Capstone Proposal

Aaron Dharna

June 18th-23nd, 2017

Proposal

Domain Background

According to Jon Weisnewski, a senior gameplay designer at Bungie, the twin pillars of Destiny are "synchronicity" and "Player Fantasy." The former meaning that one's guardian (player character) should feel the same no matter whether you play in single or multiplayer modes. And the latter meaning that when and how you play should evoke or fulfill certain feelings. That's reflected in everything from the weapon design (i.e. The Last Word) to the raid -- 6-person multi-hour team-activities -- design (i.e. The Vault of Glass). It's all built to impart feelings of empowerment onto the player. Something almost as fundamental to Destiny's design is the idea of the team. And this is reflected in everything from the Player versus Environment (PVE) game design to the Player versus Player (PVP) game design in that in order to *even participate* in the end game content, a pre-formed team is required on the part of the players.

Therefore, the question that I'm looking at is "can we build a team of players whose play-styles complement each other when given player-clusters of play-styles"? Or how do you build a team of people who will empower each other's player fantasies? Can you pull players from across multiple clusters to form a single well-balanced team? In short, how can you use clustering methods to enhance matchmaking?

To optimally understand this paper, knowledge of clustering algorithms and neural networks is required. Both topics will be briefly touched upon, but only as a refresher. Using the methods above, this research will go into matchmaking (and therefore match-prediction), uniting both things under the umbrella of gaming and player data.

Problem Statement

When I started thinking about this project, I was thinking about League of Legends and "How can one build a good team in the matchmaking process using player data". The lynchpin in that question is "what is the definition of 'good'?" My goal for this project is to use player data and machine learning to promote individual learning and lasting group creation through the use of clustering algorithms to enhance matchmaking and promote inherent design principles within the game. An application of such a system would be to open up complex game modes -- ones that require multiple skill sets to deftly complete -- to the average (and often alone) player. As such, the problem to be solved has multiple parts:

1) Feature creation:

    a) Reduce dimensionality of match statistics into fewer stats.[1]
    b) Using player history data, create behavioral/play-style clusters[2].
2) Using an RNN (LSTM?) build up two teams of players.
3) Use an Fully Connected network to predict which team will win.

Datasets and Inputs

Every time a person plays a PvP match, we get information recorded about that player as well as any other players in a match. The data collected are typically Performance data, with again Engagement information such as the duration of the match. There are many different types of PvP matches, but the data tracked is similar and we can treat them as similar for the kind of top-down work we are planning here.

All of my data is accessible through Bungie's public API[3]. In order to access it, you need an API key which can be requested through their website. However, I got my data through an intermediary -- Destiny Trials Report -- since I was having a lot of trouble figuring out how to interact with Bungie's API; Thank you, Vlad! Furthermore, data for the match-making half of the project was supplied by Dr Anders Drachen.

Up first in our data is the complete activity history of 1153 players with respect to PvP: amounting to ~1.3 Million matches. Each player's match history along with some simple stats on that match are stored in that table. For my own sanity, I've broken that table up into 26 CSV batches of 50,000 matches (and 1 of 3000ish) amounting to roughly 44 users per batch. For each match, there are 18 pieces of data recorded, making the full table 1.3m x 18 (or 50000 x 26 x 18) roughly.

Complete PvP dataset

Broken down by the 18 features, you get:

membershipId --	PSN/XBOX/Bungie account id
characterId --	whichever of your three characters is being used
instanceId --	match ID
period --	date/time of match
activityType --	game-mode (clash (6v6 free for all), trials (3v3 elimination), etc)
map --	map
standing --	win (0) or loss (1)
kills --	kill in that match
deaths --	deaths in that match
assists --	assists in that match

[1] I learned about the James-Stein Estimator recently, and I think that might be helpful for determining a "skill" metric that is more accurate than PercentContribution since the JSE will look at that metric over all players and take the given player into consideration.

[2] https://goo.gl/qClnPU This paper here by Dr. Anders Drachen provided the inspiration for this project.
[3] https://www.bungie.net/platform/destiny/help/ Bungie's public facing REST API.

score --	your total score in that match
teamScore --	total team score in that match
private --	whether the match was private
completed --	did you stay till the end
playerCount --	player Count in activity
completionReason --	did you complete the activity, if not: why?
leaveRemainingSeconds --	how much time left when you left
activityDurationSeconds --	how long the activity was

Of these, I'll be cutting out private, period, characterId, completed, completionReason, instanceId, and leaveRemainingSeconds.

membershipId will be changed from a feature into an index.
Kills, deaths, and assists will be recombined into Kills / Death and (Kills+Assists) / death.
Map and activity type will be one-hot-encoded (Although how one plays might be impacted (ie a person who specializes in close range has to compete on a large arena and therefore does worse)these might be removed completely to help reduce dimensionality).
TeamScore will be edited into PercentContribution which equals score / TeamScore

Resultant complete PvP dataset

membershipId --	PSN/XBOX/Bungie (account id)
activityType --	game-mode (clash, trials, etc)
map --	map
standing --	win (0) or loss (1)
Kd --	kills/death
KaD --	(kills + assists) / death
score --	your total score in that match
teamScore --	total team score in that match (will be cut after calculation)
PercentContribution --	score / TeamScore
playerCount --	player Count in activity
activityDurationSeconds --	how long the activity was

If in the first dataset we got the complete PvP history for ~1000 players, the dataset provided by Dr Drachen, is nearly identical; however it has been scaled up to 10,000 randomly selected players. A similar dataset could be generated by pinging Bungie's API, as was done for the first dataset; however, that would take a long time.

With that out of the way, we have the match-making dataset. This is a record of Post-Game Carnage Reports (PGCR). A PGCR is the complete team-and-singular-player-scoped record of a single match in Destiny. This includes: data about two teams -- each made up of either 3 or 6 players -- who competed against each other in the Crucible (the PvP section of Destiny), stats for each player's contribution to the

match, the result of said match, a ranking of how well each player scored (a numeric description of how well they played), and which team won the match.

Solution Statement

As mentioned, this project will have 2 (or 3 depending) parts. 1) clustering 2) Team building/learning.

For the first part, I am going to base it on the paper linked above (see subscript 1). This part is not the focus on my research, but rather a necessary first step. Ideally, I would have a classifier already, and would only need to classify each account to go into step 2. However, assuming that I need to do it, I plan to use a Gaussian Mixture Model as my algorithm to run the classification.

The second, and main, part of my research is to investigate how behavioural clusters impact matchmaking using sequences of players and an Long Short Term Memory network (LSTM). To do this, I take each PGCR and run each member of both teams through my clustering from part 1. Then, recombine the teams. Each member will have a cluster as well as their match statistics describing how well they played. Since a team can be looked at a sequence of players, the first network -- which is focusing on team building -- will train on teams of players learning what makes up a team composition. In this case, the network will attempt to predict the classification of the next player in the sequence. Then these team composition can be fed into the second network. The second network will take in a PGCR and then output the probability that team A won/lost (P(B won) = 1 - P(A won) since this a zero-sum game). Now, this might not be possible, but my thoughts here are that I should be able to "link" up the two networks. If the LSTM trains on "teams" (which are made up of players) and the vanilla NN trains on matches (which are made up of teams), then I should be able to compose the two networks and train them simultaneously by backpropagating a single gradient.[4]

Benchmark Model

I do not know of any benchmark models to compare this research against. What I could do is build a nearly identical network as above, train and test on the same data; however, do not include the cluster-classification from part 1. This would give me something to empirically compare the above model against. That being said, if you -- my reviewer -- happen to know anyone doing similar research and could offer a suggestion, I'd be more than happy to hear that.

Evaluation Metrics

Given that this problem boils down to a supervised learning problem, the metric that I think is most fitting is accuracy. I have the results that form the basis of what my neural networks will be learning (first, teams. And then, whether a team wins a match). This is what will allow me to backpropagate the gradient through my networks, but it also means that I can compare the generated result against the true result.

[4] https://www.youtube.com/watch?v=yk6XDFm3J2c an example of wiring two different architectures together

Like the above section, if you can recommend a better metric or a place where I can do some readings to put together a better metric, that'd be lovely.

Project Design

So, I've kind of done this throughout the proposal; however, I shall go through the entire process again here.

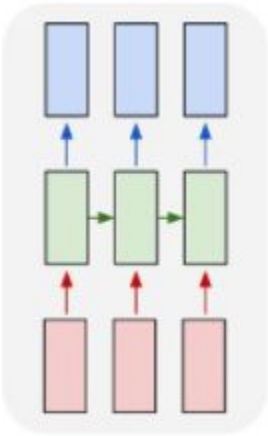
1) Preprocess both datasets according to above (player history data and PGCRs).
2) Run the clustering analysis on the player-history data generating play-styles.
3) Classify all players in the PGCR into a playstyle cluster.
4) Feed individual teams into the LSTM to learn team composition.
5) Feed both sets of teams into the vanilla network to predict a winner.
6) Backpropagate the Loss from 5 into 4 and 5, editing the weights.
7) If necessary, split the networks into two separate networks.

Strategies to employ:

One of the things I'm going to have to overcome during the preprocessing stage is making sure that both sets of data contain the same/similar information since I'm going to be using the first dataset as ground zero for my classifying of players in part two.

As a whole, This problem seems like one ideally set up for Neural Networks. Given that i have a sequence of players, a recurrent Neural Network seems like the most logical choice. Furthermore, given how an LSTM solves (or at least mitigates) the vanishing gradient problem inherent in RNNs, it again makes sense to use that formulation of the network. The LSTM, if all goes as planned, will be generating sequences of players (teams) and since teams get fed into the victory-Network, it makes sense that these networks should be composed (the output of the LSTM is ½ of the input of the victory-Network). If I cannot compose the two networks, I'll just separate the two ANNs. This would still make sense because when training the LSTM, for each team member, the next team member's playstyle-class becomes the target variable from which we can compute the Loss function. As for the vanilla network, the training would be computed the same as before since we still only care about the projected outcome the match, given the PGCR.

In order to keep the complexity down, I am going to use Tensorflow and Sci-kit Learn as my main tools for this project. SkLearn helping out with the pre-procesing and clustering, and Tensorflow for my neural networks. The structure that I'm thinking of for the LSTM -- assuming that I am working with 3 man teams in the PGCRs -- is roughly:



In this format, we first feed in team-member-a and receive a softmax over the possible clusters that the next teammate might belong to. Ideally, the network will learn to output team-member-b's playstyle-cluster. Then we feed in team-member-b and receive team-member-c's cluster. After feeding in team member c, we receive an end token signaling that the team is built since this will be trained on 3-person teams. (I am well aware that this will need to be iterated on. So any advice here would be wonderful.) What this will hopefully allow is for the network to take in a single seed player and build out a full team emulating successful teams it has seen in the past (assuming I can wire the two networks together. If I cannot wire the two networks together, this first network will simply learn team composition based on those it has seen without regard for how well that team fared.)

One thing I would like some advice specifically on is: Given the structure for my datasets/players above, what would be a good way to represent my player and team data to feed into the neural networks?