The Problem with Win Probability

Sujoy Ganguly and Nathan Frank
STATS AI Group

1. Introduction

In Game 1 of the 2017 Western Conference Finals between the Golden State Warriors and the San Antonio Spurs, with 7:53 remaining in the third quarter and the Spurs up 78-55, Kawhi Leonard re-injured his left ankle while taking a jump-shot [1]. Following Leonard's departure from the game, the Warriors went on an 18-0 run, eventually winning the game. In Figure 1, we show the "standard" win probability curve (blue curve) for that game - highlighting the specific sequence (shaded) when Leonard left the game and the Warriors went on their run. The "standard" model of NBA win probability considers game time, possession, and point differential to predict match outcome using logistic regression [2].

Even though it shows the Warriors' dramatic comeback, this plot also highlights one problem with win probability - ***Problem 1: it lacks sufficient context***. Intuition tells us that with Leonard off the court the Spurs are less likely to win; however, this is not shown in the win probability curve. Win probability models should be responsive to in-game contextual features such as injuries and fouls. Additionally, win probability models should incorporate team identity, i.e. team strength. In the example from Figure 1, the Warriors' dominance in prior seasons and demonstrated ability to come back to win in similar situations should be captured by the model (similar to the Patriots who should have been afforded more credit than the 0.3% likelihood of winning with two minutes to go in Super Bowl LI [9]). Whether or not the prediction changes, its uncertainty should change, which highlights another problem with current win probability models - ***Problem 2: there is no measure of uncertainty (i.e., there are no error bars)***. The "standard" win probability model predicts the likelihood of a single, binary outcome given an in-game scenario. In reality, there are many paths to any one outcome, and the model should highlight this complexity. The fact that such issues are common in win probability estimates highlights a final problem - ***Problem 3: There are no publicly available datasets or models against which researchers and analysts can compare*** [6].

In this paper, we propose two methods to address the issues of context and uncertainty, 1) team lineup encoding and 2) explicit prediction of the score difference distribution. Team lineup encoding is a way to generate a meaningful and compact representation of an in-game matchup. The win probability model can then use this compact representation to enhance prediction. **In the first part of this paper, we show that inclusion of the encoded lineup improves in-season accuracy of a win probability model from ~75% to ~88%.** While we obtain significant model improvement, lineup encodings tend to overfit to specific seasons. To prevent in-season overfitting, we can 1) regularize the encoding space or 2) regularize the prediction space. For this work, we chose the latter by predicting the final score difference distribution using a Mixture Density Network (MDN) [7]. **By using an MDN with lineup encoding, we achieve an in-season (internal) accuracy of ~86.7% and an out of season (external) accuracy of ~82%**. Furthermore, by predicting final score difference distributions, rather than individual values, we can directly measure the uncertainty in our predictions by capturing the myriad possible outcomes for each game state.

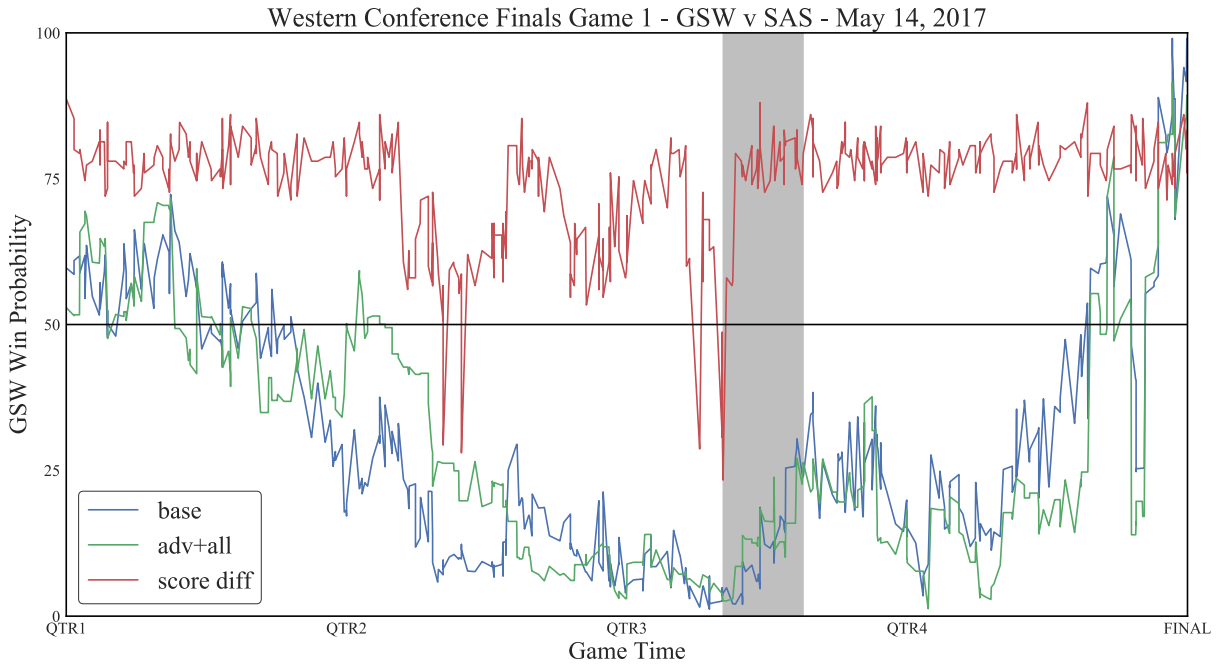

Figure 1: Win Probability versus Game Time for NBA Western Conference Finals Game 1 - Golden State Warriors vs. San Antonio Spurs (2017-5-14) – for the baseline (base) Random Forest Classifier (blue), advanced with box score and lineup encoding (adv+all) Random Forest Classifier (green), and MDN final score difference predictor (red). The shaded region shows the GSW run following the Leonard injury.

2. Introduction

For this work, we compiled a dataset consisting of over 8.7M National Basketball Association (NBA) play-by-play events from the 2002-03 through 2016-17 seasons. Each play-by-play event P_t is described by game time, ball possession, and score difference, representing the base feature set. The base features are enhanced by the inclusion of home and away team as well as event identities (P_t^+). In the analysis that follows we refer to any win probability model which only considers the base feature set as a "baseline" model and those including team and event identity as "advanced" models.

2.1. Contextual Features

In order to provide better game and matchup context, we additional include home and away team box scores X_t, pregame lineups L_t, and on-court labels O_t. These are formulated as follows:

- X_t: Each team's box score is aggregated in game up to time t and consists of team assists, blocks, fouls, rebounds (offensive, defensive, and team), steals, and turnovers.
- L_t: A lineup vector for each game ($\boldsymbol{l}_{i=\{H|A\}}^{j=\{0,\ldots,14\}}$) is constructed for every player (j) on each team (i) consisting of player identity, starter and availability flags, season to date games played, games started, minutes played, plus minus, minutes per game, plus minus per game, and fouls per game. Team lineups are the union of 15 such vectors, with padding of empty vectors (zeros) for rosters of less than 15 players. L_t is the concatenation of home and away lineups.
- O_t: We define the on-court flag at time t for player j on team i to be 1 if that player is currently active, i.e. on the court, and 0 if he is not. The on-court vector at time t is the union of 30 such flags, ordered to align with L_t.

In total this data set contains 352 features for each play-by-play event. For all models, we consider data from the 2002-03 through 2013-14 NBA seasons for training and in-season testing (internal) and leave the 2014-15 through 2016-17 seasons untouched for additional out of season (external) testing. The internal training data is split into 80/15/5 train/test/evaluation splits by game. Therefore, each set of data is from a different set of games, rather than events, from a common set of seasons.

3. Baseline Win Probability Models

The task of a win probability model is to predict the eventual game outcome given the current game state. A naïve formulation of this model may simply report the cumulative historical average for a given game state, i.e. the fraction of teams that went on to win under identical conditions. More commonly, the outcome label is predicted via logistic regression [2, 4, 5] or random forests [6] using a set of inputs similar to our previously defined base feature set. In some formulations, these features are enhanced by the inclusion of the Las Vegas point spread or betting odds.

Model	Features	Accuracy	
		Internal	External
Baseline (base)	P_t	75.4%	74.7%
Advanced (adv)	P_t^+	78.1%	74.3%
Baseline + Encoding (base + enc)	$P_t + E_t$	78.8%	74.3%
Advanced + Encoding (adv + enc)	$P_t^+ + E_t$	83.1%	74.5%
Baseline + Box Score + Encoding (base + all)	$P_t + X_t + E_t$	86.5%	75.2%
Advanced + Box Score + Encoding (adv +all)	$P_t^+ + X_t + E_t$	**88.1%**	75.1%
MDN	$L_t + X_t + O_t + P_t$	86.8%	**82.0%**

Table 1: Accuracy scores for various win probability models.

Since there are no publically available models, we establish a baseline win probability model using a mix of features. We train a random forest classifier (RFC) on various combinations of the input features using 10-fold cross validation to optimize model parameters. In each case the RFC consists of 20 classification trees, each of which sees at most the square root of available features at each level, and splits on a minimum 200 samples with a minimum of 100 samples per leaf node. We then evaluate performance on the internal test set of unseen games from the same range of seasons as training (internal). Additionally, we test on the external test set taken from the unseen seasons (external). Performance numbers are reported in Table 1.

We see that our "baseline" model (base) performs comparably to existing baselines and that we obtain a slight boost with the inclusion of team identity in our "advanced" model (adv) [2]. In both cases, however, the most important feature is point difference followed by game time. In the "advanced" model, possession is the least important feature overall.

4. Lineup Encoder

To include a compact representation of the lineups we set a Neural Network a secondary task of predicting which players are on the court O_t at every game time t given the lineup features L_t, current game state P_t, and box score X_t. Our Neural Network (Figure 7) consists of three fully connected encoding layers with $ReLU$ activation and 256, 256, 128 and ten units respectively. A drop out layer follows each layer. The last layer of the encoder gives the encoding features E_t used by other models.

Then, to predict O_t, we decode with two fully connected layers with 15 units and $ReLU$ activation, and 30 units with $sigmoid$ activation, respectively. To train the network, we minimize the cross-entropy via backpropagation using the Adam optimizer [8]. Since there are only ten valid on-court flags, we weight the cross-entropy to avoid the trivial solution of all players being off the court. We train this network on 1000 games from the 2002-2014 training set. Any model that uses E_t does not see these 1000 games.

We obtain on-court prediction accuracy of 18.10%, meaning on average we incorrectly predict that six players are on the court when they should be off the court. The poor prediction of the on-court players is not surprising given the difficulty of the task, particularly when given a single time step from which to make the prediction. However, the ten-feature encoding space should still contain substantial information about the relationship between the matchup, box score, and score difference.

4.1. Random Forest Classifiers with Lineup Encoding

We train a set of RFC models that use these encoding features and report the internal and external performance of these models in Table 1. We see incremental improvement in the internal prediction by the inclusion of our encoding features. We also find that the encoding features rank above event and team identity as well as possession in importance to the models. Finally, we achieve state of the art performance (~88%) by the additional inclusion of box score features.

4.2. Accuracy as a function of game time

In addition to computing the overall accuracy of the Random Forest models we also computed the accuracy as a function of game time (Figure 2). We compute the cumulative accuracy (internal) of each Random Forest Model over one minute bins. We find that the model accuracy increases monotonically with time for all models. We further note that models which include X_t and E_t (Figure 2 cyan and gold) rapidly increase to an accuracy of ~88% within the first few minutes of the game, indicating that the model finds the game type (manifold) rapidly, as compared to models with E_t alone.

Unfortunately, this state of the art performance does not extend to unseen seasons. When using our best performing model on the 2016-2017 season, we see a return to baseline performance (Table 1 and green curve Figure 1). This regression in performance is due to a separation in the encoding space wherein E_t for each season lives on its own manifold. Since the RFC was trained to understand the encoding space for the 2002-2014 seasons, it fails to understand the part of the encoding space in which the new season lives.

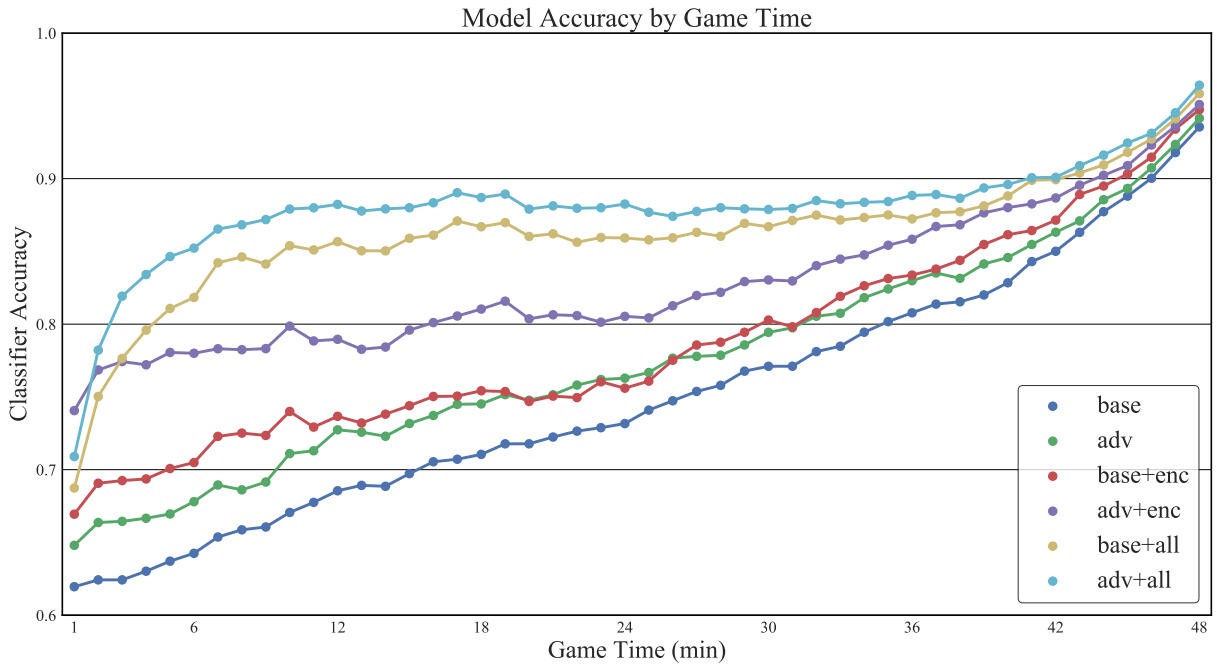

Figure 2: Random Forest Model Accuracy versus Game Time. Cumulative accuracy of the model in 1 minute bins of game time.

In this way, the RFC model is overfitting to the 2002 – 2014 seasons. Therefore, to improve our performance in new, unseen seasons, we need to regularize either the encoding space or the prediction space. For this paper, we have chosen to regularize the prediction space since this will also give us uncertainty measures on the outcome predictions. We leave the regularization of the lineup encoding space for future work.

5. Score Line Predictor

To regularize the prediction space, we chose to predict the score difference ($s = Home\ Team\ Score - Away\ Team\ Score$) distribution instead of the match outcome. By predicting the score difference distribution, we allow for the potential of various outcomes and natural measures of our prediction uncertainty. In other words, we formulate the task of outcome prediction as a one (game state) to many (possible score differences) problem.

We have chosen to use a mixture model to predict the score difference distribution since it allows us to use neural network architectures and backpropagation to find the optimal set of mixture parameters. Our Mixture Density Network (MDN) takes L_t, X_t, O_t, and P_t as inputs. The MDN (Figure 8) consists of a fully connected layer with a $tanh$ activation function and 128 units followed by a batch normalization layer. We then reinject P_t and pass this into a dense layer with $tanh$ activation and 64 units. We follow this layer by an output layer with linear activations. These outputs are the parameters of a mixture of $N = 10$ Gaussian distributions, such that the end of game score difference S_{t_f} has a distribution of

$$Q\left(s = S_{t_f}\right) = \sum_{i}^{N} \pi_i \mathcal{N}\left(\mu_i, \sigma_i\right), \quad (1)$$

where π_i is the weight of the i^{th} distribution, and μ_i and σ_i are the mean and standard deviation of the i^{th} Gaussian $\mathcal{N}$, respectively. During training, we minimize $-\log\left\{Q\left(s = S_{t_f}\right)\right\}$ to find the optimal set of mixture parameters $\{\pi_i,\ \mu_i,\ \sigma_i\}_{i\in[0,N]}$.

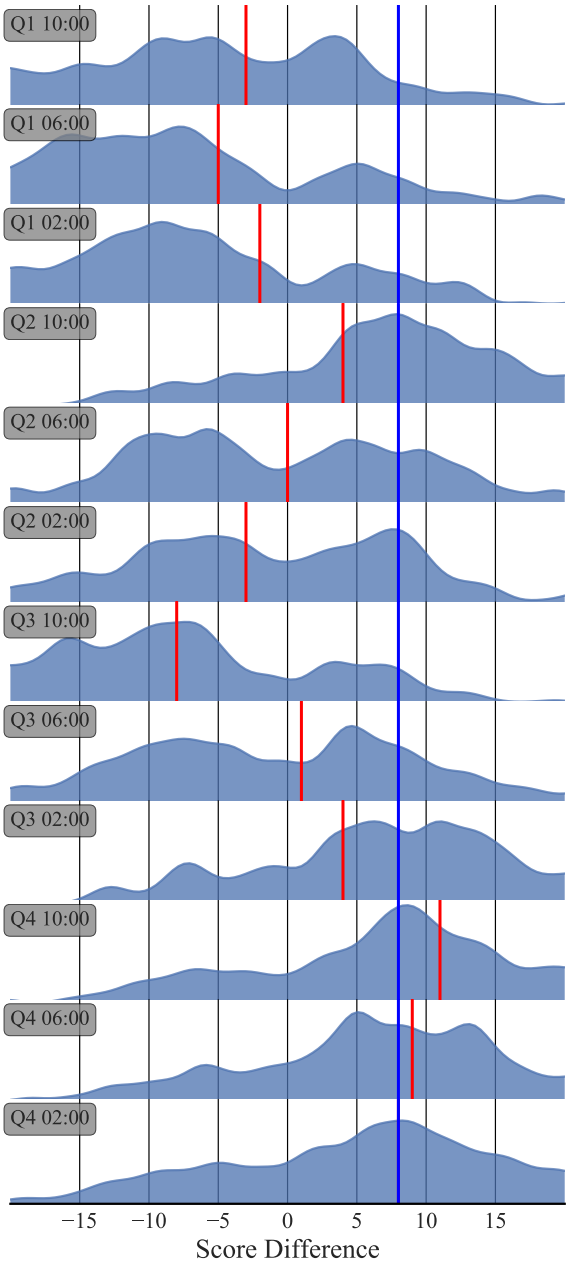

Figure 3: Predicted end of game Score Difference Distribution (shaded area), current score difference (red line), and actual end of game score difference (blue line).

We trained the MDN model on 4440 games from the 2002-2014 seasons and tested performance on 500 games from 2002-2014 (internal), and 500 games from the held out 2016-2017 season (external). We find that our in-season accuracy of 86.8% is comparable with our state of the art RFC. More importantly, we find that our out of season performance of 82% is much better than for any of our other models (Table 1).

In Figures 3-6 we plot $Q_t(s = S_{t_f})$ for 12 different game times t for four example games, including the Golden State vs San Antonio Game (Figure 6). We first notice that our predictions can be multimodal, mirroring the myriad possible outcomes a game can have. We also notice that the initial predictions are matchup specific. It is critical to note that we have not explicitly provided a pre-game prior, but rather the model learns one based on the matchup specific features. For example, in Figures 4 and 5 our initial predictions imply that the game is a toss-up between the two teams, whereas in Figures 3 and 6 the home teams are heavily favored.

We further see that as the games progress and the game state changes (red lines in Figures 3-6), the model distributions evolve as well, often showing state switching (Figures 3-5) and mean shifting over time. Again, we see that the evolution of $Q_t(s = S_{t_f})$ is context and matchup specific.

In Figures 3-5 the predictions oscillate between the two initial states as the score difference oscillates. In other cases, where the prior is strong (Figure 6) the state stubbornly refuses to change. This stubborn refusal of the MDN to change for the Golden State vs San Antonio game is also seen in the win probability curve (Figure 1 red curve) and is likely a reflection of the Warriors' dominance in prior seasons.

We also notice that our distributions do not collapse or narrow with time. The apparent insensitivity of the distribution variance to game time is a function of the Markovian nature of the current prediction architecture. Currently, the model does not have a sense of how much time remains in the game, only that some amount of time has passed. By using a recurrent architecture (instead of fully connected), this should be resolved and is left to future work.

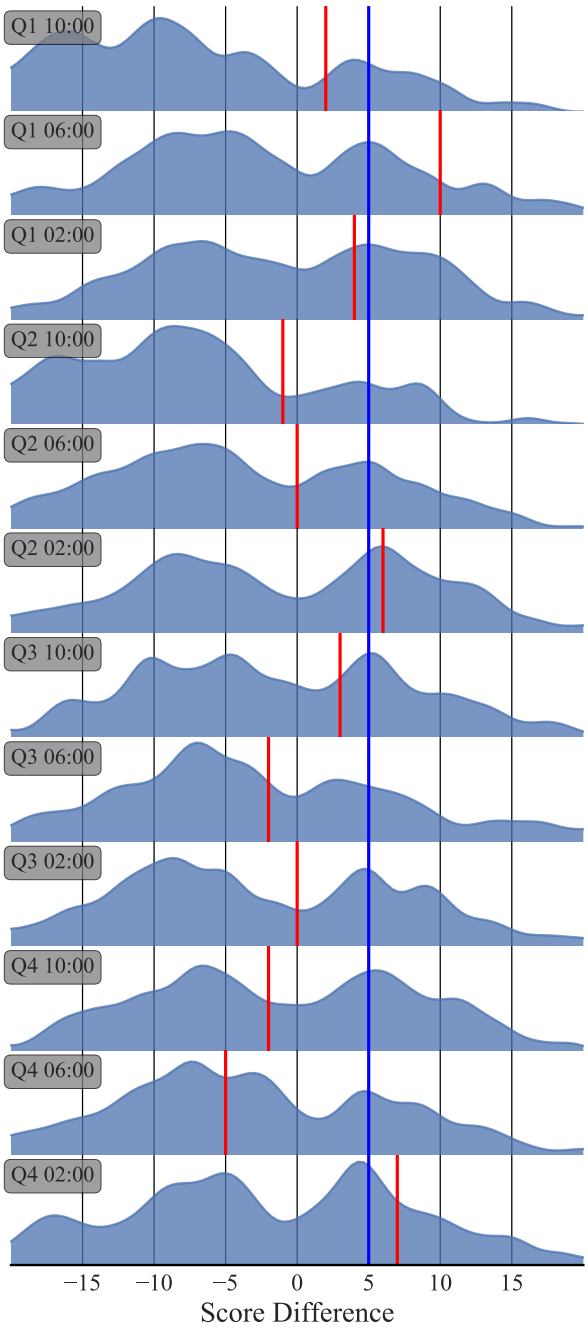

Figure 5: Predicted end of game Score Difference Distribution (shaded area), current score difference (red line), and actual end of game score difference (blue line).

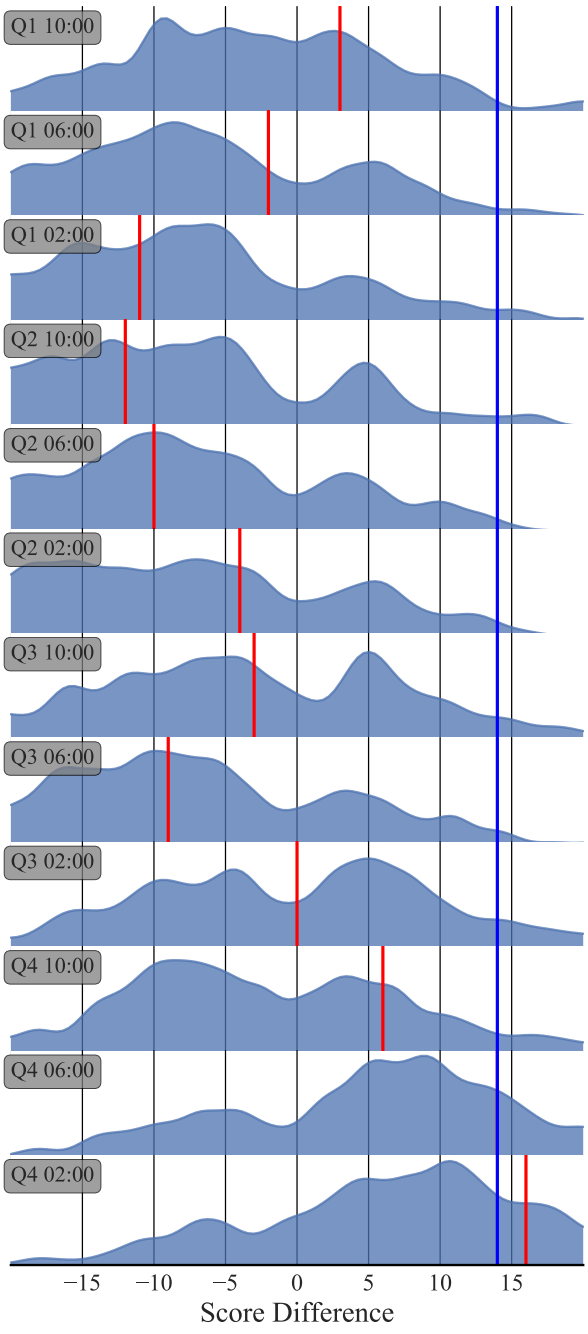

Figure 4: Predicted end of game Score Difference Distribution (shaded area), current score difference (red line), and actual end of game score difference (blue line).

Lastly, we notice that even in situations where our model incorrectly predicts the outcome, i.e. where our cumulative score difference prediction favors the wrong team (Figure 4), there is often a mode in the distribution that captures the actual result. It is these diverse, context specific predictions that allow our MDN to greatly improve outcome prediction over baseline models.

6. Summary

In this paper, we highlight three current issues with win probability models: ***i) lack of context, ii) no measure of prediction uncertainty, and iii) no publicly available datasets or models against which researchers and analysts can compare***. To remedy the last issue, we are releasing our dataset and baseline win probability model to the research community (see https://www.stats.com/data-science/). For the first issue, we were able to encode lineup specific information into our neural network leading to substantial improvements over current methods (88% vs 75%). To capture the uncertainty of our predictions, we moved from match-outcome prediction to final score difference prediction, providing a measure of uncertainty by estimating the likelihood of all possible scores. In addition to capturing the uncertainty of a given match outcome prediction, this approach allows for interactive story-telling applications by enabling the exploration of "alternative outcomes"- for example, what if Kawhi Leonard did not get injured during Game 1 of the 2017 Western Conference Finals. In the future, by complementing score prediction with a recurrent architecture, we should see the score distributions collapse, thereby allowing us to determine points of no return in a match, and determine what decisions are irreversible or lead to uncertainty growth in the inferred outcome prediction.

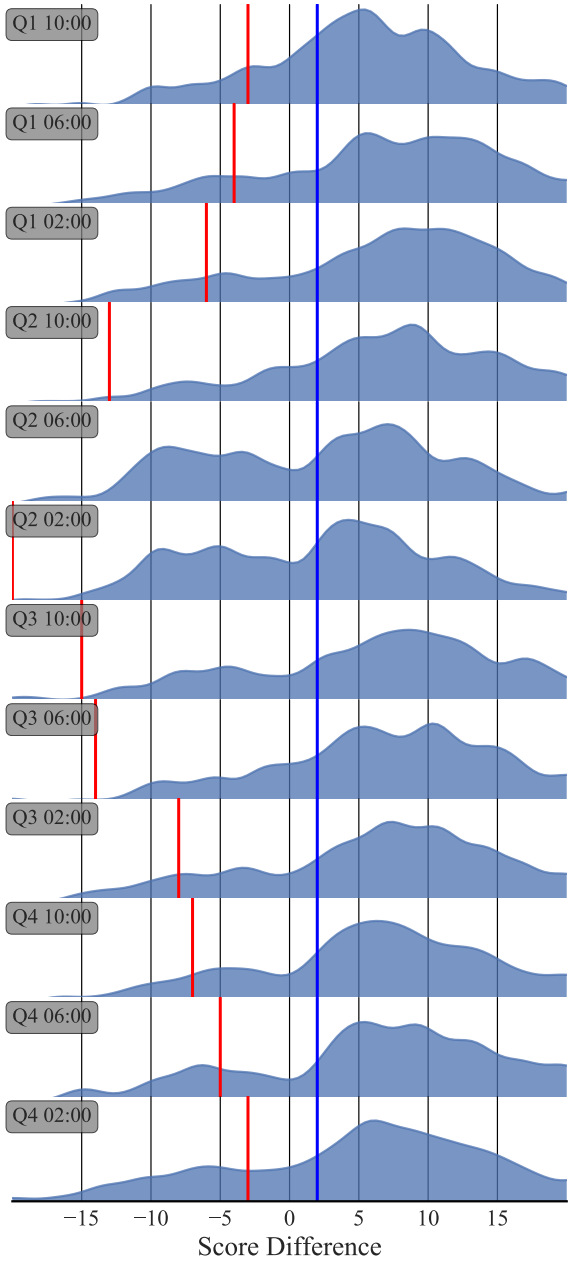

Figure 6: Predicted end of game Score Difference Distribution (shaded area), current score difference (red line), and actual end of game score difference (blue line) for the Golden State vs San Antonio game.

References

[1] Dubow, J. "San Antonio Spurs' Kawhi Leonard ruled out for Game 2 vs. Golden State Warriors." *ESPN,* 15 May 2017, http://www.nba.com/article/2017/05/15/san-antonio-spurs-kawhi-leonard-out-game-2.
[2] Beuoy, M. "Updated NBA Win Probability Calculator." *Inpredictable*, 6 Feb 2015, http://www.inpredictable.com/2015/02/updated-nba-win-probability-calculator.html.
[3] Asif, M & McHale, I.G. "In-play forecasting of win probability in One-Day International cricket: A dynamic logistic regression model." *International Journal of Forecasting*, 32(1):34-43, 2016.
[4] Pelechrinis, K. "iWinrNFL: A Simple and Well-Calibrated In-Game NFL Win Probability Model." arXiv preprint arXiv:1704.00197.
[5] Lock, D. & Nettleton, D. "Using random forests to estimate win probability before each play of an NFL game." *JQAS*, 10(2): 197–205, 2014.
[6] Schechtman-Rook, A. "Introducing NFLWin: An Open Source Implementation of NFL Win Probability." *PHD Football*, 2016 Sept. 1, http://phdfootball.blogspot.com/2016/09/introducing-nflwin-open-source.
[7] Bishop, C. M. "Mixture density networks." *NCRG*, 004, 1994.
[8] Kingma, D., & Ba, J. "Adam: A method for stochastic optimization." *ICML*, 2014.
[9] Paine, N. "*The Pats' Comeback Was Incredible — Even If You Think The Falcons Blew It*", http://fivethirtyeight.com/features/patriots-falcons-comeback-super-bowl/

Appendix

Neural Network Architectures

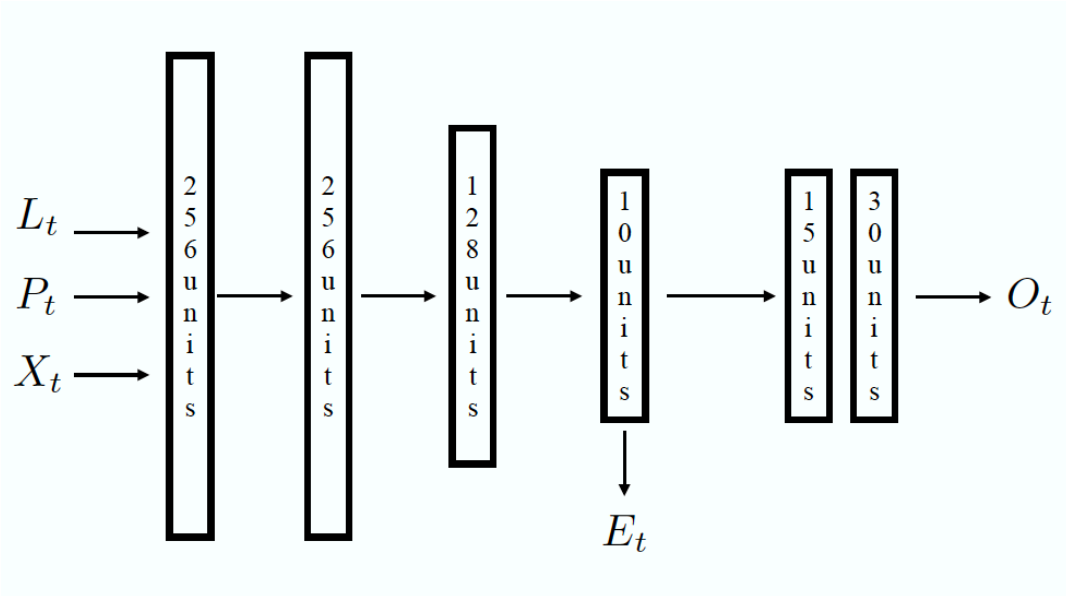

Figure 7: Lineup Encoder Network Architecture.

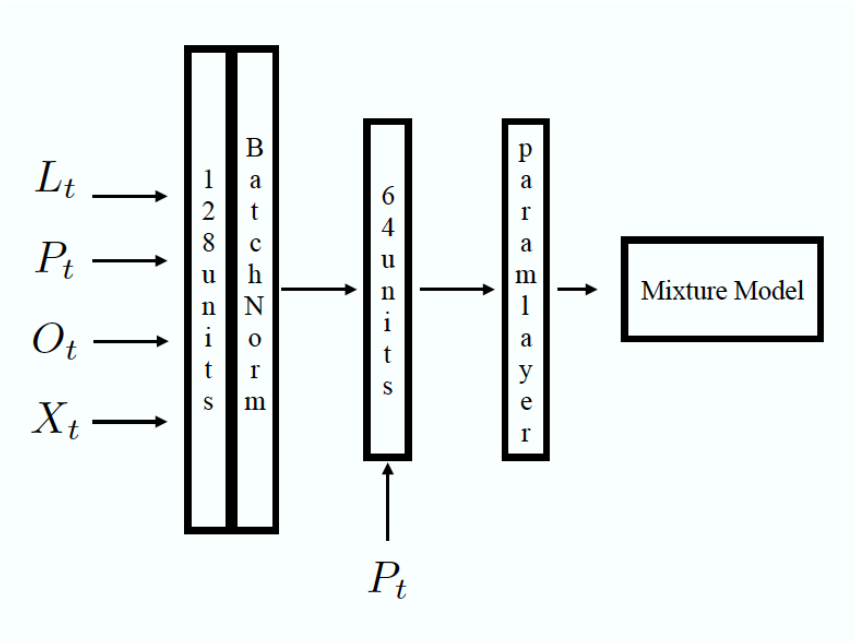

Figure 8: Mixture Density Network Architecture.