NBA Game Predictions based on Player Chemistry

Prastuti Singh[†] and Bai Yang Wang[‡]
{prastuti, bwang87}@stanford.edu

[†]Department of Applied Physics, [‡] Department of Physics, Stanford, CA 94305

***Abstract*— The popularity of statistics driven performance analysis in major sports leagues speaks to the success of machine learning in understanding complex human behavior. For instance, in the NBA, over 65% game outcome prediction accuracy has been achieved using various supervised machine learning techniques based on individual player statistics. However, despite the empirical wisdom that team chemistry and rivalry play affects a team's performance, the correlations between players within the same team and in opposing teams remains under-explored. Here we report our efforts in achieving prediction accuracies comparable to established machine learning models utilizing an original team-chemistry-focused model. In addition, our model offers qualitative insights into players' chemistry that could provide valuable information for future trades.**

I. INTRODUCTION

Machine learning provides an interesting platform to attempt to understand complex human behavior. We are interested in applying these techniques to the NBA, which is replete with detailed statistics on players and games. More specifically, the motivation of this project is to understand the role that player-player chemistry may play in predicting game outcomes and providing quantitative predictions of chemistry between players. The inputs for our algorithm will be the identity of the players on each team for a given game. We will then use logistic regression with a linear model and a quadratic model to output a prediction on which team will win the match. We also provide a comparison using a multi-layer perceptron neural net. The goal for this work is to examine the effectiveness of the quadratic model in accurately predicting NBA match outcomes and then examine the learned parameters to be able to answer questions such as whether players perform better playing with certain players over others and the opposite, do players perform worse when playing against certain players?

II. RELATED WORK

NBA games are notoriously difficult to predict. The average score differential is about 10 points with average scores varying from 100-112 points, depending on the season. Still, most machine learning models are able to achieve 65-70% accuracy in predicting the game outcome. Previous work by Uudmae [1] compared the accuracy of several models on predicting game winners based on past games. Uudmae was able to achieve a maximal accuracy of 65% using a Neural Network Regression model. Avalon, Balci and Guzman achieved similar accuracy among the many models they tested (linear regression, GDA, PCA & SVM, Random Forest, Adaptive Boost) [2]. The input features for their models was team statistics and the dataset was limited to data from one season. Miljković, Gajić, Kovačević and Konjović used data mining techniques with Naive Bayes to achieve an accuracy of 67% [3]. The best model we have seen so far was by Beckler, Wang, and Papamichel, who used a combination of linear regression and k-means clustering to obtain a maximum accuracy of 73% (single season accuracy) and an average accuracy of 70% [4]. There is also some work on predicting player chemistry using clustering and a proposed SPM framework by Maymin, Maymin and Shen [5] that predicts which players or groups of players would have game 'synergy'. It should be noted that these works all relied on extensive player statistics as input for their model. In contrast, our work here relies only on the identity of the players on the teams playing for each game (see Dataset and Features).

III. DATASET AND FEATURES

Historical data for games from the 2014-2015 season to the current season was collected from kaggle.com [6]. Data for later seasons was collected from basketball-reference.com [7]. From this dataset, we extracted the names of the teams playing (team A and team B), the date, and the number of points scored by each team (points A and points B). The game data was limited to regular season games since players often play and behave differently during the playoffs. Playoff data was collected and processed but not used for training purposes. We also collected roster data for each team from the 2014-2015 season to the current season[7]. The total number of players in the league throughout these seasons was 963. Thus, the feature vector for each team is a vector of length 963 with a 1 in every position corresponding to one of its players and 0s elsewhere. The date for each game is used to access the roster information in that season for each team. The feature vectors for each team is then stacked in order to create the input for our model. The labels are either 1 or 0, depending on whether team A won or team B won, respectively.

The dataset contains duplicates for each game (Team A vs Team B on Date D as well as Team B vs Team A on Date D). A separate dataset was created for games without duplicates. Both game sets were shuffled prior to training in order to remove time-series bias. Note that the feature vectors for our experiments only capture the player's identity

rather than any information about their skill (points per game or three-point shooting percentage, for example). This was a deliberate choice on our part in order to examine whether simply the identity of the player could accurately capture the dynamics of the player chemistry. Future work will include adding player skill information to improve model prediction accuracy and identify factors that determine two players' chemistry.

IV. METHODS

The data was fit using three models. The first was simple logistic regression with an input features vector x that is a stacked vector of the team inputs features t_1 and t_2. The prediction for this model is given by

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}.$$

The second model is a quadratic classifier that attempts to capture player chemistry. The model assumes that each team has some intrinsic symmetry amongst its players (some players play better with other players on the same team) and an anti-symmetric component that represents some adversity between the two players. The symmetry matrix will be represented by S and the adversity matrix with A. With our model, the prediction will be estimated as

$$h(t_1, t_2) = g(t_1^T S t_1 - t_2^T S t_2 + 2 t_1^T A t_2),$$

where t_1 is a Team 1's feature vector/roster and t_2 is Team 2's feature vector/roster and $g(x)$ is the sigmoid function. This prediction function can also be written as

$$h_\theta(x) = g(x^T \theta x),$$

where x is now $\begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$ and θ has the form $\begin{bmatrix} S & A \\ -A & -S \end{bmatrix}$. We then use a standard gradient descent algorithm to maximize the log likelihood in order to optimize θ. Initially we computed the gradient and wrote the code for the gradient ourselves (vectorized as much as possible) but we found that it was slow to run. We then tried mini-batch gradient descent with different batch sizes but that was still slow and the dev accuracy was lower than full batch. Finally, we used Pytorch [8] to compute the gradient and found that led to a significant increase in speed.

Simply using gradient descent however does not guarantee the symmetry requirements of θ. Therefore, we will need to impose some condition on our fit in order to ensure that S and A satisfy the appropriate symmetry conditions. We test two methods for this:

- Projected gradient descent
- Regularization

Projected gradient descent maps θ to the closest value of θ that satisfies the given conditions (S is symmetric and A is anti-symmetric) at each iteration [9]. This was implemented in our algorithm by first dissecting θ into four quadrants, as shown above, corresponding to Q_1, Q_2, Q_3, and Q_4. Then we pick out the symmetric components of $Q2$ and $Q4$ and the anti-symmetric components of $Q1$ and $Q3$:

$$Q_{2(4)}^{\text{sym}} = \frac{Q_{2(4)} + Q_{2(4)}^T}{2}$$

$$Q_{1(3)}^{\text{anti-sym}} = \frac{Q_{1(3)} - Q_{1(3)}^T}{2}$$

The $Q_{2(4)}^{\text{sym}}$ pair and the $Q_{1(3)}^{\text{anti-sym}}$ pair are then each averaged to update the S and A matrices as follows:

$$S := \frac{Q_2^{\text{sym}} - Q_4^{\text{sym}}}{2}$$

$$A := \frac{Q_1^{\text{anti-sym}} + Q_3^{\text{anti-sym}}((Q_3^{\text{anti-sym}})^T)}{2}$$

depending on whether $Q_3^{\text{anti-sym}}$ or $(Q_3^{\text{anti-sym}})^T$ is closer to $Q_1^{\text{anti-sym}}$ in terms of L_2 norm.

The second method, regularization, imposes a cost penalty if S is not symmetric and A is not anti-symmetric. This is implemented by adding a term to the loss function as follows with weight 1 (utilizing the same notation from previous paragraph):

$$\|Q_1 + Q_1^T\|_2 + \|Q_2 - Q_2^T\|_2 + \|Q_3 + Q_3^T\|_2 + \|Q_4 - Q_4^T\|_2$$

.

Finally, we also fit our data using a multi-layer perceptron neural net as a comparison. A number of different architectures and activation functions were examined for tuning the hyperparameters.

V. RESULTS AND DISCUSSION

For the experiments, data from all six seasons was shuffled together and then split into training, dev and test sets in a 64:16:20 ratio. Model selection was based on the highest dev set accuracy.

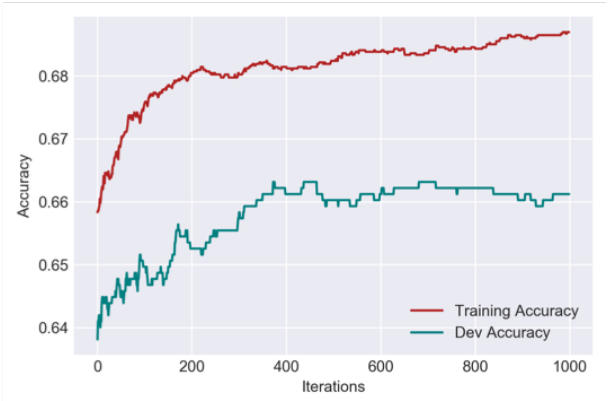

Fig. 1: Training and dev accuracy for the linear logistic classifier with the best hyperparameters.

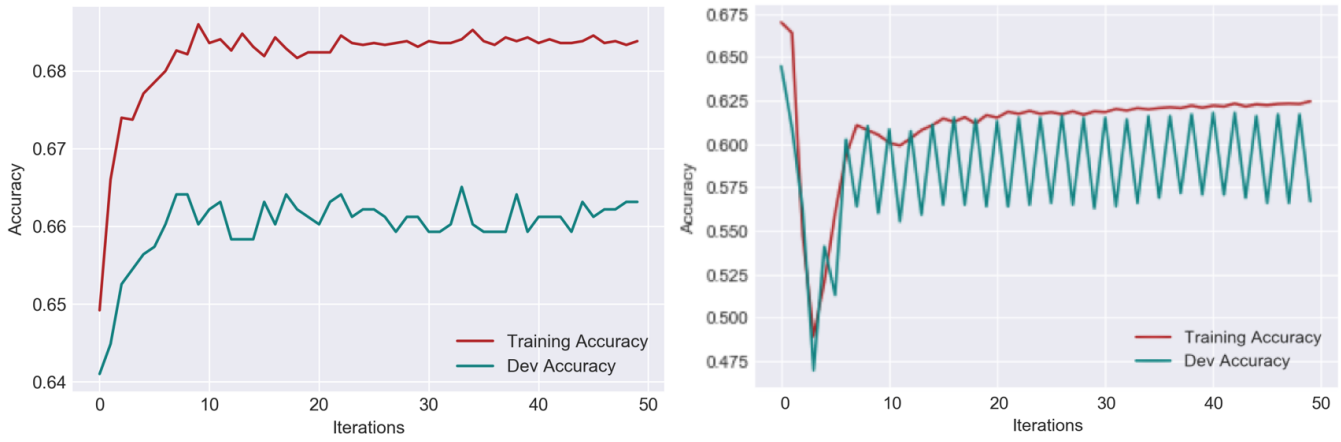

Fig. 2: Training accuracy vs dev accuracy as a function of iterations for the quadratic model with projection (left) and regularization (right) to place constraints on the S and A sub-matrices in θ.

Linear Model

Figure 1 shows a plot of the training and dev accuracy vs the number of iterations for the linear logistic model with tuned hyperparameters. The training set accuracy is 69% and the dev set accuracy is 66%. Despite the simplicity of the model, note that this is already comparable to other reported prediction accuracies in the literature.

Neural Network

The dataset was also fit using a multi-layer perceptron neural net from scikit-learn [10]. In order to determine the best hyperparameters, we optimized for the following:

$$\begin{aligned}&\text{solver} = [\text{adam}, \text{sgd}]\\&\text{hidden\_layer\_size} = [(i,j)], i,j \in 5[1,8]\\&\text{activation} = [\text{logistic}, \tanh]\end{aligned}$$

In the case of the stochastic gradient descent solver, the learning rate was also tuned. In addition, single layer architectures were considered with the number of nodes varying in the set [5, 10, 20, 30, 40, 50, 100, 964] (There are 963 players in the league + 1 for the bias term). A grid plot of the training and dev accuracies for the two layer model is shown in Fig.

It is noticeable that the architectures with the highest dev accuracies (green in the right plot) also had the lowest training accuracies (blue in the left plot). This is a strong indication that the other architectures were overfitting the model. The neural net architecture with the highest dev rate was with two hidden layers, the first with 30 nodes and the second with 25 nodes, all with a logistic activation function. The dev accuracy was 65.9% and the final test accuracy was 65.1%, which is comparable to the linear model although slightly lower. It is possible that better test accuracies could be achieved with the neural net model with more extensive parameter tuning.

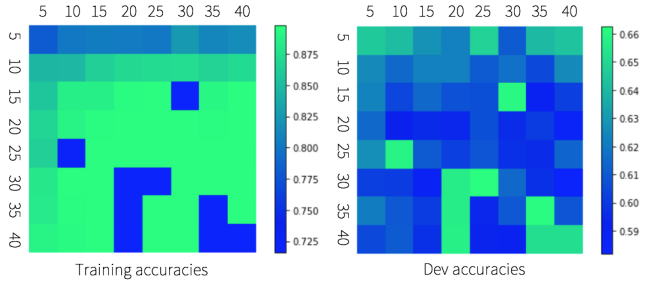

Fig. 3: Grid plot of training accuracy and dev accuracy for two hidden layer neural net model. The number of nodes in the first layer is along the vertical axis and the number of nodes in the second layer is along the horizontal axis.

Quadratic Model

The quadratic model was tested with two methods to constrain S and A to the appropriate symmetries. The training and dev accuracy for both models with best hyperparameters is shown in Fig. 2. The projection based method achieves substantially higher training (68.4%) and dev accuracy (66.5%) than the regularized model.

Note the periodic oscillations in the dev accuracy for the regularized model. The training accuracy is stable (with minor fluctuations) beyond 15 iteration but the dev accuracy oscillates between the training accuracy and 56%. Given that these oscillations are present in both the training and dev accuracy and that the maximum of dev oscillations are roughly the same as the training accuracy, we believe that this is indicative of model instability and potentially some overfitting. It may be possible to obtain a more stable fit with a lower regularization weight but then the constraints on S and A will be less strict. In contrast, with projection, we see a more stable model that guarantees a symmetric S and an anti-symmetric A.

In addition to sampling points from all seasons for the training, dev and test set, we also checked the accuracy of our

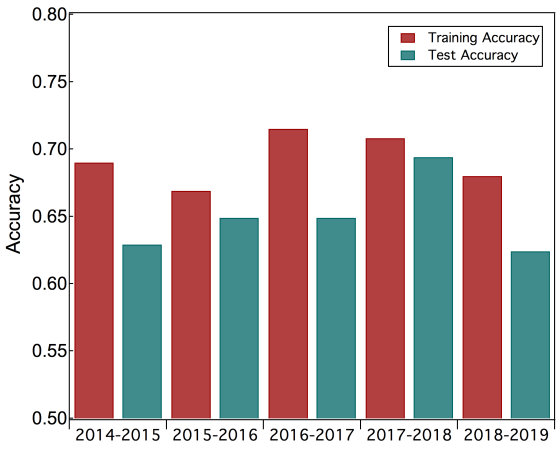

Fig. 4: Training and test accuracy for the quadratic model when applied on each season.

model by training and testing for each individual season. In this case, the data for each season was split into the training and test set in a 70:30 ratio. The hyperparameters for each season was optimized independently and the results for the accuracies are shown in Fig. 4. The average test accuracy is 64.9% and the highest accuracy is 69.4%. This is the same evaluation method used and reported by Beckler, Wang, and Papamichel in their paper although a different set of seasons/years was used for the data.

Parameter Analysis

To examine the 'synergy' between two players, we can examine the normalized θ (and therefore S and A) that was fit using the quadratic model. Fig 3. provides a potential interpretation of the S and A matrix elements. There are three types of matrix elements:

- Diagonal elements of S - indicator of player's individual skill
- Off-diagonal elements of S - indicator of how much the two players as teammates contribute positively to the win percentage of their team
- Off-diagonal elements of A - indicator of the difference in the players' contribution to their teams' winning chance when they are on opposing teams

Let us consider the S and A coefficients for two sets of players based on the θ of the best fit model (shown in Fig. 2 (left)):

- LeBron James and Kyrie Irving
  - $S_{\mathrm{LJ,KI}} = 0.246$
  - $A_{\mathrm{LJ,KI}} = 0.003$
- Stephen Curry and Kevin Durant
  - $S_{\mathrm{SC,KD}} = 0.142$
  - $A_{\mathrm{SC,KD}} = 0.144$

Based on these parameters, we could qualitatively interpret that LeBron James and Kyrie Irving have 'better' chemistry

Player 1 individual skill	player 1 & 2 contribution to winning when on same team	...
player 2 & 1 contribution to winning when on same team	Player 2 individual skill	...
...	...	...

0	player 1's contribution to winning when against player 2	...
player 2's contribution to winning when against player 1	0	...
...	...	...

Fig. 5: Interpretation of the elements of the S (left) and A (right) matrices.

than Stephen Curry and Kevin Durant[1]. The A coefficients suggest that LeBron and Irving square off evenly while Durant outshines Curry in team winning contribution when they face each other. This holds up against empirical observations since Durant was carrying the Thunders against the Warriors before switching teams. We can also look at the individual player performance according to the diagonal elements. In agreement with most NBA commentators, the model suggests that LeBron is the best player among these four superstars. Curry comes second, then Durant and finally Irving.

Focusing on LeBron James's S coefficients with other players, we find that he plays best with Danny Green. LeBron James and Danny Green have only played together in 2019-2020 of the seasons in our dataset and the Lakers have won 88% of their games so far, whereas LeBron's overall win percentage is 62.3%. This suggests that the model could be fitting to the win percentage combinations of two players. This is not surprising since this is the only game-relevant information that we have supplied through our inputs and labels.

Model Comparisons

The training and test accuracy for all three models is given below. All three models gave similar accuracies, which are also comparable with the results seen by Uudmae [1], Avalon, Balci and Guzman [2] and Miljković, Gajić, Kovačević and Konjović [3].

Full dataset		
	Training Accuracy	Test Accuracy
Linear	68.7%	64.8%
Quadratic	68.4%	65.1%
NN	71.9%	65.1%

We also evaluated model stability in two ways:

- Predicting later seasons with single season model

[1]We accidently switched the S coefficients for these two pairs in the poster. The values and analysis in the paper are the correct versions.

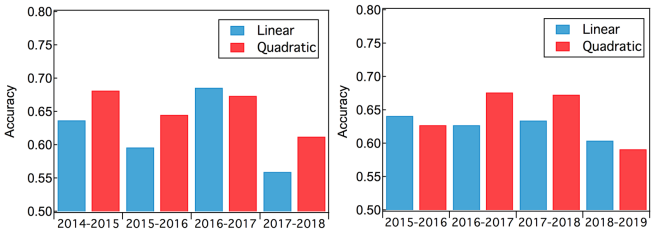

Fig. 6: Measures of model stability for the linear and quadratic model throughout the individual seasons.

- Predicting each season based on data from all previous seasons

The results from both are shown in Fig. 6. On the left, the model has only been trained on the 2014-2015 season and then tested on each subsequent season. Accuracy for the 2014-2015 season is based on dev accuracy on 20% of the dataset. Based on the standard deviations for both sets, it seems that the quadratic model is more stable from one season to the next than the linear model in predicting the outcomes of the next season. Given that the model is only trained on games from one season, as new players are added to the league, no statistic would be available for them. Additionally, we can intuitively say that while a player's individual performance may vary from season to season, usually his personal chemistry with other players will persist through time. For example, an retiring Wade would probably perform better than his average when playing with his buddy LeBron, even though age factor has begun to limit Wade's performance in general. These factors would explain a decrease in accuracy for both models over time, but would be compounded in the linear case.

In the second case, we trained each model on all the previous games until that season and tested on games for that season. So for example, for the 2017-2018 season, the models were trained on all games from 2013-2017 and then tested on all games in the 2017-2018 season (excluding playoff games in both cases). In this case, while we do expect a decrease in accuracy based on skill deteriorate from age, we would limit the accuracy decline from new players to only new players from that season. In this case, we observe that both models are more stable. The exception is the accuracy for both the linear and quadratic models for the 2018-2019 season. We suspect this is caused by significant moves from major players to new teams such as LeBron James moving to the Lakers and Kawhi Leonard to the Raptors, which indicates that our model is susceptible to changes in team rosters. This could be ameliorated by adding in data from previous seasons.

VI. CONCLUSIONS

In this work, we propose and examine a new model for predicting NBA game outcomes relying solely on the identity of each player on the two teams. The proposed model, the quadratic classifier, utilizes a 'synergy' and an 'adversity' matrix to quantify the players collaborative and competitive nature in addition to each player's individual talent. In order to evaluate the model's validity, we compare its performance in predicting game outcomes against a logistic model and multi-layer perceptron neural net, from which we conclude that the performance of all three models in predicting game outcomes is comparable. From model stability evaluations, we find that quadratic model is more stable than the linear model. Even though the performance of all three models is comparable, the quadratic model allows us to examine the fitted parameter θ in a qualitatively meaningful way for indications of chemistry amongst the players. We can examine individual cases of 'synergy' and 'adversity' between any two players and (jokingly) resolve controversial issues amongst NBA superstar players such as who is better, Stephen Curry or LeBron James (not so controversial and our model agrees!).

In order to increase the performance of our models, we believe that the next step would be to add information about the number of minutes played by each player for each game (normalized by the total number of minutes in a game). This would allow the model to account for injuries and bench status, information that is currently missing in our model. Currently, the model is fitting more broadly to team chemistry rather than player chemistry but this information would specialize it to player chemistry. Unfortunately this data has been difficult to obtain. In addition, we could add in other player statistics (e.g. 2-pt and 3-pt shooting percentage) so evaluate whether this leads to better predictions.

As explained in the paper, the model would also benefit from data from previous since currently, it cannot compute correlations for players who have never played together and it also treats players who have stayed on the same team throughout this five year period as the same person. Finally, to prevent deterioration of the model over time due to decline in player skill beyond a certain age, we could scale the input for each player based on their age since in the current model, player skill is assumed to be constant for all seasons but in reality, beyond a certain age, player skill begins to decline. Previous work on modeling win-share percentage as a function of player age [11] could easily be integrated with our current model.

CODE

All data and code can be found at https://github.com/prastutisingh/cs229.

CONTRIBUTIONS

Prastuti Singh and Bai Yang Wang contributed equally to this project.

REFERENCES

[1] J. Uudmae, "Predicting NBA Game Outcomes," Retrieved from http://cs229.stanford.edu/proj2017/final-reports/5231214.pdf.

[2] G. Avalon, B. Balci, J. Guzman, "Various Machine Learning Approaches to Predicting NBA Score Margins," Retrieved from http://cs229.stanford.edu/proj2016/report/Avalon_balci_guzman_various_ml_approaches_NBA_Scores_report.pdf.

[3] D. Miljković, L. Gajić, A. Kovačević and Z. Konjović, "The use of data mining for basketball matches outcomes prediction," In *8th International Symposium on Intelligent Systems and Informatics*, 309-312, 2010.
[4] M. Beckler, H. Wang, and M. Papamichael, “NBA Oracle,” Zuletzt besuchtam, 17:2008–2009, 2013.
[5] A. Maymin, P. Maymin, E. Shen, "NBA Chemistry: Positive and Negative Synergies in Basketball," Int. J. Comp. Sci. Sports, 12:2, 2013.
[6] https://www.kaggle.com/ionaskel/nba-games-stats-from-2014-to-2018.
[7] https://www.basketball-reference.com.
[8] A. Paszke et al. "Automatic differentiation in PyTorch, 2017.
[9] N. He, “Lower bounds & Projected Gradient Descent.” Retrieved from http://niaohe.ise.illinois.edu/IE598_2016/pdf/IE598-lecture10-projected%20gradient%20descent.pdf.
[10] F. Pedregosa et al. Scikit-learn: Machine Learning in Python, JMLR 12, 2825-2830, 2011.
[11] N. Vaci, D. Cocic, B. Gula, M. Bilalic, "Large data and Bayesian modeling—aging curves of NBA players," Behav. Res. Methods, 51:1544-1564, 2019.