Predicting Future States in DotA 2 using Value-split Models of Time Series Attribute Data

Zach Cleghern
North Carolina State University
Raleigh, NC 27695
zbcleghe@ncsu.edu

Soumendra Lahiri
North Carolina State University
Raleigh, NC 27695
snlahiri@ncsu.edu

Osman Özaltın
North Carolina State University
Raleigh, NC 27695
oyozalti@ncsu.edu

David L. Roberts
North Carolina State University
Raleigh, NC 27695
robertsd.csc.ncsu.edu

ABSTRACT

In Multiplayer Online Battle Arena (MOBA) games, teams of players compete in combat to complete an objective and defeat the opposing team. To stay alive, players must closely monitor their character's status, especially remaining health. Understanding how health may change in the near future can be vital in determining what tactics a player may use. We analyzed replay logs of the game Defense of the Ancients 2 (DotA 2) to discover methods to predict how players' health evolves over time. For DotA 2, our results suggest that forecasting changes in a player's health can be done by viewing gameplay as two separate processes: normal gameplay flow in which changes in health are smaller and more regular, and less frequent but higher-impact events in which players experience larger changes in their health, such as team battles. We accomplished this by considering health data as two separate, but interleaved, time series in which separate processes govern low magnitude changes in health from high magnitude changes. In this paper, we present a value-split approach to predicting changes in health and describe the results of our approach using autoregressive moving-average models for low magnitude health changes and a combination of statistical models for the larger changes.

CCS CONCEPTS

•**Mathematics of computing** →*Time series analysis;* •**Applied computing** →*Computer games;* •**Software and its engineering** →*Interactive games;*

KEYWORDS

Multiplayer Online Battle Arena (MOBA) Games, Game Analytics, Time Series Analysis, Prediction

ACM Reference format:
Zach Cleghern, Soumendra Lahiri, Osman Özaltın, and David L. Roberts. 2017. Predicting Future States in DotA 2 using Value-split Models of Time Series Attribute Data. In *Proceedings of FDG'17, Hyannis, MA, USA, August 14-17, 2017,* 10 pages.
DOI: 10.1145/3102071.3102095

FDG'17, Hyannis, MA, USA

DOI: 10.1145/3102071.3102095

1 INTRODUCTION

Multiplayer Online Battle Arena (MOBA) games are a growing genre in competitive gaming. Typically, MOBA games pit opposing teams in combat as they work to achieve some ultimate goal. Matches are dynamic as players must continually monitor and update their strategy and careful team coordination is required in order to win. Thus MOBAs are an interesting domain for analyzing player behavior and interaction and also for investigating how to forecast events from temporal data. Defense of the Ancients 2 (DotA 2) is a highly competitive MOBA game in which two teams of five players each take control of one of several distinct heroes, each of which has unique abilities, strengths and weaknesses. The teams battle in real time and players must work together to destroy the enemy team's Ancient, a building located in the enemy's base and surrounded by defenses. Players face evolving dangers as they must both battle computer-controlled "creeps" and the heroes controlled by the opposing team. DotA 2 has properties that are typical of MOBAs, such as competitive team-based combat, predefined heroes, skill progression, and non-player creeps that can be killed for resources.

MOBA games, and DotA 2 specifically, typically produce detailed replay logs from each match that provide a massive, rich set of time-ordered data that presents an opportunity for data-driven analysis of player behavior and to explore methods for forecasting events within a match. The problem we focused on is as follows: given the replay log up until a given time step, how can we predict changes in a hero's health attribute in the near future? The hero dies when their health drops to zero, so it is important for players to closely monitor their health. This is an important event in a match because hero deaths can turn the tide against the opposing team who must temporarily play with fewer players. Forecasting health is an especially hard problem because future changes to a hero's health are not a product of a single player's behavior and can be affected by complex interactions, such as battles between opposing teams and interactions with other teammates. A model for prediction of future states in DotA 2 would provide insight into the flow of the match and could have use both in augmenting human commentary by giving a hint to possible outcomes and

in improving AI game playing. Creating artificial agents to play MOBAs and multiplayer games in general is a difficult task because at any given time outcomes in the game are affected by the actions of many interacting human and computer-controlled characters who may be cooperating, in conflict, or in some cases neutral. Forecasting future events could have some benefit for AI agents since analyzing game state is such a complex task and imminent danger may not be obvious.

We hypothesize that there is predictive value in viewing changes in a hero's health as the result of two evolving processes at work: routine gameplay events and higher-impact events. Two examples of routine gameplay events are heroes defeating computer-controlled "creeps" to earn experience and money, taking damage while doing so, and heroes regenerating a small amount of health when they are not at maximum health. Examples of higher-impact events includes team battles, destroying important buildings such as Towers, or battling the powerful computer-controlled character Roshan. These higher-impact events can have a large effect on the overall match and are not easily captured by the same models for the smaller-scale, more regular gameplay.

The question of how to separate these processes is a difficult one because it is only an interpretation of gameplay and not explicitly represented in a replay log. We note that the hero health attribute as a time series tends to have statistically significant autocorrelation, or lagged correlation with itself. This suggests that prior changes to health may be predictive of future changes. We chose to focus on the health time series to analyze different types of changes. Our method captures this separation by splitting the time series of a hero's health into large magnitude and small magnitude changes. In particular, we use an autoregressive moving-average model, a standard time series analysis tool, to model the more frequent, low magnitude changes in health, while the less frequent, high magnitude changes are modeled using several statistical models: a nonhomogeneous Poisson process to estimate the number of high value changes in a given time interval, a logistic regression model to predict the sign (positive or negative) of the high values, and linear regression models to predict the magnitude of positive and negative health changes separately.

We describe a novel approach to viewing time-ordered gameplay logs and present the results using several statistical models to predict low and high magnitude health changes, both individually and also combined to forecast overall changes in health. Our results suggest that separating health time series data into low and high magnitude changes is a promising approach to forecasting future health changes and indicates potential as a more general approach to forecasting other attributes as well.

2 RELATED WORK

MOBA is a relatively new but growing genre in gaming, so methods for analyzing and predicting players' behavior in MOBAs is largely unexplored. However, some work has been done in the area of analyzing gameplay in DotA 2 and similar games and in prediction of both win/loss outcomes and gameplay events. Erickson and Buro identified an algorithm to extract individual battles from replay logs [5] in Starcraft, a real-time strategy game. Drachen *et al.* [4] analyzed the relationships between differences in DotA 2 player skill and spatio-temporal metrics such as how often players are moving around the map in the game. They analyzed distance between team members and number of zone changes and did cluster analysis across different player skill tiers. While Drachen *et al.* analyzed temporal data like we did, their work did not specifically focus on prediction.

Prediction of outcomes in MOBA and similar games has been done. Erickson and Buro, in addition to analyzing Starcraft battles, predicted who would win and lose Starcraft matches based on the current game state [5]. Pobiedna *et al.* [7] analyzed role distribution, experience, number of friends, and national diversity to identify factors that are predictive of team success in DotA 2. However, this did not utilize gameplay events within a match to make predictions. Rioult *et al.* predicted wins and losses based on topological information derived from player locations [8]. Examples of such topological information included the area of the polygon described by player positions, the mean of the distances to the barycenter of the polygon, which the authors referred to as gathering ability, and the inertia of the team, which the authors defined as the standard deviations of the distances of players to the barycenter of the polygon. Rioult *et al.* used standard machine learning algorithms such as decision trees, neural networks, and support vector machines to make win/loss predictions from these and other similar metrics. They achieved 85% precision and 90% recall rates using this approach. Yang, Harrison, and Roberts [10] also tackled the goal of predicting team success, but used graph mining on patterns of combat within the match itself to make predictions. This approach used information from within a match itself to determine the most likely outcome of the entire match, but did not focus on forecasting events within the game.

Prediction of smaller scale events in competitive games such as Starcraft has been done with success. Synnaeve and Bessiére used a Bayesian model to predict what strategies Starcraft players would employ in the beginning of a game [9]. Dereszynski *et al.* predicted player build strategies based on previous observations using hidden Markov models [3]. They were able to successfully predict that a player would build a certain type of unit based on previous observations with a true positive rate of 0.725 and a false positive rate of 0.272. The hidden Markov model created to make these predictions did not encode any expert knowledge about the nature of the game or its strategies and was learned from game data alone. This is similar to how we view health changes as time series data and do not generally incorporate expert DotA 2 knowledge to assist predictions.

3 METHODS

In this section we describe the different statistical modeling methods that comprise our forecasting approach.

3.1 Data Set and Preprocessing

Our DotA 2 data set consists of 542 matches. Each match replay consists of two files:

- Stats: The stats file contains statistics for each player in the game at every time step recorded. For each player, there are 48 recorded attributes, such as their chosen hero, health, level, strength, agility, and items held.

- Steps: The steps file consists of actions each player took at each time step. Additional information is recorded for actions such as "move", which logs the coordinates at which the player began the move and the destination of the move. Examples of other actions logged include leveling up, buying items, and damaging other players.

The length of an average match is 40 minutes and the data is quite granular at a resolution of 30 frames per second; each snapshot of game data is recorded approximately 15 frames after the previous one. We refer to each such snapshot as a timestep. This data set is fine-grained to the extent that the match can be fully reconstructed from the Stats and Steps files.

The focus of this work is on a hero's current health and how it may change in the immediate future. At a given time step, a hero's health is a numeric value that represents how much life that character has remaining. When this reaches zero, the hero dies and must wait a few seconds to respawn. Respawn time increases with the level attribute and is calculated as $1 + 4 * (HeroLevel)$. Upon respawn, the player can continue playing with their hero at their current maximum health. Their maximum health grows throughout the game as they gain levels and become stronger. Anytime a hero's health is not at maximum, they regenerate health at a small rate at each time step. This rate, called regen rate, is small and its magnitude depends on the hero's base rate which varies by hero, and their strength attribute which changes over time. Heroes experience a much faster health regeneration when they are near one of the two Fountains, important buildings for teams to recuperate. Other abilities and items may also influence the rate of health regeneration.

A hero's changes in health over time can be observed by differencing the time series (*i.e.*, replacing each value at time t with the difference between the values at t and $t-1$). For instance, a health value of 400 immediately preceded by a value of 450 would be replaced with -50. The health difference time series was then split into two separate time series: one containing only the small magnitude values (absolute value less than 100) and one with only the large values. The threshold between small and large changes was set as 100. One reason for this threshold can be seen in Figure 1, which shows health changes for a typical DotA 2 match. While changes smaller than 100 are somewhat evenly dispersed throughout the duration of the match, large changes are not common at the beginning, but become more likely as the match proceeds and players are more likely to directly engage with enemy players and their defenses. Splitting at 100 also significantly alters the autocorrelation, or lagged correlation with itself, of the differences in health. This is described in more detail in Section 3.2.1. Since splitting a time series leaves missing data, we used these null values as placeholders so that the number of timesteps in between the remaining values was preserved.

When evaluating our methods, we divided the 542 match data set into three sets. For a training set, we used 80% or 433 of the matches. The remaining 20% was divided into two test sets. The first, with 50 matches, was used as validation for experimentation with the parameters of the high value models and to evaluate their performance in isolation. We refer to this as the validation set. The second can be considered a more traditional test set and consisted of the remaining 59 matches. This was used to evaluate the combined low and high value approach with parameters tuned using the validation set. We used this approach so that when evaluating the combined model, we were able to test its performance on data that no part of the model had seen previously.

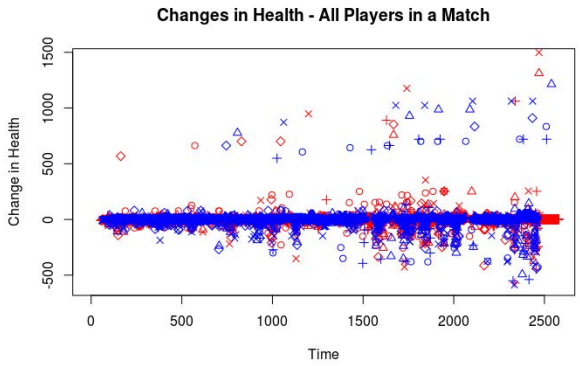

Figure 1: Changes in health for all heroes in one match. Note that while small changes are evenly dispersed, lage changes become more common as the match persists.

3.2 ARMA Model for Low Values

Here we describe the autoregressive model we implemented for low values.

3.2.1 ARMA. In DotA 2, a time series of changes in health, as shown in Figure 2, tends to have some level of autocorrelation as in the example of the hero's autocorrelation plot in Figure 3. For example, a lagged correlation 1 lag ahead means that values at time t+1 tend to correlate with values at time t. Including only small value changes, the ACF changes significantly as show in Figure 4. In Figure 3, there is statistically significant autocorrelation at lags 1,2,3, 5 and 6. However, for small changes only , the ACF is much higher and extends to further lags, which shows that an ARMA [6] model may be an appropriate fit for the small changes in health because there are several lags with large autocorrelation. For basic ARMA modeling, we have two parameters: the AR (autoregressive) term and MA (moving-average) term. These parameters are often referred to as p and q. Higher values of p and q will include more past information in the prediction, but also increase the complexity of the model. In the example autocorrelation function plot shown in Figure 3, we would not want to choose a value of p larger than 6 due to 6 being the largest lag with significant autocorrelation.

3.2.2 ARMA Fit. A more generalized form of ARMA is ARIMA, or autoregressive integrated moving-average model, which includes a differencing term. Our time series was differenced manually so that we could separate low and high differences. This means that we can ignore the differencing term and use the simpler ARMA model instead of an ARIMA model. The order of the AR term, or value of the parameter p, was calculated by examining significant lags in the autocorrelation function of a given time series and the order of the MA term was calculated similarly by examining its partial

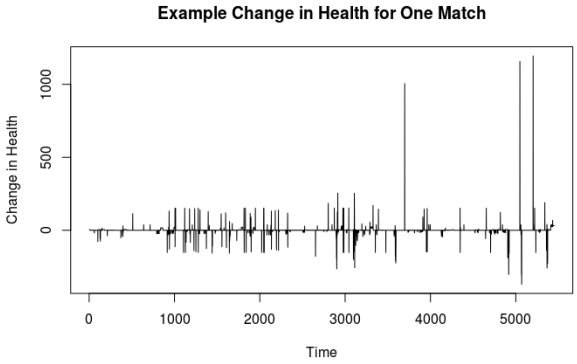

Figure 2: Example change in health for a hero in one match.

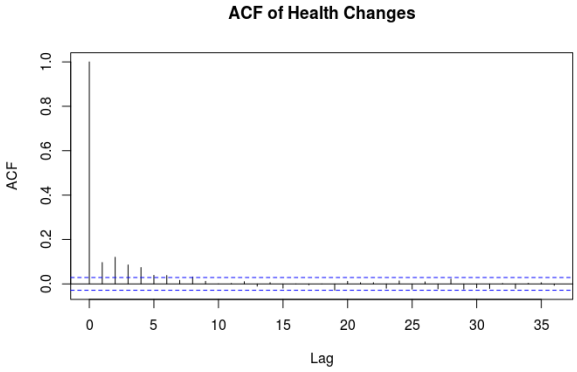

Figure 3: Autocorrelation of example hero health changes for one match. Significant lags end at lag 6.

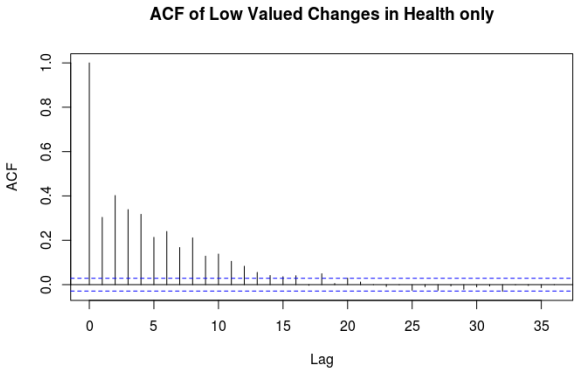

Figure 4: Autocorrelation of example hero small (less than 100) health changes for one match. Autoorrelation is higher now that large changes are removed.

autocorrelation function. The partial autocorrelation function of a given lag k is the correlation between values k time steps apart, but unlike the autocorrelation function also controls for correlation between values $1...k$ time steps apart [1].

For each time point used to test health prediction, the autocorrelation function and partial autocorrelation of the time series from the beginning of the match up until the current moment were computed. The value for the p parameter for ARMA was selected as the largest lag that was statistically significant, with a maximum of 10. The value for the q parameter was computed in the same manner, using the partial autocorrelation function instead. Only including the largest significant lag prevented over complicating the model and thus overfitting the data set. Overfitting can cause prediction errors on previously unseen data.

3.2.3 ARMA Prediction Results. For evaluation of the ARMA model to forecast low value changes in health, we did not need to divide matches into training and test sets since an ARMA model uses only previous data in a time series to make predictions. This means that to predict future health changes for a hero in a given match, we only need the hero's previous health changes from the current match. In light of this, we randomly selected 10 matches from the training set to evaluate the performance of ARMA models. For each player in the game log, we divided the match into equally sized contiguous segments. We started evaluation at 1000 time steps into a match. The reason for this was to allow the ARMA model sufficient data to make sensible predictions and because changes in health are not as interesting at the beginning of a match when players are just getting started and not heavily interacting. Every 100 time steps, we trained a new model because typically ARMA models are created at once with the entire training data and must be computed again to include new data. We computed each model using the entirety of the time series up until the end of the current segment, and from this test point attempted to predict the value of the change in health over the next several timesteps. We used the arima function in the R stats package to accomplish this. Once again, this is still an ARMA model as opposed to ARIMA because only the AR and MA parameters (p and q) were nonzero.

Once the model was computed, we used it to predict the changes in health over the next 40 time steps (approximately 20 seconds of gameplay) in the time series. For each value of n from 1 to 40, we then calculated the net result of the individual predictions from 1 to n to arrive at a forecast of the total change in the hero's health (ignoring large value changes for now) over that particular span of time. This is referred to as an n-ahead forecast. We calculated error as the difference between the sum of the health changes that the ARMA model predicted and the sum of actual changes that took place in the time series. Note that for this low magnitude approach, since the model only deals with small magnitude values, neither the sum of the predictions nor the sum of actual changes reflect the actual change in health over that time interval. Instead, these sums represent what the change in health would be if all large magnitude changes were removed. This allows us to test the model only on portion of the data for which it is intended. For 20-ahead forecasting, the distribution of errors can be seen in Figure 5. At 20 time steps ahead, most errors are concentrated below 100, but

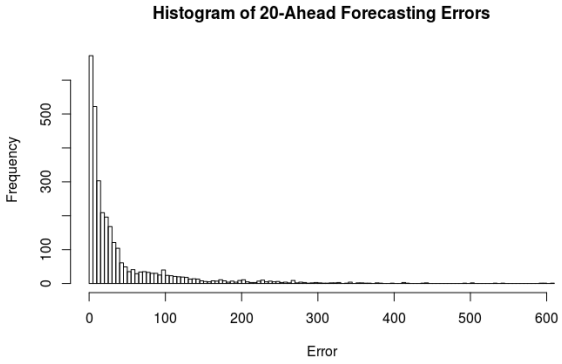

Figure 5: ARMA error distribution for 20-ahead forecasts.

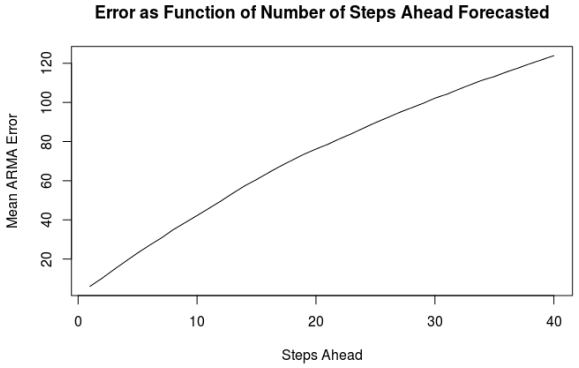

Figure 6: ARMA Mean n-ahead prediction error for n = 1..40.

occasionally the model makes very large errors (the maximum error for this distribution is approximately 600).

Figure 6 shows the mean error at every time step ahead for which the model forecasted change in health. As one may expect, forecasting error tends to increase as the forecast becomes further in the future. It is much more difficult to predict how a hero's health will change over 40 timesteps than it will change over the span of only 10 timesteps.

The ARMA prediction error was also compared with a baseline model. The baseline we used predicts each future time step as the mean of the values seen so far in the current hero's health time series, again only considering low values. The individual predictions were summed to create a forecast as we did with the ARMA model. For each time step ahead, we compared the difference between the mean ARMA error and the mean baseline error. This result is shown in Figure 7. We can see that in the range of 8 to 12 the mean error of the ARMA forecast is at its smallest relative to the corresponding mean error of the baseline. In fact, for 10-ahead forecasts, the ARMA model predicted the change in health within 50 health points of the actual change (the sum of actual low value changes) 77.2% of the time, within 100 points 87.9% of the time, and within 200 points 95.6% of the time.

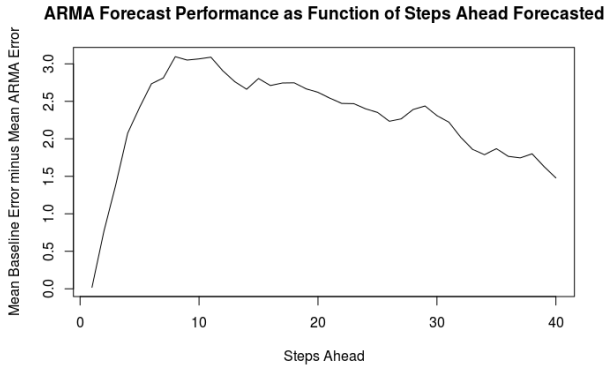

Figure 7: Mean of the net baseline errors minus ARMA errors at each time step ahead. The ARMA model's performance peaked 8 to 12 time steps ahead.

3.3 Marked Poisson Process For High Values

Here we describe the Poisson process model we developed for forecasting high values.

3.3.1 Marked Nonhomogeneous Poisson Process. Modeling large changes in health is quite a different task since large changes (defined as magnitude 100 or greater) are less frequent and are spread out sporadically across the duration of a match. This can be seen in Figure 2. To model when these large changes, or jumps, appear we used a marked nonhomogeneous Poisson point process model. Poisson point processes are a common way of modeling points distributed across some space and the process being nonhomogeneous means that the points are distributed with varying intensity across the space. Here, the space of the point process refers to time. Referring to the process as "marked" means that point in the Poisson point process is associated with some value which here designates the magnitude of the health change at that jump point. Consider the nonhomogeneous Poisson Process $N(I)$ such that $N(I)$ denotes the number of jumps in the time interval $I = [t_{begin}, t_{end}]$ and $\lambda(t)$ denotes the intensity function of the process, we can use the equation

$$P(N(I) = k) = e^{-\Delta(t)} \frac{\Delta(t)^k}{k!}, k = 0, 1, \ldots, \quad (1)$$

where $\Delta(t) = \int_{t_{begin}}^{t_{end}} \lambda(s)ds, t \leq 0$ can be used to forecast how many jump points a specified time interval will contain. To estimate the intensity function for a hero's health time series across an entire match, we used the equation

$$\lambda(t) = \frac{1}{nh} \sum_{i=1}^{n} k(\frac{t - t_i}{h}), t \in [0, T], \quad (2)$$

where

$$k(x) = (2\pi)^{-1/2} e^{(\frac{-x^2}{2})}, x \in \mathbb{R}, \quad (3)$$

and $h = cn^{-1/5}, c \in (1/4, 4)$. However, this model only produces an estimate for the number of jumps over an interval of time. It does not predict the magnitude or sign of these jumps, which is vital to forecasting health as it could mean the difference between a hero's

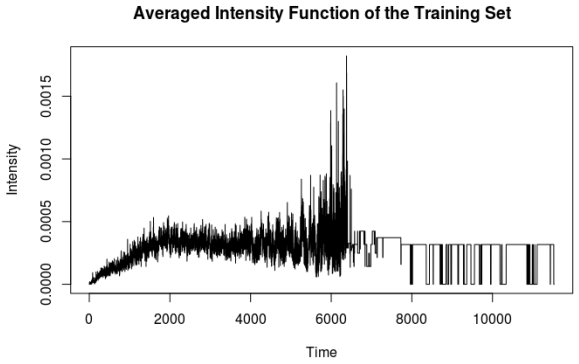

Figure 8: Averaged Intensity Function representing nonhomogeneous Poisson process of large jumps in health.

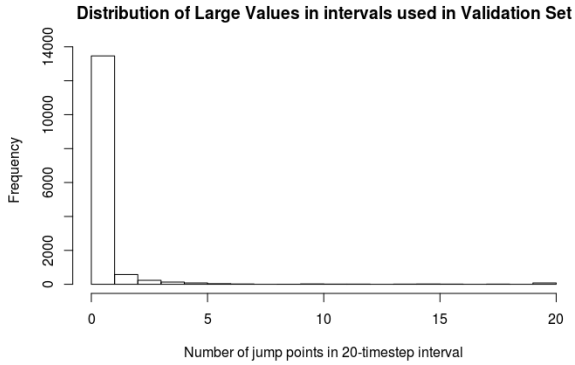

Figure 9: Histogram showing the distribution of high value changes ("jump points") in the time intervals used to test the Poisson process model.

life or death. For prediction of the magnitude of change, we further separate the task into the problem of predicting the sign of the next jump in health and predicting its magnitude.

3.3.2 Poisson Process Results. We calculated the intensity function for the jump points of each player in the training set. This was then averaged, resulting in a representation of the average intensity of jump points. This is shown in Figure 8 as a plot of averaged intensity over time. After roughly $t = 6000$ the plot takes on a much different form due to the relative absence of matches in the training set that reach a duration spanning that many timesteps. This model of intensity can be used to estimate the number of jump points over any given time interval as in Equation 1.

The values of the averaged intensity function were too low for estimation because in any individual match, jump points are relatively sparse. Thus the intensity function of an individual time series had low valued and infrequent spikes. Once the average intensity function was computed from the training set we multiplied the intensity function by a constant scalar to allow for estimation of jump points. We found that multiplying the intensity function by 200 allowed the equation for number of jump points to produce estimates other than zero and thus used this constant for estimation. We estimated the number of jump points at regular intervals in the validation set using the same process as we used for ARMA to divide the time series into a number of test intervals. Each interval in the validation set consisted of 20 time points. For the validation set we used, the distribution of the number of jump points in each interval is shown in Figure 9. Note that while the frequency sharply declines as the number of jump points increases, there is a large increase for 20 jump points, which is the maximum possible in a 20-timestep interval. The error at a time interval was defined as the difference between the number of jump points in the interval and the number of jump points estimated by the model. This distribution is shown in Figure 10. The distribution of errors closely resembles the distribution of actual numbers of jump points, indicating that this model did not perform well when there were many jump points in the interval.

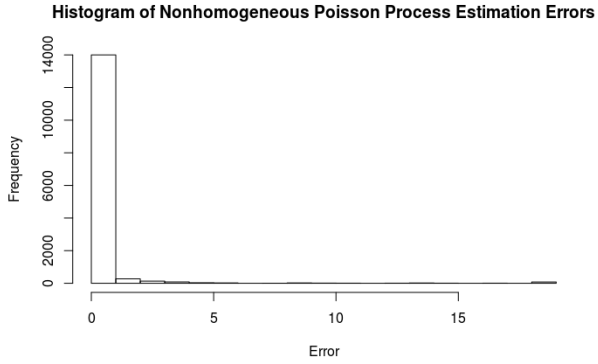

Figure 10: Histogram showing the distribution of errors produced by the Poisson point process model for jump points.

3.3.3 Logistic Regression for Signs of High Values. To predict the sign of the large values in the time series, we used a logistic regression model [2] to evaluate the probability of the sign being positive given the signs of the previous K large values. This was chosen due to patterns that appear to be present in the plot of time series of health changes; it was noted that large positive changes (healing) tends to occur after sequences of large amounts of damage taken and that this could ideally be captured by a simple linear model. Logistic regression is defined in terms of the logistic function shown as Equation 4 and its inverse, the logit function, shown as Equation 5.

$$f(S_{i-1} + \ldots + S_{i-K}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 S_{i-1} + \ldots + \beta_1 S_{i-K})}} \tag{4}$$

$$\ln \frac{p_i}{1 - p_i} = \beta_0 + \beta_1 S_{i-1} + \ldots + \beta_1 S_{i-K} \tag{5}$$

In Equation 4 and Equation 5, p_i is the probability of the ith large value being positive, $\beta_0, \ldots, \beta_K$ are the logistic regression coefficients to be optimized, and $S_{i-1}, \ldots, S_{i-K}$ are the signs of the

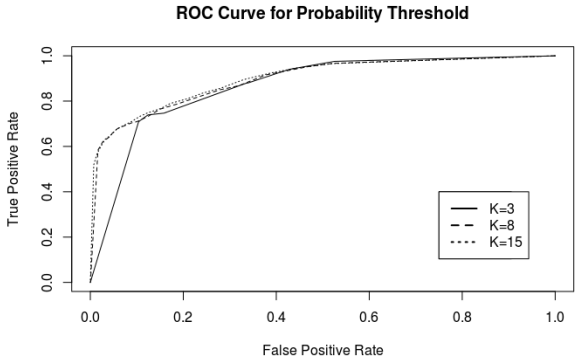

Figure 11: ROC Plot of TPR/FPR as function of probability threshold. Note that K=8 performs comparably with K=15.

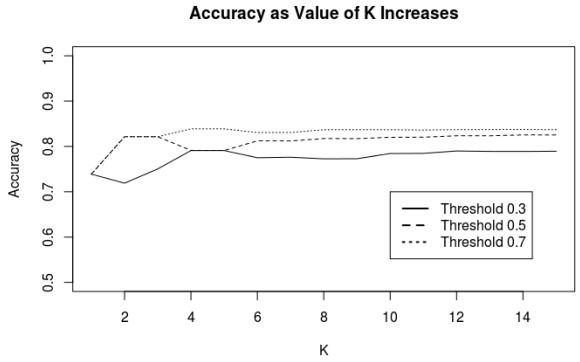

Figure 12: Accuracy plot for K=1,...,15. Prediction performance is maximal at K=4,5,6 and threshold = 0.7.

previous K large values. The key difference between logistic regression and linear regression is that the goal of logistic regression is to classify a categorical variable, which in this work is a binary variable representing the sign of a large change in health. Since the output of a logistic regression model is a probability and not the binary prediction, we need to specify a threshold at which to separate positive from negative predictions. Thus predicting the sign in this way requires two major choices: K, the number of recent large values in the hero's history to consider and the probability cutoff on which to base the binary prediction. In general, a cutoff that is too low will increase the number of false positive predictions and a cutoff that is too high will result in increased false negative predictions.

3.3.4 Logistic Regression Results. Using the training set to build the logistic regression model with the glm function in the R stats package, we evaluated the performance in predicting the signs of the validation set. The ROC plot, or receiver operating characteristic plot, is a plot of true positive rate versus false positive rate and shows the performance of a classification model as a function of some model parameter. Figure 11 depicts the ROC curve as the classification probability threshold is adjusted from 0 to 1 for several values of K, which again is the number of recent large magnitude values for the logistic regression algorithm to consider. After including several previous high value time points, there is less value added by including more. This is demonstrated by the K=8 and K=15 ROC curves. Figure 12 shows the accuracy for each value of K at three different threshold levels. Based on these results, we can see that the values of K that performed best were 4,5, and 6 at a threshold of 0.7. To include this logistic regression approach in a complete model of health forecasting, we selected K=4 and a threshold of 0.7.

3.3.5 Linear Regression for Large Values. Next, there is the question of the magnitudes of large changes in health. The reason this task was separated from the problem of sign prediction is two-fold. First, the distributions of positive large values and negative large values are somewhat different. In the training set, the mean of positive magnitudes is 335.8733 and the median is 230.5, while the mean of negative magnitudes is 251.477 and the median is 185.5. In

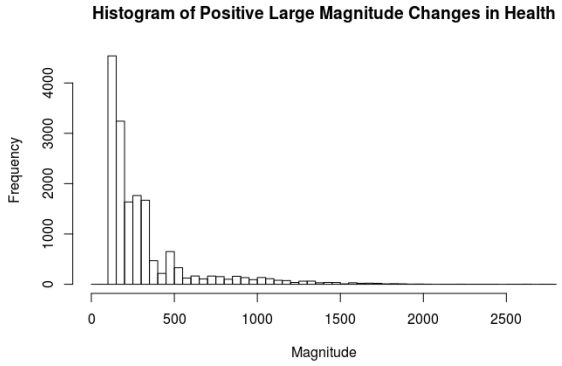

Figure 13: The distribution of positive high value changes in hero health. Each bar represents a span of 50 health points.

addition, the range tends to increase with time at a faster rate for the positive values compared with the negative values. The second reason for splitting the sign and magnitude prediction problems is due to the inherent gap in the positive and negative jumps. This method is part of the high magnitude value approach and the values between -100 and 100 are removed from this time series so the data set is essentially separated into two distributions of points: positive changes and negative changes. The accuracy of the sign prediction model using logistic regression described previously is such that we can be reasonably confident from which of these sets of points (positive or negative) the next large value will come.

Figure 13 shows the distribution of positive large values and Figure 14 shows the distribution of negative large values. Note that while both histograms have the same bar width of 50 health points, the positive values are more widely distributed with a total range of 100 to 2787. The negative values, however, only span a range of -1154 to -100. The negative values are more concentrated at smaller magnitudes than the positive values. This illustrates that positive and negative large changes in hero health follow different overall distributions and can be captured by different models.

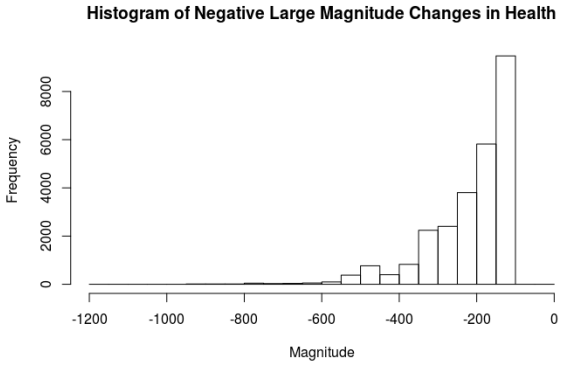

Figure 14: The distribution of negative high value changes in hero health. Each bar represents a span of 50 health points.

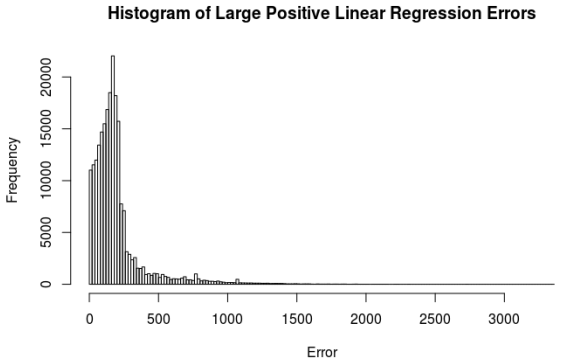

Figure 15: Distribution of errors for linear regression models for positive high values.

The model we chose for predicting magnitudes is a simple linear regression model with the total duration of the match so far as the lone independent variable. In other words, to predict how large a large value will be, we only consider how much time has passed since the match began.

3.3.6 Linear Regression Results. The linear regression models were fit by combining the high values from the training set, building a model, and testing against data points from the validation set. We again used the R stats package to produce the model. The resulting error distributions are displayed in Figures 15 & 16 for the positive value model and negative value model respectively. The mean error for the positive model was 200.5 and the mean error for the negative model was 116.6. The negative value model more frequently produced errors that were smaller than the positive value model. One possible reason for this is that whenever a hero respawns back into the game after death, they experience a large jump in health equal to their current maximum health. This is not going to be captured by such a simplistic model that lacks context about the specific game state other than duration.

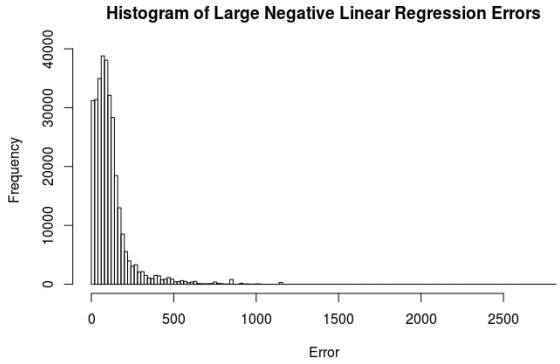

Figure 16: Distribution of errors for linear regression models for negative high values.

3.4 Combined Forecasting for Both Low and High Values

In this section we describe the results of the full forecasting model, combining both the high and low value models.

3.4.1 Combining the Models. Separately these models may be effective in a controlled environment. The Poisson process model is only concerned with the timing of when jump points will appear. The logistic regression model only predicts the sign of the next high value associated with said jump point and not when it appears. The linear regression models assume the sign is correct and only predict magnitude. However, it is not useful to simply accomplish these tasks separately. A combined model to create complete health forecasts is the goal—ideally in as close to real time as possible.

Therefore, we combined these models into a single forecasting system. Low values were processed by an ARMA model and high values processed by the three models described above. Given a particular time step, we used the nonhomogeneous Poisson model to estimate the number of high values in the interval from the given time step to 20 time steps ahead. Then, the logistic regression model predicted the sign of the first high value, then incorporated that prediction into the input to the logistic regression model if there was more than one high value to be estimated in the interval. Once a prediction for the signs was produced, we used the linear regression model to estimate the magnitudes, using the average of the time interval as the input because the equation for number of jump points in the Poisson point process doesn't tell us where the points may be in the timeline. The sum of these predictions was added to the sum of the low value predictions to produce a final forecast for the hero's change in health over the time interval.

3.4.2 Combined Model Results. For the combined approach, we utilized the parameters discovered while testing the models on the validation set and the test set for evaluation. Unlike previous experiments, here we only took 4 test intervals from each player in each match due to computation time. The results of forecasting health changes using the combined model resulted in the error distribution shown in Figure 17. The mean error was 244.1 and the median was 232.3. However, when the forecast change minus

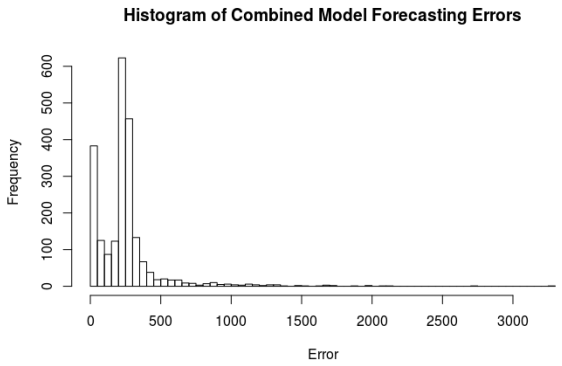

Figure 17: Distribution of errors for combined low and high value models.

actual change was greater than zero, the mean error was 179.0, but when the forecast change minus actual change was less than zero, the mean error was 262.6. The model had more difficulty predicting positive change than negative change. One reason for this could be the quantity of negative changes being much larger than positive changes overall. Contrasted with the histograms from previous models, the errors for the combined approach take on an interesting distribution. Instead of frequencies somewhat smoothly decreasing around the most frequent errors, there are two peaks. One peak is near zero and the other (and highest) peak is in the 200-250 range. While the model often had low error, an error closer to the mean of 244.1 was more frequent.

4 CONCLUSIONS AND FUTURE WORK

4.1 Conclusions

In this work, we introduced an approach to forecast change in hero health in DotA 2 by splitting the data into small and large changes. Using this value-split approach we predicted both types of changes separately using different statistical models. For small changes, we used an autoregressive moving-average (ARMA) model and for large changes we used a combination of methods. Large changes ("jump points") were predicted using nonhomogeneous Poisson point process estimation while logistic regression and linear regression were used to predict the sign and magnitude of these points, respectively. We then combined these methods into one forecasting system. Using these methods we were able to forecast small changes in health with better accuracy than a baseline. We achieved greater than 80% accuracy in predicting the sign of large changes in health and reasonably accurate results with linear regression models for the magnitudes of such large values, in particular negative changes. Classification accuracy on the sign of the next large change in health was quite high, suggesting that our logistic regression model successfully captured patterns that are indicative of what may happen to the player. The combined model performed well, with most errors being less than 300 health points away from the actual change in health. Again, negative changes were overall easier to predict than positive changes. Our model made no distinction about recovery in health due to a previously dead hero respawning into the game and this could have been a source of larger error for positive changes. A forecasting model should incorporate this type of context into its predictions.

Overall, our results suggest that there is value in considering changes to a hero's health as the result of separate processes that may not be captured easily by a single model. Prediction of events such as change in health is possible by observing past data and considering the various processes that influence health, but remains a difficult problem as health is a result of complex, multiplayer interaction.

However, there are some limitations to this work. Our approach does not incorporate any expert knowledge about DotA 2 and considers only health data and not other information about the game state. In addition, our statistical models for both predicting when large health changes will occur and for predicting their magnitudes are quite simple in that they only incorporate the match duration in their prediction. This led to poor performance particularly with the nonhomogeneous Poisson process model.

4.2 Future Work

Future work will include incorporating other types of information into the forecast. We calculated the cross-correlation between health and another attribute, mana, which represents a resource that is spent by heroes to use special abilities and automatically regenerates in a fashion similar to the health attribute. We found that there is some lagged correlation between health and mana, meaning that it may be possible to use a hero's current remaining mana to improve forecasts. Other potentially useful information is that large changes in health tend to occur close to one another, even among players on opposing teams. Knowing this, we can improve the process for predicting when large changes may occur. To further add to predictive power, information about both other players and non-player characters should also be considered because changes to health are typically a direct result of interaction between characters.

Another consideration is the more general problem of how to properly identify and capture separate processes involved with gameplay. We identified one such separation between low and high value changes in health and set the threshold at 100, but this could be varied to determine a good cutoff in terms of forecasting ability. A more context-aware method of capturing disparate gameplay processes may also improve prediction. Large battles and destroying buildings are clearly important events, and understanding when these take place could allow a model to adapt accordingly. Distance from the player to other players and also to important structures such as Towers and Ancients could be used as an indicator to instruct when to use a specific forecasting model.

REFERENCES

[1] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control.* John Wiley & Sons.
[2] David R Cox. 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), 215–242.
[3] Ethan W Dereszynski, Jesse Hostetler, Alan Fern, Thomas G Dietterich, Thao-Trang Hoang, and Mark Udarbe. 2011. Learning Probabilistic Behavior Models in Real-Time Strategy Games.. In *AIIDE*.
[4] Anders Drachen, Matthew Yancey, John Maguire, Derrek Chu, Iris Yuhui Wang, Tobias Mahlmann, Matthias Schubert, and Diego Klabajan. 2014. Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2). In *Games media entertainment (GEM), 2014 IEEE*. IEEE, 1–8.

[5] Graham Kurtis Stephen Erickson and Michael Buro. 2014. Global State Evaluation in StarCraft.. In *AIIDE*.
[6] G Gardner, Andrew C Harvey, and Garry DA Phillips. 1980. Algorithm AS 154: An algorithm for exact maximum likelihood estimation of autoregressive-moving average models by means of Kalman filtering. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 29, 3 (1980), 311–322.
[7] Nataliia Pobiedina, Julia Neidhardt, Maria del Carmen Calatrava Moreno, and Hannes Werthner. 2013. Ranking factors of team success. In *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 1185–1194.
[8] François Rioult, Jean-Philippe Métivier, Boris Helleu, Nicolas Scelles, and Christophe Durand. 2014. Mining tracks of competitive video games. *AASRI Procedia* 8 (2014), 82–87.
[9] Gabriel Synnaeve and Pierre Bessiere. 2011. A bayesian model for opening prediction in rts games with application to starcraft. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*. IEEE, 281–288.
[10] Pu Yang, Brent E Harrison, and David L Roberts. 2014. Identifying patterns in combat that are predictive of success in MOBA games.. In *FDG*.