### 深圳杯数学建模挑战赛2024C题

编译器版本的识别问题

作为一种重要的工具，电子计算机自诞生以来，经历了极为快速的发展。区区百年的时间内，无论从体积、能耗、计算速度，还是应用能力等方面，电子计算机都出现了极为显著的变化。但要充分利用这一工具，必须使用能够被电子计算机解释执行的指令序列，即程序。

最早可用于在电子计算机上执行的程序通常使用机器语言（machine language）编制。由于该类语言并不直观，故它极大地限制了电子计算机的普及。为克服这一困难，1957年诞生了第一个自动编译器，FORTRAN。此后，大量性能更高并支持近乎自然语言的编译器被设计了出来，例如，著名的C/C++编译器，Python编译器等。编译器的出现极大地推动了电子计算机在当代的广泛应用。

为方便使用电子计算机，人们需首先按照一定的规则（即程序设计语言）将需要电子计算机完成的指令以特定的顺序集成在一起，形成脚本（即程序），然后使用编译器自动将脚本翻译为一系列机器语言的组合（即编译），编译器的编译结果最后会提交电子计算机执行。

随着程序设计语言的不断变化，编译器也会不断更新。例如，GCC（the GNU Compiler Collection）就已经更新到了13.2.0版本[1]。不同版本的编译器在编译同一程序脚本时，编译结果会存在一定的差异；相同版本的编译器在使用不同编译选项时，编译结果也会出现差异。能否利用编译结果差异区分编译器的版本？你们的任务是

问题1 使用GCC中不同版本的C++编译器编译附件1中的程序源代码[2]，并对比使用默认编译选项时的编译结果。找出区分这些编译结果的主要特征。

gcc编译分为四个阶段预处理🡪编译🡪汇编🡪链接，对应分别生成.i , .s , .o , .exe四个文件，其中前两个文件是可读文件，后两个是二进制文件，不可直接读取

在问题1，我们使用每个gcc的大版本生成的.s和.o文件进行读取，在.s当中统计指令出现的次数，在.o文件当中，我们统计各二进制码出现的次数。每个文件包含大量特征需要处理，我们选取指令出现次数，函数名翻译的汇编名，各二进制码出现的次数进行区分不同gcc版本

问题2 根据问题一中得到的特征，构造一个判别函数，使得能从各版本C++编译器使用默认编译选项时的编译结果，判别区分编译器版本。

很明显问题2是一个分类问题，由于特征值很多，所以不适用于SVM进行分类，因此我们选择使用决策树进行分类，均匀选择一个每个大版本的小版本的gcc的编译特征进行预训练，以此函数进行区分所有使用gcc版本，最终显示出大版本的版本号

问题3 用GCC中不同版本的C++编译器编译附件2中的源程序代码[3]，给出直接使用问题2中得到的判别函数区分编译器区版本的结果。

研究使用附件1、2原代码编译结果之一都能区分GCC中不同版本的C++编译器的判别函数。

问题4 给出几条提高由编译结果区分编译器版本的判别函数性能的建议，包括区分度和对原代码的泛化性。

参考文献

[1] https://bigsearcher.com/mirrors/gcc/releases/

[2] https://www.jb51.net/article/170405.htm

[3] <https://zhuanlan.zhihu.com/p/499738807>

sudo apt-get update

sudo apt install gcc-10 g++-10 gcc-11 g++-11 gcc-12 g++-12

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-12 120 --slave /usr/bin/g++ g++ /usr/bin/g++-12 --slave /usr/bin/gcov gcov /usr/bin/gcov-12

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 110 --slave /usr/bin/g++ g++ /usr/bin/g++-11 --slave /usr/bin/gcov gcov /usr/bin/gcov-11

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 100 --slave /usr/bin/g++ g++ /usr/bin/g++-10 --slave /usr/bin/gcov gcov /usr/bin/gcov-10

sudo update-alternatives --config gcc