Clustering Gamers' Time Series Data : A Case Study

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

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

Moises Goldszmidt[*]
Affiliation, Address
moises.goldszmidt@gmail.com

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

Abstract

The game industry presents a very interesting source of challenges to machine learning methods and algorithms. Games continuosly collect data about the basic actions of players, and from that raw data, companies would like to extract actionable information to both delight players with better game experiences, and improve business metrics. In this paper we explore an approach to address a key enabler to understand players behavior. Given the players time series with raw data about their actions, (1) what are the latent states that govern that behavior, and (2) what are the natural clusters of players given the latent state dynamics. Out approach is based on using Hidden Markov Models (HMMs) to identify the basic players latent states, and then use a clustering method based on Dirchlett Process to group the players according to the time series of these 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, is the declared interest of companies like Amazon (purchased Twitch), Sonny (playsattion, Microsoft (XBOX), among others. Games are also an incredibly rich source of data about human behavior ranging from social interactions to economical and rational decision making.

From the points of view of improving the players' experience and also from the business aspects of retention, engagment and payments, it is very important for game companies to understand player's 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 behavior, the proper analysis of these patterns should be done by looking at the time series of the players actions. These time series are multivariate in nature, as players can take over hundreds of actions in some of these games, and these actions are further parameterized by real values.

In this paper we explore an approach to characterizing players behavior from the multivariate time series of their raw actions in the game based on 1) inducing a time series of latent states that abstract their 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 playing states as the hidden nodes and the players' 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

[*]This work was done while Yuanli Pei and Moises Goldszmidt were at Zynga Inc.

groupings of players using an unsupervised clustering method based on modeling this grouping as a Bayesian version of a Dirichlet Process [].

We test this approach on a mobile strategy game developed by Zynga Inc. Our method found three basic latent states generating all the raw actions, and the clustering results reveal three interesting groups that contains players with similar patterns of transition states within each group. As we will argue these results greatly simplify the task of understanding players' behaviors to improve the game design and other business metrics as monetization and engagement. This characterization enables the game designers and managers to track and identify on a weekly basis the changes in behavior and how it correlates to changes in design, reaction to incentives, and A/B testing.

2 Related Work

There exists a few work on clustering time series data with HMM model, such as [1, 2, 3]. Most of the work focus on finding out the final clustering result, while we are also interested in studying the underlying states of each player at each time period. To our knowledge, we are not aware of any work on clustering time series data in the game domain. The most related work in the game domain is [7], where gamers' time series data are used to extract the user profile. However, the task considered there is different with our paper.

3 Method

Let $X_{it} = [X_{it1}, \cdots, X_{itD}]^\top$ 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. its time series of actions in the game, is denoted as $X_i = [X_{i1}, \cdots, X_{iT}]^\top$. The time series of data for N players, is denoted by $X = \{X_1, \cdots, X_N\}$. 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.

We are now ready to introduce the HMM induction and the clustering process in the next two subsections.

3.1 Uncovering Latent Players States Using HMMs

We assume that there is a latent playing state that controls user's behavior at each time period, and that furthermore the state evolves as the player changes his strategy over time. 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 gamer's latent state at time t, and Y_{it} will be regarded as discrete taking on S values $\{1, \cdots, S\}$.

The mechanism of the players HMM model is as below: 1) initially, a player starts with state $Y_{i1} \in \{1, \cdots, S\}$ according to an initial distribution π, with π_s being the probability of starting at state s; 2) all the Y_{it}'s evolves 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 t; 3) at each time t, the observations X_{it} only depends 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 can estimate the parameters by maximizing the likelihood of the observations

$$\max_{\pi, A, B} P(X|\pi, A, B) = \prod_{i=1}^{N} P(Y_{i1}|\pi) P(X_{i1}|Y_{i1}, B) \prod_{t=2}^{T} P(Y_{it}|Y_{it-1}, A) P(X_{it}|Y_{it}, B) \; .$$

As usual the parameter S is unknown. In this work we use the BIC criteria to fit it.

After estimating the parameters, the state sequence $Y_{i1}, \cdots, Y_{iT}$ for each user can be found using Viterbi decoding [9] which maximizes $P(Y_{i1}, \cdots, Y_{iT}|X, \pi, A, B)$.

3.2 Clustering Players Behavior Based on State Transitions

The next step is to clustering the players based on the time series of state transitions. In our approach we adapt the method proposed in [10], as they also look 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 the clusters evolve according to a Dirichlet distribution with parameter α.

Let $Y = [Y_1, \cdots, Y_N]^\top$ be the state transitions for all the players, where $Y_i = [Y_{i1}, \cdots, Y_{iT}]^\top$ denotes the i-th player's states from 1 to the T-th time. As it is common in this approach we us Z_i as an auxiliary variable denoting the cluster assignment for the i-th player. We use K be 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 vector for the initial state distribution, and Φ^k is the $S \times S$ transition matrix. We use the prior distribution for parameters in each cluster is $G_0(\{\lambda^k, \Phi^k\}) = Dir(\hat{\pi}) \prod_{s=1}^{S} Dir(\hat{B}_{s.})$, where $\hat{\pi}$ and $\hat{B}$ are the estimated parameters at the first step. 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 likelihood of the data of state transitions for all players is

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

where $\mathbf{1}[\cdot]$ is the indicator function, and n_{irs} is the 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 [10] 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 Dirichlet distribution can be equivalently done as below [8]: set $Z_1 = 1$; for subsequent players, sample Z_i according to the following distribution

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

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

4 Experiments

We apply our method to Zynga Inc's strategy game called "Empires and Allies". In the game, the goal is to conquer all the battlefields in a global map, either with machine or with other players. To conquer the battles, the players need to build/upgrade base resources such as weapons and troops,

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
Our	**3**	**0.968**
Kmeans	5	2.628
GMM	4	1.803

which in turn requires game points that can be obtained from winning battles. Thus, the players need to tradeoff between building resources and conquering battlefields.

We form a dataset that contains player's 10 consecutive days data after installation, and subsample users that are active at all the 10 days. This results in 1719 players. Each player is measured with 67 metrics at each day, and we select 5 important features based on prior experience: `PvP` (people vs people battle), `Pve` (people vs machine battle), `Points` (number of points gained), `Session` (number of session started), `LevelUp` (whether a player level up) and `isPayer` (whether the player paid). We model `Points` with Gaussian distribution, `Session` with Poisson distribution, and the rest features with Bernoulli distribution.

4.1 Identifying Players' Daily States

We fit HMM model with different number of playing states and found that 3 states is reasonable. Table 1 lists the discovered playing states, **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 build the resources. 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 improve.

We then decode player's states at each day. Figure 1 plots the transitions for all the users among the 10 days. 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 also consistent with the game design since the players can not start with many battles at the beginning due to the resources restriction, but they ultimately need to conquer all the battlefields.

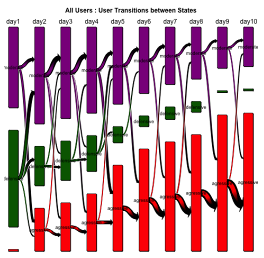

Figure 1: State transitions of all the Players at the first 10 days of installing the game.

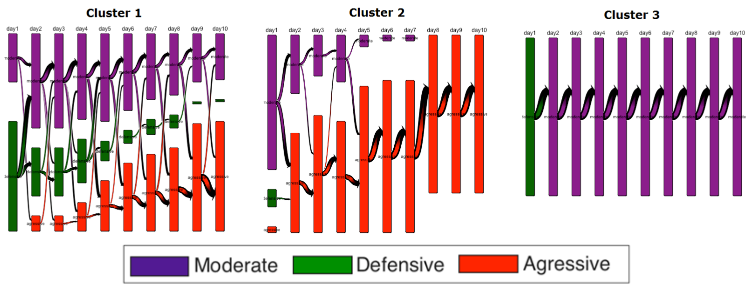

Figure 2: State transitions within three clusters found by our method.

4.2 Clustering Results

Next we clustering users based on their state transitions with the method explained in Sec. 3.2 . We compare with two baselines: Kmeans and Gaussian Mixture Model (GMM). Due to the fact that we do not have actual cluster labels, we evaluate the results using a polular internal evaluation method Davis-Bouldin (DB) index [4]. 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 report the clustering results of all methods, where our method outperforms the baselines. 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 at Figure 2.

5 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 sates 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 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 state 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 Dirichilet process approach also provide means for 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] 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.