Clustering Players Time-Series Data: A Case Study

Yuanli Pei*
Oregon State University
Corvallis, OR 97330
peiy@eecs.oregonstate.edu

Amy Tan
Zynga Inc.
San Francisco, CA 94103
atan@zynga.com

Moises Goldszmidt*
Samsung Research America
Mountain View, CA 94043
m.goldszmidt@samsung.com

Alexandros Ntoulas
Zynga Inc.
San Francisco, CA 94103
antoulas@zynga.com

Abstract

The game industry presents a very interesting source of challenges to machine learning methods and algorithms. Games continuously collect data about the basic actions of players. Such data are very useful for the game developers to extract actionable information, in order to both delight players with better game experiences and improve business metrics. In this paper, we explore an approach to address a key enabler for understanding player behaviors . Given players' time series raw data about their actions, our goal is to identify (1) what are the latent states that govern that behavior, and (2) what are the natural clusters of players given their latent state dynamics. Our approach is based on employing Hidden Markov Models (HMMs) to identify the basic latent states of the players, and then use a clustering method based on Dirichlet Process to group the players according to their time series of the latent states. We present our preliminary results on real data taken from one of the strategy games developed by Zynga Inc.

1 Introduction

The game industry is steadily gaining grounds in the competition for our digital time. Besides the traditional economic and business analysis reports, evidence of this fact includes the declared interest of companies like Amazon (purchased Twitch), Sony (playStation), Microsoft (XBOX), to name but a few. Games are also an incredibly rich source of data about human behaviors ranging from social interactions to economical and rational decision making.

From the perspective of improving the player experience and also from the business aspects of enhancing retention, engagement and payments, it is very important for game developers to understand player behaviors and take appropriate actions accordingly. As players change their playing strategies and patterns during their involvement with the game (which can take years), and as these patterns are clearly not independent from previous playing behaviors, the appropriate analysis of these patterns should be done by looking at the time series of the player actions. These time series are multivariate in nature, as players can take over hundreds of actions in some of these games, and such actions are further parameterized by real values.

In this paper, we explore an approach to characterizing player behaviors from the multivariate time series of their raw actions in the game. Our method involves two steps 1) inducing a time series of latent states that abstract players' raw actions and 2) clustering these time series of latent states. To induce the time series of latent states we rely on Hidden Markov Model (HMM) [9], with the

*This work was done while Yuanli Pei interned and Moises Goldszmidt worked at Zynga Inc.

playing states as the hidden nodes and the player action statistics at each time period as the observed features. We fit the HMM model and induce the latent states by applying the Viterbi decoding algorithm. We then find the groupings of players using an unsupervised clustering method which models this grouping as a Bayesian version of a Dirichlet Process [6].

The first step provides game designers and product managers with visibility into what are the individual playing modes so that they can examine how a mixture of game actions give raise to strategy and enable them to plan for changes. The second step provides visibility into the aggregate behavior of players so that business decisions can be made regarding key metrics such as retention, engagement, and monetization.

Of course, there exists previous work on clustering time series data with HMM models [1, 2, 3]. However, most of the work we reviewed focuses on finding out the final clustering result, while as explained above, we are also interested in studying the latent states of each player at each time period and the interpretability of the final model. In the game domain, Menéndez et al. [7] propose an approach to extract user profiles from player time series data (based on pre-defined metrics). That task is very different from ours as we are interested in also identifying the player latent states.

We test our approach on a strategy game developed by Zynga Inc. Our method found three basic latent states generating all the daily raw actions, and the clustering results reveal three groups that contain players with similar patterns of state transitions and strategy changes over time. These groupings enabled product managers to estimate correlations with key business metrics and what changes would induce changes in these metrics. The understanding of player latent states provided evidence for hypothesis and A/B testing experiments to improve on player experience and their progress through different levels of the game.

2 Methodology

Let $X_{it} = [X_{it1}, \ldots, X_{itD}]^\top$ be the measurements of player i at the t-th time epoch, where D is the number of features. We assume that time is discrete and advances in epochs. The measurements of player i from $t = 1$ to $t = T$, i.e. her time series of actions in the game, is denoted as $X_i = [X_{i1}, \ldots, X_{iT}]^\top$. The time series of data for N players is denoted by $X = \{X_1, \ldots, X_N\}$. We remark that, for different users, the total time epochs may not be the same; for simplicity of exposition, in this paper we assume that the time series for all users are of the same length.

Modeling Latent Players States with HMMs. We assume that there is a latent playing state that controls user behavior at each time period, and additionally that the state evolves as the player changes her strategy overtime. We represent each players' data using a Hidden Markov Model (HMM) [9] chain with length T, the total time epochs. Let Y_{it} be the hidden state representing the i-th player's latent state at time t, and Y_{it} will be regarded as discrete taking on S values $\{1, \ldots, S\}$.

The mechanism of the player HMM model is as follows: 1) initially, a player starts with state $Y_{i1} \in \{1, \ldots, S\}$ according to an initial distribution π, with π_s being the probability of starting at state s; 2) all the Y_{it}'s evolve according to the *Markov property*: given Y_{it-1}, the state Y_{it} is independent of all the states prior to $t-1$, and the transition matrix is A, with A_{rs} being the probability of transitioning from state r to state s; 3) at each time t, the observations X_{it} depend only on the state Y_{it} parametrized by B, with B_s controlling the probability of observing X_{it} at state $Y_{it} = s$. Given the observed data X and the number of states S, we estimate the parameters, i.e. the transition matrix A, the emission matrix B, and the initial distribution π, by maximizing the likelihood of the observations. As usual, the free parameter S is fitted via a scoring function. In this work we rely on BIC [10].

After estimating the parameters, we find the state sequence $Y_{i1}, \ldots, Y_{iT}$ for each user by maximizing $P(Y_{i1}, \ldots, Y_{iT} | X, \pi, A, B)$ using the Viterbi algorithm [9].

Clustering Player Behaviors. Our approach to clustering consists of adapting the method proposed in [11], as they also looked at clustering time series of integer data (albeit in a completely different domain). We adopt the mixture of Dirichlet process model (DP) for clustering [6]. Thus, we assume that the clusters evolve according to a Dirichlet distribution with parameter α.

Let $Y = [Y_1, \ldots, Y_N]^\top$ be the state transitions for all the players, where $Y_i = [Y_{i1}, \ldots, Y_{iT}]^\top$ denotes the i-th player's states from 1 to the T-th time. As it is common in this approach, we use Z_i

as an auxiliary variable denoting the cluster assignment for the i-th player. We use K to represent the total number of (unknown) clusters. Again, given our model, the number of clusters will be fitted automatically as part of the model, and will be continuously updated as we collect more data.

We assume that each cluster k generates a Markov chain parametrized by $\{\lambda^k, \Phi^k\}$, where λ^k is the $S \times 1$ vector of the initial state distribution, and Φ^k is the $S \times S$ dimensional transition matrix. For parameters in each cluster, we use the prior $G_0(\{\lambda^k, \Phi^k\}) = Dir(\hat{\pi}) \prod_{s=1}^{S} Dir(\hat{B}_{s.})$, where $\hat{\pi}$ and $\hat{B}$ are the estimated parameters at the first step. Then, we can determine the conditional probability:

$$P(\{\lambda^k, \Phi^k\}_{k=1}^K | Z) = \prod_k G_0(\{\lambda^k, \Phi^k\}) \,. \tag{1}$$

Given the clustering model, the data likelihood of state transitions for all players is

$$P(Y|Z, \{\lambda^k, \Phi^k\}_{k=1}^K) = \prod_{i=1}^{N} \left(\prod_{s=1}^{S} (\lambda_s^{Z_i})^{\mathbf{1}[Y_{i1}=s]} \prod_{r=1}^{S} \left(\Phi_{rs}^{Z_i}\right)^{n_{irs}} \right) \,, \tag{2}$$

where $\mathbf{1}[\cdot]$ is the indicator function, and n_{irs} is the total number of transitions from state r to state s for the i-th player.

We follow a Bayesian approach to inference, and even though some parts can be done in closed form, we still need to resort to sampling methods for computing the posterior. Following [11], we use a collapsed-space sampling method [8, 5] to obtain samples from the reduced-spaced posterior distribution $P(Z|Y)$, instead of the full-space distribution $P(Z, \{\lambda, \Phi\}|Y)$. This allows for easy sampling steps and faster convergence rate. The reduced-space posterior distribution is

$$P(Z|Y) \propto P(Z, Y) = P(Y|Z)P(Z).$$

The likelihood $P(Y|Z)$ can be computed by integrating out the cluster-specific parameters $\{\lambda^k, \Phi^k\}_{k=1}^K$. Substituting (1) and (2), we obtain

$$\begin{aligned} P(Y|Z) &= \int P(Y|Z, \{\lambda^k, \Phi^k\}_{k=1}^K) P(\lambda^k, \Phi^k|Z) d\lambda^k d\Phi^k \\ &= \prod_{k=1}^{K} \left[\frac{\prod_s \Gamma(\bar{\pi}_s) \Gamma(\sum_s \hat{\pi}_s)}{\Gamma\left(\sum_s \bar{\pi}_s\right) \prod_s \Gamma(\hat{\pi}_s)} \right] \times \prod_{k=1}^{K} \prod_r \left[\frac{\prod_s \Gamma(\bar{B}_{rs}) \Gamma(\sum_s \hat{B}_{rs})}{\Gamma\left(\sum_s \bar{B}_{rs}\right) \prod_s \Gamma(\hat{B}_{rs})} \right] , \end{aligned}$$

where $\bar{\pi}_s = \hat{\pi}_s + \sum_i \mathbf{1}[Z_i = k, Y_{i1} = s]$, and $\bar{B}_{rs} = \hat{B}_{rs} + \sum_i n_{irs} \cdot \mathbf{1}[Z_i = k]$.

Sampling Z from a Dirichlet distribution can be equivalently done as follows [8]: set $Z_1 = 1$; for subsequent players, sample Z_i according to the following distribution

$$\begin{aligned} P(Z_i = k | Z_1, \ldots, Z_{i-1}) &= \frac{|\{i' < i : Z_{i'} = k\}|}{i - 1 + \alpha} \,, \quad \text{for } k \in \{Z_{i'}\}_{i' < i} \\ P(Z_i = Z_{i'}, \forall i' < i | Z_1, \ldots, Z_{i-1}) &= \frac{\alpha}{i - 1 + \alpha} \,, \end{aligned}$$

where $|\cdot|$ denotes the number of elements in a set.

3 Experiments

We apply our method to one of Zynga's strategy games where the goal is to conquer all the battlefields in a global map (players can play against other players or against the game itself). The players need to build/upgrade base resources with weapons and troops, which in turn requires game points that can be obtained from winning battles. Thus, players need to tradeoff between building resources and conquering battlefields.

We subsample players from a dataset of 10 consecutive days data after installation. Our final dataset contains 1719 players. Each player is characterized with 67 features at each day, and we select 5 important features based on prior experience and feature selection methods: `Pvp` (people vs people battle), `Pve` (people vs machine battle), `Points` (number of points gained), `Session` (number of sessions started), `LevelUp` (whether a player levels up) and `isPayer` (whether the player paid). We model `Points` with a Gaussian distribution, `Session` with a Poisson distribution, and the remaining features with a Bernoulli distribution.

Table 1: Discovered Playing States

Feature / State	*Aggressive*	*Defensive*	*Moderate*
Prob. `Pvp`	0.2472	0.0295	0.0947
Prob. `Pve`	0.2430	0.0581	0.1044
Mean `Points`	88.10	1238.36	476.22
Mean `Session`	4.58	34.19	12.95
Prob. `LevelUp`	0.1664	0.0177	0.0553
Prob. `Pay`	0.0031	0.0956	0.0320

Table 2: Clustering results.

Method	#Cluster	DB
Ours	**3**	**0.968**
Kmeans	5	2.628
GMM	4	1.803

Identifying Players' Latent States. Our experiments yielded 3 states that were interpretable given the distributional characteristics of the features (see Table 1): ***Aggressive***, ***Defensive***, and ***Moderate***. The ***Aggressive*** state captures the mode where the players focus on conquering battles, while the ***Defensive*** state describes the stage that they acquire the resources and build their base. The ***Moderate*** is a mixture of the two. These states interestingly identified the design of the game explained previously, namely, the players have to conquer battles and build resources intermittently in order to progress.

Figure 1 plots the transitions for all the players among the 10 days (decoded using Viterbi). The result shows that most of the users starts with the ***Moderate*** or ***Defensive*** state, and then gradually transition to the ***Aggressive*** state. This is consistent with the initial game design as it is difficult for players to start with many battles due to resource restrictions, but they ultimately need to become aggressive and conquer all the battlefields.

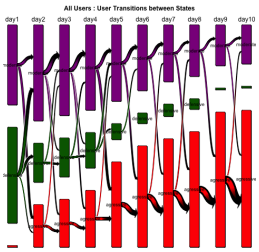

Figure 1: State transitions of all the players at the first 10 days after installation.

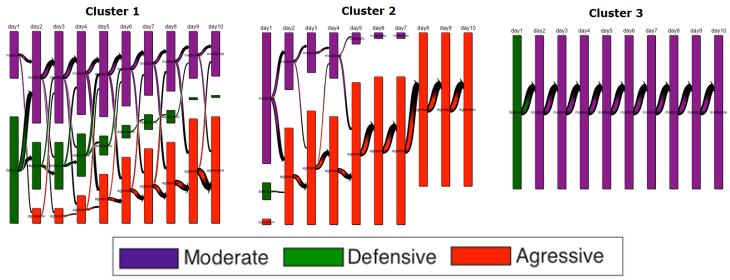

Figure 2: State transitions within three clusters found by our method.

Clustering Results. We compare the results of our clustering method with two alternatives: Kmeans and Gaussian Mixture Model (GMM), using a popular internal clustering evaluation metric *Davis-Bouldin (DB) index* [4] (we remind the readers that we don't have the ground truth). We tune the number of clusters for Kmeans using DB index, and report the one that has the best (the smallest) DB index value. Table 2 reports the clustering results of all methods, where our method outperforms the alternatives. We also plot the state transitions within each clusters and found that our method produces the most meaningful results. Here, we show the within cluster transitions found by our method in Figure 2.

4 Conclusion

There are many ways to cluster time series and in previous sections we reported on some of those. The specific approach we took, to first identify latent states and then cluster the time series of the latent states, is motivated by the need to have game designers interact with the results. There are mainly two kind of interactions we aimed at: first, the results have to be interpretable by the game designers, and second we need to give game designers a way to provide side information and influence the results. We found that by using latent states, and "naming" the latent states using the properties of the distribution of the actions they generate, the game designers and product managers could get to actionable information from the clustering. The use of the Dirichlet process approach also allows them to provide side information regarding which players should and should not be in the same clusters. We are currently studying the best ways to quantify these interactions.

References

[1] Manuele Bicego, Marco Cristani, and Vittorio Murino. Unsupervised scene analysis: A hidden markov model approach. *Computer Vision and Image Understanding*, 102(1):22 – 41, 2006.

[2] Manuele Bicego, Vittorio Murino, and Mário AT Figueiredo. Similarity-based clustering of sequences using hidden markov models. In *Machine Learning and Data Mining in Pattern Recognition*, pages 86–95. Springer, 2003.

[3] Emanuele Coviello, Antoni B Chan, and Gert RG Lanckriet. Clustering hidden markov models with variational HEM. *The Journal of Machine Learning Research*, 15(1):697–747, 2014.

[4] David L Davies and Donald W Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):224–227, 1979.

[5] Michael D Escobar. Estimating normal means with a dirichlet process prior. *Journal of the American Statistical Association*, 89(425):268–277, 1994.

[6] Michael D Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

[7] Héctor D Menéndez, Rafael Vindel, and David Camacho. Combining time series and clustering to extract gamer profile evolution. In *Computational Collective Intelligence. Technologies and Applications*, pages 262–271. Springer, 2014.

[8] Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[9] Lawrence R. Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[10] Gideon Schwarz et al. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[11] Dawn B. Woodard and Moises Goldszmidt. Online model-based clustering for crisis identification in distributed computing. *Journal of the American Statistical Association*, 106(493):49–60, 2012.