Statistical Methods in CS - Project Idea
Google Code Jam (Contest Programming)

Motivation

In many courses and classes students are asked to implement a specific algorithm or solve a certain problem and to submit their solutions to an automatic grading tool. This is the case for many classes in algorithms and data structures, especially in online universities (like coursera.org).

Also very popular are programming contests like the ACM International Collegiate Programming Contest (ICPC) or the International Olympiad in Informatics (IOI) for high school students. There students need to solve algorithmic puzzles in a limited amount of time. Often partial score is given according to the runtime of the student's implementation.
One of the largest online coding competitions is the Google Code Jam (https://code.google.com/codejam), where over 10'000 programmers compete every year. In this competition each task has two sets of inputs, a small one and a large one. While the small one can often be solved using brute force or exhaustive search techniques, the large test set often requires a clever idea or algorithmic insight to solve it in the available time.

Goal

We want to study whether it is possible to automatically predict the runtime of a given implementation, without actually executing it. So in the code jam scenario this would mean whether we can predict if a program is capable of solving also the large test set.

Research Questions

- Which features indicate best, whether a program is fast or not?
- Does it depend on the specific task at hand which features are important?
- Can we derive best coding practices?
- Are there any clusters of similar solutions? (within the equally fast solutions)

Project Outline

- fetch and prepare all the source files from the code jam archive
- write tools to extract the features
- run the sources on input data to get timing information
- apply statistical methods
    - first Naive Bayes
        - all features binary (or over small discrete range)
    - if time permits, possible extensions:
        - structure into more complex Bayes network

        - allow for continuous values
        - logistic regression to allow for feature correlation

Application

Such an automatic evaluation tool could help both the organizers and participants of such contests, as well as instructors and students of such courses.

By detecting programs that stand out of the ordinary, organizers can effectively detect the submissions that they need to double check manually. Especially in real-time contests (like the IOI) or on grading systems with many users (like coursera) it is very hard to stay on top of all the submissions that come in, so it is essential that the system supports the organizers here.

From the students perspective such a tool can be helpful either while debugging live during the contest or when analyzing and comparing the own solution with others afterwards. Statements of the form “90% of the programs that are faster than yours are using std::map” might be very helpful starting points in order to improve the student’s learning process.

Literature

[We should search for more relevant literature]

Code Mining

- http://www.youtube.com/watch?v=18ZCKI9NySs
- Gao, Yongqin, Yingping Huang, and Greg Madey. "Data mining project history in open source software communities." *NAACSOS Conference 2004*. 2004 http://alliance.casos.cs.cmu.edu/events/conferences/2004/2004_proceedings/Gao_Yongqin.pdf.
- Rosenblum, Nathan, Xiaojin Zhu, and Barton P. Miller. "Who wrote this code? Identifying the authors of program binaries." *Computer Security–ESORICS 2011*. Springer Berlin Heidelberg, 2011. 172-189. http://pages.cs.wisc.edu/~nater/esorics-supp/Rosenblum11AuthorshipTR.pdf

Automatic Grading Systems and Plagiarism Detection

- Elenbogen, Bruce S., and Naeem Seliya. "Detecting outsourced student programming assignments." *Journal of Computing Sciences in Colleges* 23.3 (2008): 50-57. http://delivery.acm.org/10.1145/1300000/1295123/p50-elenbogen.pdf?ip=205.175.97.83&acc=PUBLIC&key=C2716FEBFA981EF1CF50BCE6855DF75EAEC7F9D83A1D11F2&CFID=210040376&CFTOKEN=86631419&__acm__=1367021241_6861815ef53034a6fdc0444027db5c14
- Matt, Urs von. "Kassandra: the automatic grading system." (1998). http://drum.lib.umd.edu/bitstream/1903/636/4/CS-TR-3275.pdf
- Jackson, David, and Michelle Usher. "Grading student programs using ASSYST." *ACM SIGCSE Bulletin*. Vol. 29. No. 1. ACM, 1997. http://delivery.acm.org/10.1145/270000/268210/p335-jackson.pdf?ip=205.175.97.83&acc

=ACTIVE%20SERVICE&key=C2716FEBFA981EF1CF50BCE6855DF75EAEC7F9D83A1D11F2&CFID=210040376&CFTOKEN=86631419&__acm__=1367020829_cffcb60412fdb4bc668b4536f029cb48

- Morris, Derek S. "Automatic grading of student's programming assignments: an interactive process and suite of programs." *Frontiers in Education, 2003. FIE 2003 33rd Annual*. Vol. 3. IEEE, 2003. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1265998
- Cosma, Georgina, and Mike Joy. "An approach to source-code plagiarism detection and investigation using latent semantic analysis." *Computers, IEEE Transactions on* 61.3 (2012): 379-394. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6086533

Data

- submitted and accepted programs from thousands of code jam participants
  - various programming languages, nationalities, task types
  - see http://www.go-hero.net/jam/13
- if useful also task submissions from the Swiss olympiad in computer science (also includes wrong submissions), see http://soi.ch

possible Features

- User (who wrote it)
- Code length
- Programming language
- # variables
- # nested loops
- correctness
- time in contest (not from code jam)
- number of wrong tries, total penalty time (only code jam)
- which task
- variable names used
- constants
- included libraries

Brainstorming for other questions to ask

- autodetect programming language
- classify task difficulty based on average code length needed, penalty time, etc.
- does the nationality of the coder give any indication?
- Naive Bayes / Bayes Network
  - P(task | code) (detect the task based on the variable names used)
  - P(correct | code, time in contest)
  - P(user | code) (might also be useful for fraud detection, similar code submissions)

- Hidden Markov Model
    - Model programming as state diagram, like
        - reading, coding, debugging (and this for each task)
    - predict who might submit in the near future (in an online setup)
        - try to extract successful strategies
    - might be hard as we don’t have labelled training data