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 (LR), Gaussian discriminant analysis (GDA), and support vector machines (SVMs) 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]. Furthermore, LoL is a representative MMORPG of its genre, with many similar counterparts such as World of Warcraft's Dota 2 [8]—giving us a measure of generalizability to other games in its genre. 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) [9]. 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

[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

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 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 $S = \{S_1, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares; i.e., find the minimizer $S^\star$ of the distortion function:

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

where x is an observation and μ_i is the ith 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 [10].

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, which 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 [11], [6] for a thorough review of DP-means.

Recall that the standard mixture of Gaussians assumes that one chooses a cluster with probability π_c and then generates an observation from the k Gaussians corresponding to that chosen cluster. In contrast, the DP mixture of Gaussians is a Bayesian extension to this model that arises by first placing a Dirichlet prior $\text{Dir}(k, \pi_0)$ on the k mixing Gaussian coefficients (i.e., the probability of choosing a cluster) for some initial set of coefficients π_0 (e.g., uniform prior). As observations are made, the prior is updated and the mixture coefficients change to reflect these new knowledge.

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, suppose that the generative model for the EM algorithm was a DP mixture of Gaussians model with covariances equal to σI. Letting $\sigma \to 0$ for the DP mixture model yields the objective function

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

where $S = \{S_1, \ldots, S_k\}$ is the set of clusters, x is an observation, and μ_i is the ith 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, λ: 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**
        $c \leftarrow \text{argmin}_{i \in \{1, \ldots, k\}} \|x - \mu_i\|_2^2$
        if $\|x - \mu_c\|_2^2 > \lambda^2$ **then**
            $k \leftarrow k + 1$
            $\mu_k \leftarrow x$
        else
            $S_c \leftarrow S_c \cup \{x\}$
    // 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. Numerical results

Due to the random initializations, we ran 20 trials for each clustering algorithm in order to obtain the best locally optimal centroids. These optima correspond to 12 and 8 clusters for k-means and DP-means, respectively. All code were implemented in MATLAB and computations executed on a 2.7 GHz Intel Core i7 with 8 GB RAM. Figure 1 shows an example of the log of distortion values attained over the range of k values for the k-means algorithm ran with 10-fold cross validation. Table I summarizes the results for the clustering algorithms. The recorded computation times were averaged over the 20 trials, and do not include preprocessing and transforming data into features, etc.
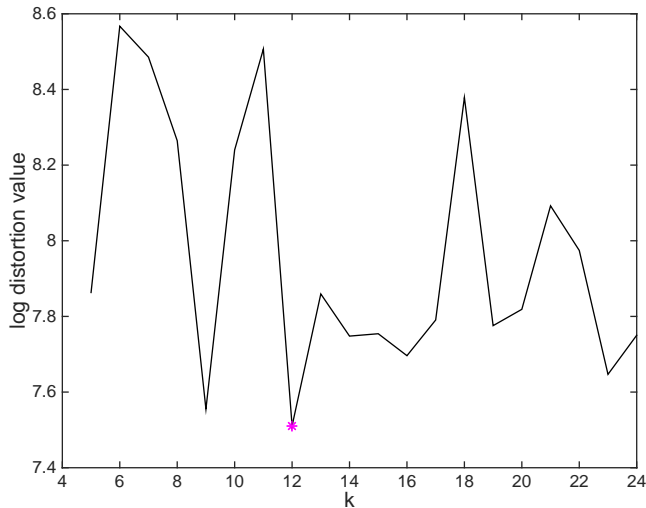

Fig. 1. Best trial out of 20: The log distortion values show a local optimum at k = 12 over the range of 5 to 24 clusters (magenta asterisk).

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 interpretation

Surprisingly, our consultations with expert, highly-ranked (top 0.2% worldwide) LoL players corroborated the correctness of the behavior clusters learned by our algorithms. By checking the centroid values corresponding to each feature and using information about the frequency of in-game characters used for each cluster, these expert players were able map each cluster to a specific gameplay type that they had experienced in-game. This suggests that our clustering were intuitively correct. The mappings determined for the 12-clusters k-means result are as follows.

- **Ranged physical attacker** Clusters 1, 7, and 9
  - Players who maintain distance from fights while dealing high damage with long-range attacks

- Players in each cluster differ in risk attitudes, such as whether they attack deeper in enemy territory
- **Ambusher** Clusters 3, 8, 11, and 12
  - Players who move stealthily around the battlefield and engage in quick, close-ranged combat
  - Some players prefer a team oriented style, whereas others prefer a more "lone wolf" approach
  - Includes "hybrid" roles with other behavior clusters
- **Team support** Cluster 5
  - Players who typically assist ranged physical attackers (healing, cooperative attacks, etc.)
- **Magic attacker** Clusters 6 and 10
  - Players who rely on magic-based attacks; as opposed to physical damage in the above clusters
  - Differ in preference for close- or ranged-combat
- **Miscellaneous** Clusters 2 and 4
  - No clear style preference
  - Differs in skill: either a novice player or prefers an all-around gameplay style

Interestingly, we notice from expert analysis that there appears to be a hierarchy of clusters. For instance, clusters 6 and 10 fall under the broader "magic attacker" category. This suggests that we might consider other clustering models than k-means or DP-means, as these methods assign each observation to only one cluster. We address this further in Section V.

E. 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 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.
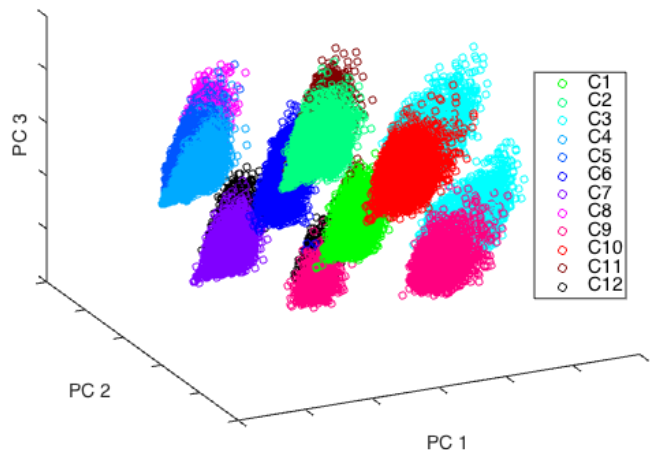

Fig. 2. Visualizing our data with 3 principal components reveals at least 8 distinct clusters. The 12-clusters k-means results are overlaid in color.

IV. Game Outcome Prediction

We illustrate the accuracy of game outcome predictors that use team composition features based on our gameplay style clusters learned in the previous section. We present our feature selection, the classification models used to learn our predictors, and the accuracies for our predictors.

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

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

where x is the feature vector and y is the sample binary label.

B. Classification models

To obtain the best win/loss outcome predictor, we consider different classification models. To determine the accuracy of our predictors learned using the various models, we held out 10% of our total sample set (over 130,000 in total) for training, and used the held-out samples for testing.

1) Logistic regression: For this model, we use the Bernoulli family of distributions to model the conditional distribution of winning or losing given the team composition features. That is, adhering to our notation introduced above, $y \mid x; \theta \sim \text{Bernoulli}(\phi)$, where θ is our model parameter and $\phi = h_\theta(x) = 1/(1 + \exp(-\theta^T x))$ is our hypothesis, which is derived from formulating the Bernoulli distribution as an exponential family distribution. To learn our model, we find a parameter θ that maximizes the log-likelihood function

$$\ell(\theta) = \log \prod_{i=1}^{m} p\left(y^{(i)} \mid x^{(i)}; \theta \right) \tag{3}$$

$$= \sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)} \right) + \left(1 - y^{(i)} \right) \log \left(1 - h_\theta\left(x^{(i)} \right) \right), \tag{4}$$

where m is the sample set size. We used stochastic gradient ascent to efficiently find the optimizer $\theta^\star$.

2) Gaussian discriminant analysis: In this model, we assume that the input features x are continuous-valued random variables and model $p(x \mid y)$ using a multivariate normal distribution. In other words, we use a generative learning model. In our case,

$$y \sim \text{Bernoulli}(\phi) \tag{5}$$
$$x \mid y = 0 \sim \mathcal{N}(\mu_0, \Sigma) \tag{6}$$
$$x \mid y = 1 \sim \mathcal{N}(\mu_1, \Sigma), \tag{7}$$

where μ_0, μ_1, and Σ are the means and covariance of the Gaussian distributions. Here, we maximize the log-likelihood of the m-samples data

$$\ell(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma \right). \tag{8}$$

The result of maximizing ℓ with respect to the model parameters is a set of exact analytic equations [10], which we compute directly. The derivation of these equations are simple, and we omit them for brevity.

3) Support vector machine: Assuming that our data are separable with a large "gap," a support vector machine model posits that the size of the geometric margin between some observation point and the decision boundary is proportional to "confidence level" that the observation is classified correctly. The result of this model is an optimization problem that seeks the maximum margin separating hyperplane for our samples.

For our problem, we use ℓ_1 regularization since we are uncertain about whether our data is linearly separable (e.g., outliers, erroneous data). The resulting problem is solved using the sequential minimal optimization algorithm [**?**].

C. Evaluation criteria

To evaluate the usefulness of game outcome predictor models with features based on our learned behavior clusters, we compare them against a baseline predictor with features based on the game developers' official gameplay classes. As introduced in Section III, the game developers have grouped the in-game characters into six broad categories, such as assassin or support, which supposedly reflects the character's gameplay style. We learn a logistic regression model with features constructed using these categories and use the 10% hold-out method for cross validation.

D. Results and discussion

To ensure fairness of results, we ran 20 trials for each model to determine the predictor accuracies, which are based on different randomized train and test sets. As with our clustering algorithms, all code were implemented in MATLAB, the computations were executed on a 2.7 GHz Intel Core i7 with 8 GB RAM, and the computation times were averaged over the 20 random trials. Again, these times do not include preprocessing and transforming data into features, etc.

As we observe in Table II, the best predictor learned using our behavior clusters-based features uses an SVM model with features derived from our k-means clustering. This predictor did significantly better (16% better) than the baseline algorithm on the test sets, and the other predictors were also competitive—all were only less accurate by a tiny percentage.

Other than illustrating the relatively high accuracy of our team composition-based outcome prediction approach, this result also implies that our behavior clusters learned had more descriptive power than the official game developers' version. This indirectly concurs with what we have shown from our clustering models: The official gameplay style

4

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

categories that were used for the baseline algorithm do not necessarily correspond to the behaviors of actual players in games.

Overall, our results validate our framework of first clustering players by their gameplay style and then using team composition features based on these learned styles to predict team performance. And since our target game is a representative title for games of the same type (i.e., team-based role-playing games), we expect this framework to also be effective and generalizable to other multiplayer games.

V. CONCLUSION AND EXTENSIONS

In this brief, we have presented an algorithmic framework for outcome prediction: By learning in-game player behavior categories through clustering and using them in features for game outcome predictors based on classification models, we are able to determine wins and losses with over 70% accuracy for our target game. This approach could be used to evaluate how team compositions can affect performance in games other than the one we have considered.

Future work will include adding time-dependent player statistics features. Unlike the overall game statistics we used, these timed statistics might give an additional layer of descriptive power, allowing the model to differentiate between clusters based on how players behave early and later in the game. This might lead to a better features for a more accurate team composition-based win/loss predictor. As another extension, we could also consider different clustering models, such as one that captures the ostensibly hierarchical clustering seen in the expert analysis of the k-means results. For instance, the BP-means model described in [11] is designed to capture such hierarchical clustering relationships.

ACKNOWLEDGMENTS

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] S. Ford, "League of Legends: Marc Merrill Q&A," *Warcry Network*, 2009. [Online]. Available: http://www.warcry.com/articles/view/interviews/5686-League-of-Legends-Marc-Merrill-Q-A

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

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

[11] 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.