Player Behavior and Optimal Team Composition in Online Multiplayer Games

Hao Yi Ong[1], Sunil Deolalikar[2] and Mark V. Peng[3]

***Abstract*— We consider clustering player behavior and learning the optimal team composition for multiplayer online games. The goal is to determine a set of descriptive play style groupings and learn a predictor for win/loss outcomes. The predictor takes in as input the play styles of the participants in each team; i.e., the various team compositions in a game. Our framework uses unsupervised learning to find behavior clusters, which are, in turn, used with classification algorithms to learn the outcome predictor. For our numerical experiments, we consider League of Legends, a popular team-based role-playing game developed by Riot Games. We observe the learned clusters to not only corroborate well with game knowledge, but also provide insights surprising to expert players. We also demonstrate that game outcomes can be predicted with fairly high accuracy given team composition-based features.**

***Index Terms*— team performance, team composition, player behavior, video games, multiplayer games, game prediction**

I. Introduction

Online virtual worlds are an increasingly significant venue for human interaction. By far the most active virtual worlds belong to a genre of video games called massively multiplayer online role-playing games (MMORPGs), where players interact with each other in a virtual world [1]. In an MMORPG, players assume the role of in-game characters and take control over most of their characters' actions, often working in teams to accomplish a common objective, such as defeating opposing teams. Due to the shared, persistent nature of these virtual worlds, user behaviors and experiences are shaped by various social factors.

Besides profit-making, an understanding of these social dynamics would provide insight to human interactions in the real world and the potential of virtual worlds for education, training, and scientific research [2], [3]. Numerous prior studies in social sciences and management have investigated how team compositions can affect team performance [4], [5]. However, little is understood about player behavior and team performance and factors contributing to it in competitive MMORPGs. To address this, we develop a machine learning framework that uses game histories to learn player behavior clusters and predict the outcome of games given prior knowledge about the game and its players.

The contributions of this paper are twofold. First, we present several approaches that group player behaviors in online games. Second, we develop predictors that determine how likely it is that a team of players can emerge victorious given the said team's composition of players, all of whom may have different play styles. Specifically, we consider k-means and DP-means—an expectation maximization algorithm [6]—for clustering play styles and logistic regression, Gaussian discriminant analysis, and support vector machines for determining win/loss outcomes. The rest of the paper is structured as follows. Section II describes the target game of our numerical experiments and our data collection method. Sections III and IV demonstrate several methods and their effectiveness for learning play style clusters and outcome predictors. Some concluding remarks are drawn and future works mentioned in Section V.

II. Target Game Description

We begin with a description of the MMORPG used for our numerical experiments and the data acquisition method.

A. *League of Legends*

For this project we consider a popular MMORPG—the League of Legends (LoL). LoL is a multiplayer online battle arena video game developed and published by Riot Games with 27 million daily players [7]. In this MMORPG, a standard game consists of two opposing teams of five players. Each player assumes the role of one of over 120 different characters battling each other to destroy the opposing team's "towers"—structures that fall after suffering enough attacks from characters. A game is won when all of either team's towers are destroyed.

B. *Data set acquisition*

The developer of LoL has made the game's player statistics and match histories freely available through a web-based application programming interface (API) [8]. We randomly gathered over 100,000 instances of player statistics and over 10,000 instances of match histories from the 2013-2014 season. We then parsed and cleaned the raw game data to construct our training and testing sets, depending on the features we chose. Player statistics include performance indicators such as average damage dealt and number of wins. Match histories contain information such as participant ID numbers and character choices.

III. Behavioral Clustering

The target game's developers have grouped the 120 different in-game characters into six classes, such as assassin or support, which indicates the character's gameplay style. While these classes reflect the developers' design intent for the characters, they do not necessarily reveal the behavior of

[1]Mechanical Engineering Department, Stanford University
[2]Aeronautics and Astronautics Department, Stanford University
[3]Computer Science Department, Stanford University
Email: {haoyi,sunild93,mvpeng}@stanford.edu

actual players in games. Using statistics from various players, we present our feature selection method and the gameplay styles learned by applying various clustering algorithms to our data set. We validate our results and the insights derived from it with expert analysis from ranked players.

A. Feature selection

For our clustering algorithms, the features were 21 normalized player statistics, such as average damage dealt and money earned. The statistics were normalized over their range of values, preventing clusters from being formed due to order of magnitude differences between statistics. For instance, damage dealt values are often 7 orders of magnitude greater than kill streaks, which means small variations in damage dealt are erroneously considered as much more important than kill streaks if taken directly as feature values.

B. Clustering models

1) k-means: Given a set of observations, k-means clustering aims to partition them into k sets $\mathcal{S} = \{S_1, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares; i.e., find the minimizer $\mathcal{S}^\star$ of the distortion function:

$$\sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|_2^2, \tag{1}$$

where x is an observation and μ_i is the i^{th} cluster centroid.

In general, this problem is computationally difficult (NP-hard). For our clustering, we employ Lloyd's algorithm, which is a heuristic that consists of randomly choosing observations as cluster centroids and iteratively assigning observations to their closest centroids and updating the centroids with the mean of their respective clusters [9].

To select the number of clusters k, we run 10-fold cross validation over k to find a local optimizer. The scoring function for the cross validation is simply the average distortion given by (1) over the held-out sets.

2) DP-means: DP-means is a nonparametric expectation-maximization (EM) algorithm derived using a Dirichlet process (DP) mixture of Gaussians model. In other words, the user does not choose the number of clusters beforehand. The technique being the topic of a series of papers, we will only provide a brief description of the algorithm. The reader is referred to [10], [6] for a thorough review of DP-means.

The derivation of DP-means is inspired by the connection between k-means EM with a finite mixture of Gaussians model. Namely, the k-means algorithm may be viewed as a limit of the EM algorithm if all of the covariance matrices corresponding to the clusters in a Gaussian mixture model are equal to σI. As $\sigma \to 0$, the negative log-likelihood of the mixture of Gaussians model approaches the k-means clustering objective (1). Correspondingly, the EM steps approach the k-means steps in Lloyd's algorithm.

In the case of DP-means, [6] shows how to perform a similar limiting argument. Specifically, if the generative model for the EM algorithm was a DP mixture of Gaussians model with σI, letting $\sigma \to 0$ yields the objective function

$$\sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|_2^2 + (k-1)\lambda^2, \tag{2}$$

where $\mathcal{S} = \{S_1, \ldots, S_k\}$ is the set of clusters, x is an observation, and μ_i is the i^{th} cluster centroid. Note that, unlike in k-means, k is now a variable to be optimized over.

This leads to an algorithm with clustering assignments similar to the classical k-means algorithm and the same monotonic local convergence guarantees. (See Algorithm 1.) The difference is that a new cluster is formed whenever an observation is sufficiently far away from all existing cluster centroids, with some user-defined threshold distance λ. Intuitively, λ is a penalty on the number of clusters, on top of the original k-means distortion function.

Algorithm 1: DP-means

```
input : $\mathcal{X}$: input data, $\lambda$: threshold distance
output: Clustering $S_1, \ldots, S_k$, number of clusters $k$
$k \leftarrow 1$
$S_1 \leftarrow$ random observation $\{x^{\text{rand}} \in \mathcal{X}\}$
$\mu_1 \leftarrow x^{\text{rand}}$
repeat
    $X^{\text{perm}} \leftarrow$ random ordered permutation of $\mathcal{X}$
    // cluster assignments
    for $x \in X^{perm}$ in order do
        $c \leftarrow \operatorname{argmin}_{i \in \{1,\ldots,k\}} \|x - \mu_i\|_2^2$
        if $\|x - \mu_c\|_2^2 > \lambda^2$ then
            $k \leftarrow k + 1$
            $\mu_k \leftarrow x$
        else
            $\mathcal{S}_c \leftarrow \mathcal{S}_c \cup \{x\}$
    // centroid updates
    for $i = 1, \ldots, k$ do
        $\mu_i \leftarrow \frac{1}{|S_i|} \sum_{x \in S_i} x$
until $S_1, \ldots, S_k$ converge
```

We ran DP-means with 10-fold cross validation over a range of λ values, setting our scoring function as the average of the objective values from (2) over the held-out sets.

C. Results and interpretation

TABLE I
PLAY STYLE CLUSTERING SUMMARY RESULTS

	cross val.	param. range	clusters	cpu time
k-means	10-fold	$k = 5, \ldots, 24$	12	154.1 s
DP-means	10-fold	$\lambda = 2.5, 2.6, \ldots, 4.4$	8	65.4 s

D. Cluster visualization with PCA

Fig. 2 shows the result of applying principal component analysis (PCA) to reduce our feature dimension and visualize it in three dimensions. Notice that the data is clearly clustered

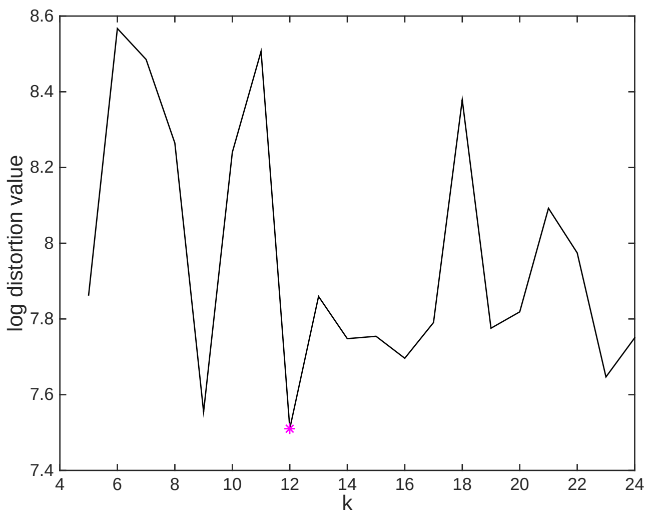

Fig. 1. The log distortion values show a local optimum at k = 12 over the range of 5 to 24 clusters (highlighted above as a magenta asterisk).

into 8 distinct groups, suggesting that in higher dimensions there are probably more clusters. Overlaying our 12-groups clustering from the k-means technique in color, we observe that they are consistent with the PCA results: Almost all points in any k-means cluster are in the same PCA cluster.

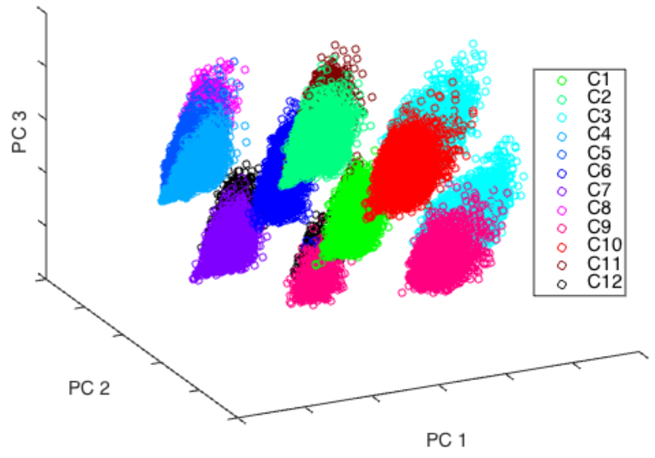

Fig. 2. Visualizing our data with 3 principal components reveals at least 8 distinct clusters. The 12-clusters k-means results are overlaid in color.

IV. Game Outcome Prediction

A. Feature selection

For our classification algorithms, the features are the two teams' player compositions. Associated with each team is a vector of counts of players that fall into a certain play style category, which were, say, derived from one of the clustering algorithms. The feature vector is the concatenation of the count vectors of teams 1 and 2. The labels for each sample are the win/loss indicator for the game, with 1 corresponding to a victory and 0 a loss by team 1 to team 2. For instance, there are 8 clusters, teams 1 and 2 have the count vectors $x_1 \in \mathbf{R}^8$ and $x_2 \in \mathbf{R}^8$, and team 1 beats team 2. The feature vector-label pair would then be

$$(x, y) = \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, 1 \right).$$

B. Baseline evaluation criteria

C. Classification models

1) Logistic regression:

2) Gaussian discriminant analysis:

3) Support vector machine:

D. Results and discussion

TABLE II
Outcome prediction summary results

	k-means			DP-means		
	train acc.	test acc.	cpu time	train acc.	test acc.	cpu time
LR	72.25%	68.75%	7.4 s	69.67%	67.11%	7.1 s
GDA	74.79%	70.14%	7.7 s	70.88%	68.39%	7.1 s
SVM	74.75%	70.39%	91.2 s	71.71%	69.21%	41.6 s

V. Conclusion and Extensions

Acknowledgments

The authors thank Prof. Andrew Ng and the course staff.

References

[1] E. Tomai, R. Salazar, and R. Flores, "Simulating aggregate player behavior with learning behavior trees," in *Proceedings of the 22nd Annual Conference on Behavior Representation in Modeling and Simulation*, W. G. Kennedy, D. Reitter, and R. S. Amant, Eds. BRIMS Society, Jul. 2013.

[2] W. S. Bainbridge, "The scientific research potential of virtual worlds," *Science*, vol. 317, pp. 472–476, 2012.

[3] M. D. Dickey, "Three dimensional virtual worlds and distance learning: Two case studies of active worlds as a medium for distance education," *British Journal of Educational Technology*, vol. 36, pp. 439–451, 2005.

[4] H. E. Spotts, "Evaluating the effects of team composition and performance environment on team performance," *Journal of Behavioral and Applied Management*, 2011.

[5] K. Hellerstedt and H. E. Aldrich, "The impact of initial team composition and performance on team dynamics and survival," *Academy of Management*, p. 6, 2008.

[6] B. Kulis and M. I. Jordan, "Revisiting k-means: New algorithms via bayesian nonparametrics," in *Proceedings of the 29th International Conference on Machine Learning*, J. Langford and J. Pineau, Eds. Omnipress, Jun. 2012.

[7] P. Tassi, "Riot's 'League of Legends' reveals astonishing 27 million daily players, 67 million monthly," *Forbes*, 2014.

[8] Riot Games, Inc., "Riot Games API," 2014. [Online]. Available: https://developer.riotgames.com/

[9] A. Ng, "Cs 229: Machine learning course notes," 2014. [Online]. Available: http://cs229.stanford.edu/materials.html

[10] T. Broderick, B. Kulis, and M. I. Jordan, "MAD-Bayes: MAP-based asymptotic derivations from Bayes," in *Proceedings of the 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester, Eds. Omnipress, Jun. 2013.