Project Proposal Draft

Guang Yang

Research question: An analysis of the Codeforces rating system

Problem description:

Codeforces is a website that hosts competitive programming contests
As of 2018, it has over 600,000 registered users –Wikipedia
https://en.wikipedia.org/wiki/Codeforces
Competitors participate in rated contests, then their rating will change. I will research the way people's ratings change. Also, I would provide a strategy to get higher scores and push to higher ratings.

Description of Codeforces contest mechanism

Use official blogs to illustrate this part. It includes time penalty, submission penalty, successful hacks that increase scores, and unsuccessful hacks that decrease scores. The mechanism is a bit complicated.

My ranking algorithm: Elo method that applies to one-versus-many (I don't know if there is this kind of algorithm. If there is then I can just adopt that name)

In competition, contestants have their scores for each problem. I hypothesize that the rating change of an individual is based on the following aspects:
Rating before the contest (very likely)
Difficulty of each problem: different difficulty of problems indicates different weights in score/rating computation. It can be defined by the problem givers or reflected by participants' scores.
Score on each problem
Total score in the contest (very likely)
Ranking in the contest (very likely)

I will design a ranking algorithm: a modified version of Elo, referencing the official rating system of Codeforces at https://codeforces.com/blog/entry/20762. But this system was designed six years ago, so its method can only be as a reference. I may modify this method slightly to make it correspond to my data.

Some experiments

Next, I will do some experiments to test which aspects consist of a part of rating computation.

I will collect the data of 20 rated contests in different categories – different periods but within one year ago, and different divisions.
Then run the algorithm, check my result, and improve using the predictability index: using R-squared to measure the variance of the difference between my predicted rating change and the actual rating change.

After that, I will modify the method and re-run and repeat the process until I get a satisfactory result: hopefully improving R-squared to 95%

A clever strategy of attaining higher scores and ratings in Codeforces

This part will be finished after I developed and verified my Codeforces rating algorithm. It involves the efficient allocation of competition time, and which problems (of higher value or with lower time cost) to pick up when time is really limited.

Conclusion

Repeat the core of my algorithm.
Show the best R-square value in my experiments to prove that the algorithm is reliable.
Briefly introduce the strategy of attaining a higher rating.

References:

https://en.wikipedia.org/wiki/Codeforces
https://en.wikipedia.org/wiki/Elo_rating_system
https://dreamer.blue/blog/post/2018/02/26/codeforces_rating_system_algorithm.dream
http://codeforces.com/blog/entry/20762
http://codeforces.com/blog/entry/44214
https://blog.csdn.net/fu_xuantong/article/details/82145660
https://blog.csdn.net/chenshige/article/details/108053781

Many websites introduce the Codeforces rating system, but I decided to create one on my own and their systems just as references. Maybe I will not need to use them in my essay.