Learning To Grade Student Programs in a Massive Open Online Course

Anna Drummond
Yanxin Lu
Swarat Chaudhuri
Chris Jermaine
Joe Warren
Scott Rixner
Rice University

So, What's a MOOC?

- A MOOC is on online course open to everyone who signs up
- Students typically watch videos (instructional content)
- They interact with peers/TAs/instructors in online forums
- Take online exams/quizzes
- Turn in assignments for grading
- Then they might get some sort of certificate in the end!

So, What's a MOOC?

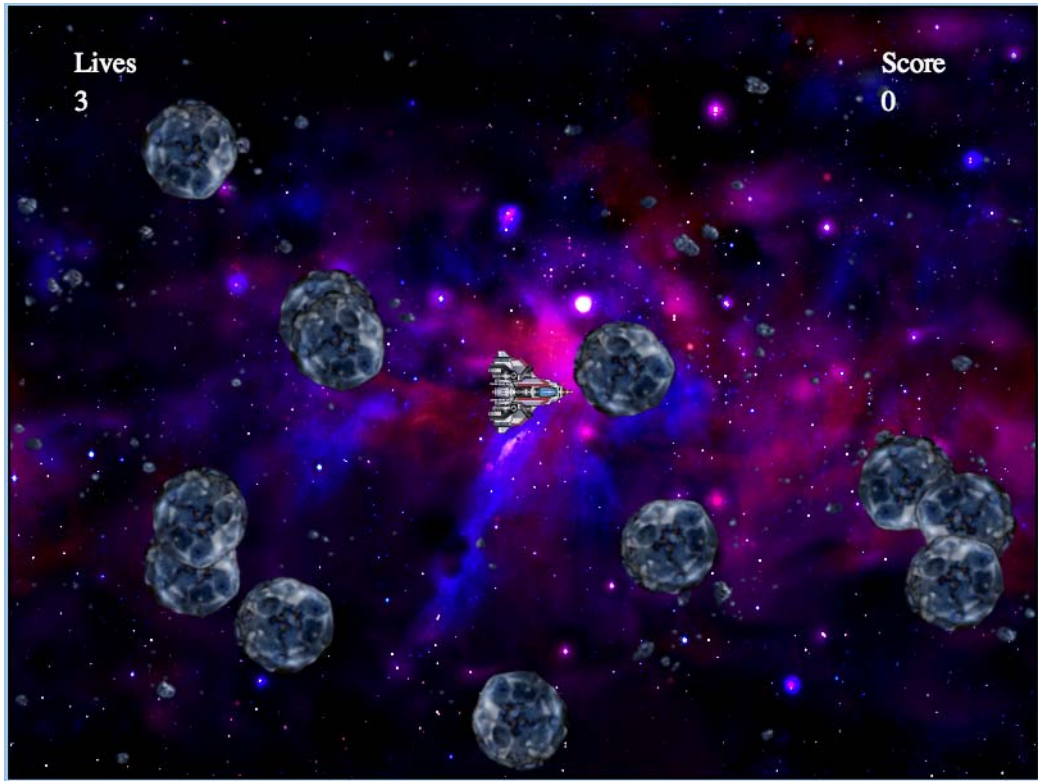
- A MOOC can be very large (in terms of number of participants)
- Example: IPP (run by last two authors of this paper)
  - — 350,000 students have signed up
  - — 30,000 have actually completed the course
  - — Spread over three (offerings) so far
  - — Each offering lasts two months

So, What's a MOOC?

- A MOOC can be very large (in terms of number of participants)
- Example: IPP (run by last two authors of this paper)
  - 350,000 students have signed up
  - 30,000 have actually completed the course
  - Spread over three (offerings) so far
  - Each offering lasts two months
- Managing that many students requires smart automation
  - Many opportunities for research in applied machine learning/data mining!

This Paper: Focus on Grading

- IPP covers basic interactive game programming
- Ex. programming assignment: Asteroids (classic arcade game)



This Paper: Focus on Grading

- IPP covers basic interactive game programming
- Ex. programming assignment: Asteroids (classic arcade game)
- Grading rubric contains items such as:
  - 1 pt: The program spawns multiple rocks
  - 1 pt: The number of lives decreases by one when the ship collides with a rock
  - 1 pt: The program correctly determines whether a missile and a rock collide
  - 1 pt: The score is updated appropriately after missile/rock collisions

This Paper: Focus on Grading

- IPP covers basic interactive game programming
- Ex. programming assignment: Asteroids (classic arcade game)
- Grading rubric contains items such as:
  - 1 pt: The program spawns multiple rocks
  - 1 pt: The number of lives decreases by one when the ship collides with a rock
  - 1 pt: The program correctly determines whether a missile and a rock collide
  - 1 pt: The score is updated appropriately after missile/rock collisions
- Note: no easy way to grade these automatically
  - They really require playing the game
- And instructors/TAs can't grade 10,000 submissions
- Hence, a class such as IPP is forced to rely on *peer grading*

Peer Grading

- Just like it sounds: Each program is graded by ~5 other students
- Great, idea, right?

Peer Grading

- Just like it sounds: Each program is graded by ~5 other students
- Great, idea, right?
- Well it is, but there's a huge variance in peer grading quality
  - — Some students grade carefully, writing hundreds of lines of comments
  - — Some do a terrible job, just giving full scores to every assignment

Peer Grading

- Just like it sounds: Each program is graded by ~5 other students
- Great, idea, right?
- Well it is, but there's a huge variance in peer grading quality
  - — Some students grade carefully, writing hundreds of lines of comments
  - — Some do a terrible job, just giving full scores to every assignment
- What we'd like to do:
  - — Predict quality of graders (possible to do early in the class; are reasonable signals)
  - — Predict quality of programs
  - — Get a post. dist. over true score and predicted, grader-assigned score
  - — Assign graders to programs so that we minimize

$$E\left[\sum_{\text{submitted programs}} \left(\text{true score} - \text{assigned score}\right)^2\right]$$

How to Predict Program Quality?

- This is the problem at the heart of the paper
- Basic idea... break the student program into a set of fragments
- Define a set of distance metrics over fragments
  - — Compare code fragment types... in IPP, this is the type of event handler (keydown, keyup, etc.)
  - — Jaccard distance between bags of system calls
  - — Jaccard distance between bags of guarded updates
  - — Tree edit distance between abstract syntax trees
  - — Graph edit distance between dependence graphs over library functions
  - — Graph edit distance between dependence graphs over functions + variables
- Then use those distance measures to do some sort of regression

Two Ways We Tried to Do Regression

- First way: kNN regression
  - — Turn each distance into a similarity
  - — Similarity between programs is cost of max matching between sets of fragments
  - — The do kNN regression using 1/similarity as the distance measure
  - — Are two flavors of this in the paper

Two Ways We Tried to Do Regression

- First way: kNN regression
- Second way relies on choosing a number of prototype fragments
- Then we can define the "closeness" between prototype p_k and the jth fragment in a program i as:

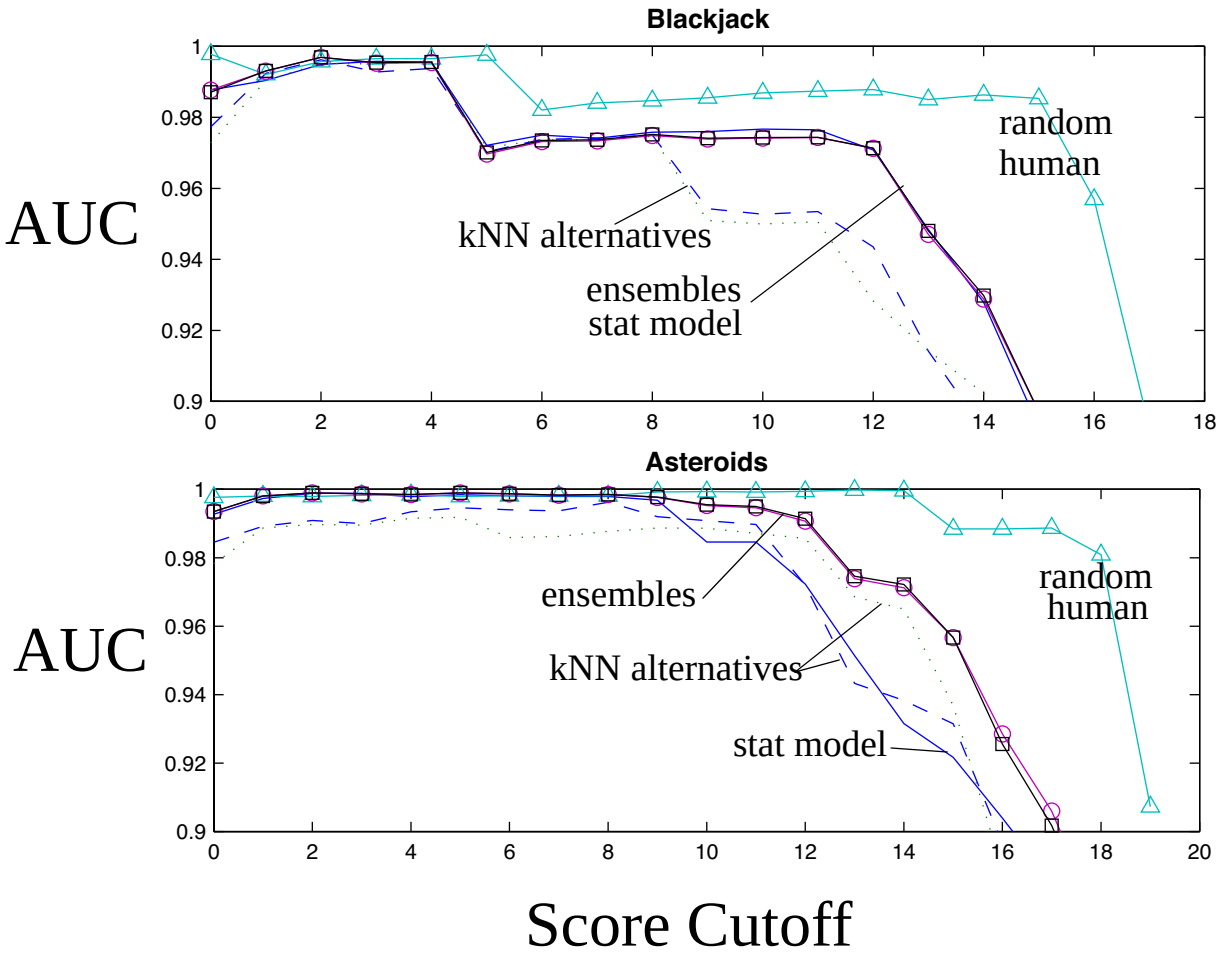
$$w_{ijk} = \frac{\exp(dist(x_{ij}, p_k))}{\sum_{k'} \exp(dist(x_{ij}, p_{k'}))}$$

- Assume score for program i is generated as

$$y_i \sim \text{Normal}(n_i^{-1} \sum_{jk} w_{ijk} s_k, \sigma^2)$$

- Learn the identity of the prototypes, as well as the weights using Bayesian methods

How Well Can We Do?



Questions?