Level Difficulty and Player Skill Prediction in Human Computation Games

Anurag Sarkar, Seth Cooper
College of Computer and Information Science
Northeastern University
Boston, Massachusetts, USA
sarkar.an@husky.neu.edu, scooper@ccs.neu.edu

Abstract

Human computation games (HCGs) often suffer from low player retention. This may be due to the constraints placed on level and game design from the real-world application of the game. Previous work has suggested using player rating systems (such as Elo, Glicko-2, or TrueSkill) as a basis for matchmaking between HCG levels and players, as a means to improve difficulty balancing and thereby player retention. Rating systems typically start incoming entities with a default rating. However, when applied to HCGs, incoming entities may have useful information associated with them, such as player behavior during tutorials and properties of the tasks underlying the levels. In this work, we examined using features derived from player behavior and level properties to predict their eventual Glicko-2 ratings in the HCG Paradox. We found that using regression produced rating estimates closer to the actual ratings than default or baseline average ratings. The use of rating systems allows a unified approach to predicting both player skill and level difficulty.

Introduction

Human computation games (HCGs) often suffer from low player retention, with most players leaving soon after starting, and the bulk of work being accomplished by a small fraction of players (Sturn et al. 2015). This contribution pattern holds true in other human computation systems, such as online citizen science projects (Sauermann and Franzoni 2015). Improving retention of members of the “long tail” of participants may help in increasing the effectiveness of human computation systems in general. This, in turn, may help advance the range of problems to which HCGs can be applied, which have included image labeling (von Ahn and Dabbish 2004), graph-theoretic problems (Cusack et al. 2010), neuron reconstruction (Kim et al. 2014), gathering common-sense facts (Siu and Riedl 2016), and protein folding (Cooper et al. 2010).

Due to the constraints of solving pre-existing problems, HCG designers do not have complete freedom to modify levels. Previous work (Cooper, Deterding, and Tsapakos 2016) has suggested that these constraints may contribute to low retention, and has suggested using existing player rating systems as a basis for matchmaking between HCG levels and players, as a means to improve difficulty balancing and thereby player retention. Rating systems—such as Glicko-2 (Glickman 2001)—are often used for player-versus-player matchmaking, though they have been used in other applications involving pairwise comparison of entities, including player-versus-level matchmaking. In such systems, each entity has a rating, which is updated based on the result of some comparison (for example, a player attempting a level, and either winning or losing).

Rating systems typically start incoming entities (i.e. players or levels) with a default rating. They are essentially treated all the same—a “blank slate” about which there is no information. However, in the case of HCGs, there may be some information about incoming entities that can be used to inform their initial rating. For players, this can include their behavior in the game up to the point where a rating is needed—for example, their performance in any tutorials or onboarding before they are to be matched with a level derived from a real task. For levels, this can include various properties of the underlying task—for example, size, estimates of complexity, and so on.

In this work, we gathered gameplay data in the HCG Paradox from players recruited using Mechanical Turk. We used this data to assign ratings to the players and levels using the Glicko-2 system. For use in a rating system, we treated each instance of a player successfully completing a level as a win for the player, and a player failing to complete a level as a win for the level. Thus, the most difficult levels (i.e. the levels that players most often fail to complete) are those that “win” the most, and therefore end up with the highest ratings, similar to a player-vs-player setting where the players with the highest ratings are those that win the most matches against other players. Hence, using rating systems offers a reasonable way of assessing both level difficulty and player skill, giving us a suitable ground truth on which to make our predictions.

We then used linear and Gaussian process regression to predict the ratings of players—based on their behavior in the game’s tutorial levels—and levels—based on properties of the underlying problem defining the level. We found that rating predictions based on regression were closer to the actual ratings than both the default Glicko-2 rating and a baseline method using average ratings. Predictions generally performed better for level ratings than for player ratings.
and dynamic to our puzzle games. Their predictions relied on, in addition to difficulty (on a subjective scale of 1-10) of levels in several approaches to predicting level difficulty have been developed. Closely related to our work is that of van Kreveld et al. (2015), who used a linear function to predict the level's actual difficulty) than the default rating. It is this rating can attempt to affect player engagement by personalizing the difficulty of a game for each player’s skill. Much work has examined the relationship between skill, difficulty, and engagement (Engeser and Rheinberg 2008; Alexander, Sear, and Oikonomou 2013; Denisova and Cairns 2015; Lomas et al. 2013).

### Dynamic Difficulty

Although the immediate goal of our work in this paper is predicting player and level ratings, this will be applied to matchmaking between players and levels, with the goal of assigning levels of appropriate difficulty to players based on their skill. This relates to dynamic difficulty adjustment (Hunicke 2005), which has received considerable attention in the game research community. Inspired by the flow theory of Csikszentmihalyi (1990), dynamic difficulty adjustment can attempt to affect player engagement by personalizing the difficulty of a game for each player’s skill. Much work has examined the relationship between skill, difficulty, and engagement (Engeser and Rheinberg 2008; Alexander, Sear, and Oikonomou 2013; Denisova and Cairns 2015; Lomas et al. 2013).

### Data Collection

**Paradox** (Dean et al. 2015) is a 2D puzzle HCG originally designed for the purpose of crowdsourced formal verification of software. Each level in the game represents a MAXSAT problem, which the players attempt to solve by making use of both manual and automated tools (represented as “brushes” in-game) to assign values to different variables. Players are assigned a score based on the percentage of clauses they are able to satisfy and can “complete” a level by reaching a pre-determined target percentage.

The game starts with a set of 9 hand-crafted tutorial levels, intended to introduce the game, which include instructions and hints on gameplay. After completing the tutorial, players then proceed to the challenge levels derived from MAX-SAT problems. In this work, we used levels generated from a number of MAX-SAT problems encoded in the DIMACS file format. We decided upon a set of 50 levels, 38 of which were derived from the SATLIB Benchmark Problems1 while the remaining 12 were generated using randomized algorithms.

For player recruitment, we used a Human Intelligence Task (HIT) posted on Amazon Mechanical Turk (MTurk) that paid $1.50 upon completion. Through this HIT, we recruited 150 players. Players were required to complete all tutorial levels, after which they advanced to the challenge section where they were served the 50 aforementioned levels in random order. Players were not served the same level more than once until they had seen all of the 50 levels. They did not have to complete these challenge levels, and had the option of skipping (moving to the next level without making any progress).

1http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

---

**Figure 1:** Screenshot from the HCG Paradox.
Table 1: Root Mean Square Error (RMSE) in rating predictions for players and levels.

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>Average</th>
<th>LR</th>
<th>GPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Players</td>
<td>211.1</td>
<td>198.0</td>
<td>195.2</td>
<td>197.3</td>
</tr>
<tr>
<td>Levels</td>
<td>403.8</td>
<td>412.0</td>
<td>358.9</td>
<td>319.2</td>
</tr>
</tbody>
</table>

Table 2: Percentage of predictions closer to the actual rating than baseline default predictions.

<table>
<thead>
<tr>
<th></th>
<th>LR</th>
<th>GPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Players</td>
<td>66%</td>
<td>65%</td>
</tr>
<tr>
<td>Levels</td>
<td>22%</td>
<td>68%</td>
</tr>
</tbody>
</table>

Table 3: Percentage of predictions closer to the actual rating than baseline average predictions.

<table>
<thead>
<tr>
<th></th>
<th>LR</th>
<th>GPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Players</td>
<td>55%</td>
<td>58%</td>
</tr>
<tr>
<td>Levels</td>
<td>68%</td>
<td>68%</td>
</tr>
</tbody>
</table>

Analysis

To predict ratings for players and levels, we derived several features to be used in regression. The features used for players were based on their behavior during the game’s tutorial levels. This is data that would be available before players started the game’s challenge levels and thus before they need to be assigned a rating. The following three features were aggregated across all tutorial levels:

- Total time taken (in seconds) to complete all tutorial levels.
- Total number of moves used to complete all tutorial levels.
- The amount by which the player’s total score for the tutorials exceeded the total target score of the tutorials.

The features used for levels were based on static properties of the levels themselves. These included properties of the underlying MAX-SAT problem:

- Average number of variables per clause.
- Average number of clauses per variable.
- 1 if all clauses are satisfied when all variables are set to true, otherwise 0.
- 1 if all clauses are satisfied when all variables are set to false, otherwise 0.
- Percentage of clauses satisfied when all variables are set to true.
- Percentage of clauses satisfied when all variables are set to false.

For our analysis, we used the player and level features described above, along with the Glicko-2 ratings obtained from the HIT, to train and test different approaches for predicting ratings. To fit regression models on the feature data for the purpose of ratings predictions, we used the scikit-learn Python module (Pedregosa et al. 2011). We used the following prediction methods:

- **Default**: Always predicting the default rating (of 1500). This was a baseline for comparing to using default values.
- **Average**: Predicting the average of all training ratings. This was a simple baseline prediction.
- **LR**: Using linear regression for predictions. scikit-learn’s LinearRegression was used.
- **GPR**: Using Gaussian process regression for predictions. scikit-learn’s GaussianProcessRegressor was used, with a kernel consisting of RBF, constant, and noise terms. Each feature was offset and scaled so that the minimum value in the training set was 0 and the maximum value was 1. Ratings were scaled by dividing by 2000.
To evaluate the different methods, we used leave-one-out cross validation. Scatter plots comparing the actual ratings obtained from the HIT and predicted ratings are shown in Figure 2 (players, linear regression), Figure 3 (levels, linear regression), Figure 4 (players, Gaussian process regression) and Figure 5 (levels, Gaussian process regression).

We used two approaches to compare the methods. First, we computed the root mean square error (RMSE) between the actual ratings and each of the prediction methods. These results are given in Table 1. Second, we calculated the percentage of predictions that were closer to the actual ratings than a baseline method. These results are given in Table 2 (for comparing against default) and Table 3 (for comparing against average). Overall, predictions performed relatively better for levels than for players.

We found that both linear regression and Gaussian process regression improved RMSE over using the baseline default rating—improving rating RMSE by up to 8% for players and 21% for levels. Gaussian process regression showed the greatest RMSE improvement for predicting level ratings, though it did not do much better than average for players. Average predictions actually had worse RMSE than default for levels. However, linear regression improved prediction RMSE over both baselines and thus may overall be a more consistently useful approach.

In terms of percentage of predictions that improved over baselines, both linear regression and Gaussian process regression improved a similar percentage of predictions, with Gaussian process performing slightly better than linear regression for players as compared to average predictions.

**Conclusion**

In this work, we examined using various methods to predict player and level ratings in a rating system used in an HCG. The use of rating systems allows a potentially unified approach to predicting both player skill and level difficulty.

We found that regression techniques improved over using the default rating; improvements were generally greater for levels than for players. This may allow accurate player and level ratings to be reached more rapidly by starting ratings out closer to their eventual values, and could have applications in improved difficulty adjustment using rating-based matchmaking. Although the predictions moved many ratings closer to their actual values, many are still far off. Thus, continuing to use the rating system during gameplay, and updating the ratings on-line based on outcomes from player attempts at levels remains useful.

In the future, this approach can be further validated on other HCGs or even entertainment games such as platformers or shooters, for example. The player features used, based on behavior in tutorials, included time spent, moves made,
and scores, which may generalize fairly easily to other games. However, level features were quite specific to *Para-
doxt*, though some might be applicable to games with an under-
lying graph structure to their levels. Additionally, other fea-
tures might be useful to incorporate, potentially including AI performance on levels (before they are given to players).

Beyond the scope of this current work, but additionally worth exploring, is the impact of using rating systems on player engagement in HCGs and other games (Sarkar et al. 2017). Rather than starting out players and levels with default ratings, it would be interesting to see if using features and parameters to inform initial player and level ratings as described in this work is able to more accurately assess ratings and thereby improve player engagement.

**Acknowledgments**

This work was supported by a Northeastern University TIER 1 grant. This material is based upon work supported by the National Science Foundation under Grant No. 1652537. We would like to thank the University of Washington’s Center for Game Science for initial *Paradox* development.

**References**


Cooper, S.; Khatib, F.; Treuille, A.; Barbero, J.; Lee, J.; Bee-
en, M.; Leaver-Fay, A.; Baker, D.; Popović, Z.; and Foldit

Players. 2010. Predicting protein structures with a multi-


