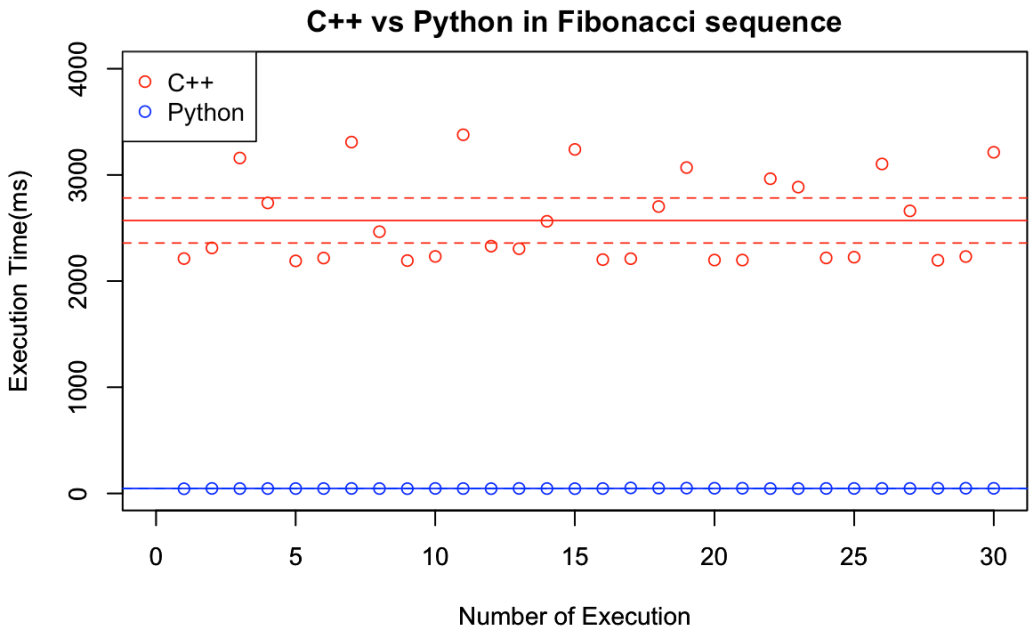

Figure 2: Fibo

5 Conclusion

5.1 limitation

There are many kinds of algorithm. It is almost impossible every single algorithm to implement and measure execution time of it. It would be desirable

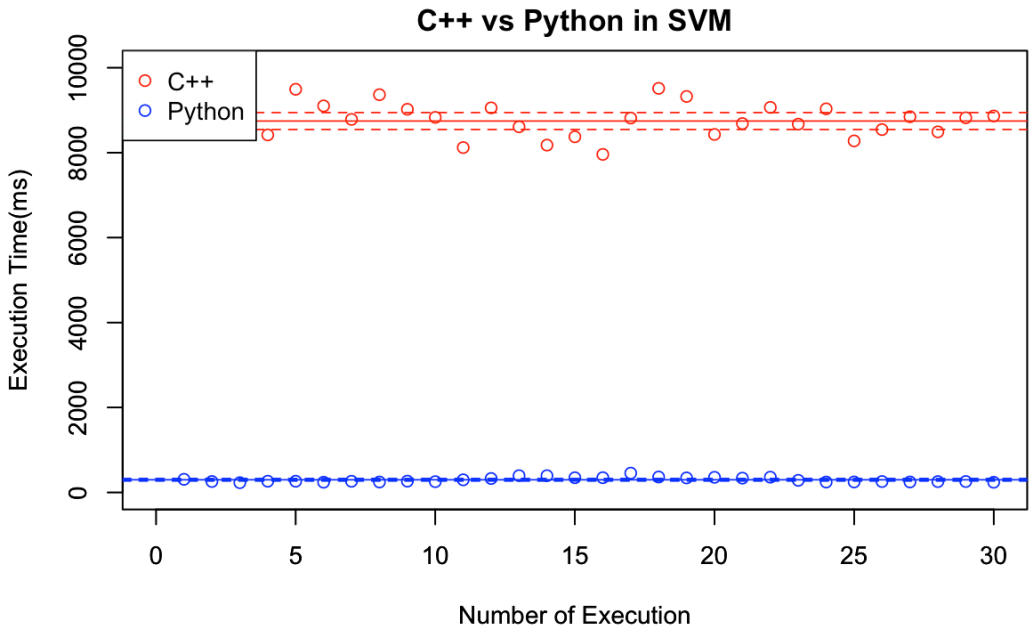

Figure 3: Fibo

to test on other environment(e.g., Windows, Linux) not only MacOS.

5.2 discussion

C++ is much faster as I expected. My hypothesis is C++ runs 5 times faster than running in Python. Based on my gathered data and data analysis, C++ runs more than 5 times faster. I need to dig into more wide range of algorithm in the next assignment or propose new hypothesis.