UNIVERSITY OF CALIFORNIA, MERCED

Markov Chain Models and Data Science Applications

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Applied Mathematics

by

Li-Hsuan Huang

Committee in charge:

Professor Harish S. Bhat, Chair
Professor Rick Dale
Professor Arnold D. Kim

2018

The dissertation of Li-Hsuan Huang is approved, and it is acceptable in quality and form for publication on microfilm:

(Professor Arnold D. Kim)

(Professor Rick Dale)

(Professor Harish S. Bhat, Chair)

University of California, Merced

2018

DEDICATION

To my parents and sister.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank several people for making my PhD journey possible. First are my family. They have been phenomenal for the past five years. Their encouragement and support really kept me going when I was struggling, doubtful and thought of giving up. Second is my PhD and research adviser, Harish Bhat. I recall numerous fun and interesting conversations regarding research, basketball, and life in general to help propel my research. Mostly important, he helped steer my research in a clear direction when I seemed lost. Third is my friend and collaborator, Sebastian Rodriguez. Sebastian was an amazing undergraduate student I worked with between 2015 and 2016. He helped mine all the NBA data I needed for the basketball projects. I owe many thanks to him. Fourth are my committee members, Professor Rick Dale and Professor Arnold D. Kim. Their valuable feedback has helped me think about various aspects of basketball modeling. Fifth are my PhD year 2013 cohort and the Applied Math Unit, especially Michael Stobb and Mario Banuelos for contributing ideas and troubleshooting some of my coding problems. Finally, I would to acknowledge funding and resource support I received during the PhD program: UC Merced Applied Mathematics Summer 2017 Research Fellowship, Summer 2018 Applied Mathematics Travel Scholarship, and the Merced cluster, which is funded by the National Science Foundation (Grant No. ACI - 1429783).

VITA

2018	Ph.D., Applied Mathematics, University of California, Merced
2013-2018	Graduate Teaching Assistant, University of California, Merced
2013	Postgraduate Work in Applied Mathematics, California State University, Northridge
2011	Undergraduate Workshop Assistant, California State University, Fullerton
2011	B.A., Mathematics, California State University, Fullerton

PUBLICATIONS

Bhat, H.S., Rodriguez, S., Huang, L., "Eliminating Absorbing States and Driving Markov Chains to Desired Equilibria", *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2018.

Bhat, H.S., Rodriguez, S., Huang, L., "Learning Stochastic Models for Basketball Substitutions From Play-by-Play Data", *Machine Learning and Data Mining for Sports Analytics Workshop (MLSA) at European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD)*, 2015.

Bhat, H.S., Rodriguez, S., Dale, R., Heit, E., Huang, L., "Citation Prediction Using Diverse Features", *Data Science and Big Data Analytics Workshop at the International Conference on Data Mining (ICDM)*, 2015.

ABSTRACT OF THE DISSERTATION

Markov Chain Models and Data Science Applications

by

Li-Hsuan Huang
Doctor of Philosophy in Applied Mathematics

University of California Merced, 2018

Professor Harish S. Bhat, Chair

National Basketball Association basketball is a dynamic sport played by various 5-person units (lineups) from two teams for 48 minutes. Researchers have attempted to model the dynamics of the game using play-by-play data, which contains a game log of events on the court and times at which those events happened. Recent work involving NBA play-by-play data used discrete-time Markov chains to model team possession-related events to predict final game scores. Moreover, the availability of optical player tracking data has allowed researchers to model game progression on an individual player level. While the work captured details of games, game progression driven by lineups used by each team and the amount of time played by each lineup on the court were not considered.

Using NBA play-by-play data, we sought models for time series detailing transitions of lineups (states) and the times at which a new lineup was used. Because substitutions occur at random times, we built a continuous-time Markov chain (CTMC) model for each NBA team in which each state corresponds to a unique lineup. Using the 2014-15 NBA season, we correctly predicted 12 out of 15 playoff outcomes. Nevertheless, all CTMC models suffered from absorbing states.

Each time series might have a final state which has never been reached previously. In this case, the final state will end up being an absorbing state for the maximum-likelihood-estimate Markov model. In the long run, an absorbing Markov chain has equilibrium distribution supported entirely on the set of absorbing states. To remedy this problem in a basketball simulation, we considered three strategies: i) reroute to a more probable state if an absorbing state is reached, ii) remove lineups that played less than 2 minutes and lineups corresponding to absorbing states in a season, and iii) allow simulation to stay in the absorbing state if it is the final state reached before the game is over. Combining these strategies with scoring-rate models built using singular value decomposition and weighted observations, we obtained best training and test accuracy of 75% and 70%, training on the first 75 games and testing on the remaining 7 games for the 2015-16 NBA season. A test accuracy of 70% compares favorably with state-of-the-art models.

These ad hoc absorbing-state strategies, developed for NBA data, however, might not be valid in other applications; hence, we developed stable optimization algorithms that systematically solve the issue of absorbing states so that the Markov chains reach desired equilibria and achieve zero long-term training error. We defined long-term training error as the one-norm difference between the equilibrium distribution of an MLE derived Markov chain and the actual fraction of time spent in each state on the training data. We then applied the optimization methods to the NBA data and to two finite-state, discrete-time biomedical data sets. The results showed smaller long-term training and test errors compared to naive MLE models. Furthermore, for the biomedical data, the methods yield similar long-term errors as hidden Markov models while requiring significantly less training time.

Chapter 1

Introduction

Within the field of sports analytics, basketball modeling has attracted significant recent interest. Machine learning and deep learning methods have been applied to predict both outcomes and margins of victory for NBA games for betting purposes. Nonetheless, all such efforts require manual feature extraction and hyperparameter tuning in order to achieve acceptable performance.

National Basketball Association basketball is a dynamic sport played by various 5-person units (lineups) from 30 teams. In a regulation-length game, two teams play against each other for 4 quarters of 12 minutes each. The objective of each team is to outscore the opponent. Researchers have attempted to model the dynamics of the game using play-by-play data, which contains a game log of events and times for which those events happened. Recent work involving NBA play-by-play data used discrete-time Markov chains to model possession-based events to predict final game scores (Shirley 2007; Štrumbelj and Vračar 2012). Moreover, the availability of optical player tracking data has allowed researchers to model game progression on an individual player level, to gain insights into players' decision-making tendencies (Oh *et al.* 2015; Cervone *et al.* 2016). While this captures details of games, it does not consider substitution, lineup usage, and the amount of time played by each lineup.

Using NBA play-by-play data, we seek to model substitutions, i.e., transitions between lineups (states) and the times at which substitutions/transitions occur. Each state maps to a unique 5-person unit. Because substitutions of lineups occur at random times, we saw a need for a continuous-time stochastic model over a discrete-state space model. In this dissertation, we consider a continuous-time Markov chain (CTMC) model for each NBA team, augmented by scoring-rate models.

We mined play-by-play data from the 2014-2016 NBA seasons. We obtained structured data containing information such as which lineup played on the court for two teams before a substitution occurred. We also recorded team scores and point differentials when substitutions occurred. More features such as player fouls were also mined.

I now outline each chapter of the dissertation.

In Chapter 2, using play-by-play data from all 2014-15 regular season NBA games, we built a generative model that accounts for substitutions of one lineup by another together with the plus/minus rate of each lineup. The substitution model consists of a CTMC with transition rates inferred from data. We compared different linear and nonlinear regression techniques for constructing the lineup plus/minus rate model. We used our model to simulate the NBA playoffs; the test error rate computed in this way is 20%, meaning that we correctly predicted the winners of 12 of the 15 playoff series. However, the substitution models suffered from absorbing states. In many regular-season NBA games, several rare lineups are played when a team is leading or trailing by large margin. Typically those lineups are the last lineups to play on the court till the end of game and might not have been observed previously. Hence, in the estimated models, those lineups will be the absorbing states. The equilibrium distributions of those models do not capture the actual fraction of time spent in each state.

In Chapter 3 we extended our work from Chapter 2 to the 2015-16 NBA season. We delved into the issue of absorbing states in Markov chain models whose equilibrium distributions differ significantly from the actual fraction of time spent in each state. For predicting a player or a lineup playing time, absorbing Markov chains have equilibria supported entirely on the set of absorbing states. For a basketball game, the last 5-person unit on the court might not be playing much. Hence, such CTMCs yield poor predictive accuracy. To resolve the absorbing state issue, we considered three ad hoc strategies: i) reroute to a more probable state if an absorbing state is reached, ii) remove lineups that played less than 2 minutes and lineups corresponding to absorbing states in a season, and iii) allow simulation to remain in the absorbing state if it was the final state reached before the game is over. Combining these strategies with scoring-rate models built using singular value decomposition, weighted observations based on dates by percentage and exponential functions, we obtained best training and test accuracy of 75% and 70%, training on the first 75 games and testing on the remaining 7 games for the 2015-16 NBA season. The 70% test accuracy compares favorably to state-of-the-art models.

Motivated by our work in NBA analytics, we answered in Chapter 4 the question, "How can we eliminate absorbing states from absorbing Markov chains so that the new Markov chains have desired equlibria?" We considered an absorbing Markov chain model a naive model, whereas the new Markov chain model after absorbing state removal is a fixed model. A Markov chain has a desired equilibrium if it is exactly the actual observed fraction of time spent in each state. To remedy the problem of absorbing state(s), we developed stable Markov chain optimization (MCO) algorithms for discrete-time Markov chain and continuous-time Markov chain models. Each algorithm finds a sparse perturbation to the absorbing Markov matrix so that the new estimated matrix has equilibrium distribution matching exactly the observed fraction of time spent in each state. We formulated this as a linear programming problem and solved it efficiently and accurately using Python linear programming solvers `cvxopt` and `mosek` (Andersen *et al.* 2018; ApS 2018).

Applying MCO to the NBA data, we achieved zero long-term training errors for lineup playing times for all teams. The long-term training (respectively, test) error is defined as the one-norm error between the equilibrium of a Markov chain and the true fraction of time spent in each state in the training (respectively, test) set. Training on the first 40 and first 60 non-overtime games for each team, and testing on an average of 40 and 20 games, root mean-squared error going from naive to fixed models is reduced by 60%. The fixed model, on average, is off by 30 seconds for each lineup. This is a significant improvement from previous models.

We further tested the optimization method on two discrete-time biomedical data sets obtained from the R package `markovchain`: Holson and preproglucacon. The Holson data set contains life history trajectories for 1000 unique patients, each measured at 11 points in time. The measurement at each time has value 1, 2, or 3. Preproglucacon data is the DNA sequence for the gene that encodes the protein preproglucacon. This data consists of 1572 observations with bases A, T, C, G coded numerically as 1, 4, 2, 3. Similar to the continuous-time counterpart, the MCO on the two data sets achieved zero long-term training errors and smaller long-term test errors compared to absorbing Markov chain models (naive models). While these long-term errors are comparable to those from hidden Markov models (HMM), we observed that MCO has much shorter training time compared to HMM.

Chapter 2

Modeling Basketball Substitutions and Scoring Rates

2.1 Introduction

If one watches a basketball game played in the NBA, one cannot help but notice that players are substituted in and out with regularity. A large difference between basketball and other sports such as football/soccer and baseball is that a player who is substituted out of the game can return to play at a later time. Substitutions can substantially alter the strategy employed by the 5-person unit on the court. Many teams field, at different times of the game, different lineups in order to change the emphasis placed on aspects such as (but not limited to) rebounding, pace and fast break opportunities, or long-range shooting.

In short, an NBA team is actually a collection of different 5-person units. On average, in the 2014-15 regular season, teams used 15.1 different 5-person units per game. In this work, we use play-by-play data to build stochastic models for the dynamics of these 5-person units. Combining this model of substitutions with scoring models for each 5-person unit, we obtain generative models that can be used to simulate games. The ultimate goal of these models is to answer questions such as: in a 7-game series between two teams, what is the probability each team will win? Motivated by this goal, in the present work, we seek baseline continuous-time stochastic models that can be used as a starting point for further modeling efforts.

Our work builds on different strands of the literature. Discrete-time Markov chain models of basketball have been considered in Shirley (2007), for instance. One particularly successful model uses a discrete-time Markov chain to rank NCAA basketball teams (Kvam and Sokol 2006). Classification methods from machine learning have been applied to basketball "box score"-type data to make daily predictions of the winners of college basketball games (Shi *et al.* 2013). Continuous-time stochastic models have been considered by Peuter (2013), though in these models the lineup of players on the court is ignored. Finally, very recent work models the spatial location of all players on the court during the game (Cervone *et al.* 2016), with possessions modeled as a semi-Markov process. Perhaps the work closest to ours is Oh *et al.* (2015), which develops a probabilistic graphical model to simulate matches, including changes to team lineups. One of the main conclusions of Oh *et al.* (2015) is that the outcome of individual games and series are sensitive to changes in the team lineup. Our work uses this as a starting point for modeling.

2.2 Data Collection

Although sources such as NBA.com and ESPN provide statistics of teams and individual players, we found it difficult to obtain, from such sources, statistics on the performance of 5-person units or lineups. To obtain this information, we mined 2014-15 regular season NBA play-by-play data from `knbr.stats.com`, supplemented by data from `Basketball-Reference.com`, to account for substitutions taking place between quarters. For each play-by-play HTML page, we used `Beautiful Soup`, a Python package, to scrape the information needed from the text descriptions of particular plays.

To give an example of how the data appears after processing, we present Table 3.1. Each row of this data set corresponds to a group of 10 players who played on the court for a positive amount of time before at least one substitution was made

1 (Date)	2 (Home Team)	3 (Visiting Team)	4 (Home Player 1)	5 (Home Player 2)	6 (Home Player 3)	7 (Home Player 4)	8 (Home Player 5)	9 (Visiting Player 1)	10 (Visiting Player 2)	11 (Visiting Player 3)	12 (Visiting Player 4)	13 (Visiting Player 5)	14 (Seconds Played)	15 (Home Events)	16 (Visiting Events)	17 (Total Events)	18 (Home Score)	19 (Visiting Score)	20 (Δ_i—see Eq. (3.1))
20150127	Mia	Mil	478	479	480	487	481	57	426	425	431	427	350	13	21	34	15	17	-2
20150127	Mia	Mil	479	480	487	481	484	57	426	425	431	427	149	8	27	14	20	22	0
20150127	Mia	Mil	480	487	484	485	478	57	426	425	431	427	124	7	32	12	22	24	0
20150127	Mia	Mil	487	484	485	478	185	57	425	427	430	429	97	14	6	13	29	30	1
20150127	Mia	Mil	478	484	485	185	483	425	429	430	428	432	73	4	4	8	29	30	0

Table 2.1: Sample rows of data frame produced by scraping play-by-play data.

by either team. Columns 1-3 record the date of the game and the identities of the home and visiting teams. Columns 4-8 record the identities of the five players on the court for the home team, while columns 9-13 record the same information for the visiting team. Column 14 contains the number of seconds this group of 10 players (5 from each team) played just before one substitution was made by either team. Columns 15-17 record the number of play-by-play events that have occurred for the home, visiting, and both teams since the last substitution. Columns 18-19 record the home and visiting scores at the time just before the substitution was made.

Column 20, the last column, records the change in point differential. Let the current home and visiting scores (recorded in columns 18-19) be H_i and V_i, respectively. Then the change in point differential Δ_i is

$$\Delta_i = (H_i - V_i) - (H_{i-1} - V_{i-1}), \tag{2.1}$$

with the understanding that the initial scores are $H_0 = V_0 = 0$. This quantity is the "plus/minus" of the two 5-person units on the court. If we start the first row of Table 3.1, we see that at the time the first substitution is made, $\Delta_1 = -2$, corresponding to home and visiting scores of 15 and 17, respectively. At the time the next substitution is made, the score is 20 to 22 in favor of the visiting team. Because the differential is still -2, the *change* in differential is zero, i.e., $\Delta_2 = 0$.

Viewed from the point of view of the home 5-person unit, this means that even

1 (Atl)	2 (Bkn)	3 (Bos)	4 (Cha)	5 (Chi)	6 (Cle)	7 (Dal)	8 (Den)	9 (Det)	10 (GS)	11 (Hou)	12 (Ind)	13 (LAC)	14 (LAL)	15 (Mem)
401	669	309	516	372	426	387	446	523	466	518	841	504	504	481
16 (Mia)	17 (Mil)	18 (Min)	19 (NO)	20 (NY)	21 (OKC)	22 (Orl)	23 (Phi)	24 (Pho)	25 (Por)	26 (SA)	27 (Sac)	28 (Tor)	29 (Uta)	30 (Was)
415	404	472	702	362	420	358	586	563	807	574	470	541	265	417

Table 2.2: For each of the 30 NBA teams, we record the total number of 5-man units used by the team during the 2014-15 regular season. In our Markov chain model, this is the number of states N_i for each team $i \in \{1, 2, \dots, 30\}$.

though the unit scored 5 points on offense, the unit yielded 5 points on defense. We see that the Δ_i value encapsulates both the offensive and defensive performance of a particular 5-person unit. It is better to score only 3 points on offense and yield 0 points on defense than it is to score 20 points on offense while yielding 25.

2.3 Substitution Models

Our model consists of two parts: (i) a model for substituting one 5-person unit by another, and (ii) a model for how each 5-person unit contributes to the overall score of the game. In this section, we begin by describing a continuous-time Markov chain model for substitutions. We construct one Markov chain for each of the 30 teams in the NBA; let M_i denote the transition rate matrix of the Markov chain for team i. The Markov chain for team i is completely specified by M_i. Each state of M_i is a different 5-person unit that appears in the training data for team i. Let N_i be the number of states for M_i; using the entire 2014-15 regular season as training data, we obtain the following counts: For each i, we infer the $N_i \times N_i$ transition rate matrix M_i using the MLE (maximum likelihood estimate) as derived in Section 3.7:

$$\widehat{M}_i^{j,k} = \frac{\#(j \to k)}{\alpha(j)}. \tag{2.2}$$

Here $\widehat{M}_i^{j,k}$ is the estimate for the (j,k)-th entry of M_i, $\#(j \to k)$ denotes the number of observations of a transition from state j to state k, and $\alpha(j)$ denotes the total time spent in state j. All of these values can be computed using the play-by-play data.

To validate this model's performance on the training set of all regular season 2014-15 NBA games, we simulate 8200 games for each team. We count the number of substitutions made by each team and divide by 100 to obtain a Monte Carlo estimate for the total number of substitutions made by each team in one full season of play. The simulation follows standard algorithms for sampling from a continuous-time Markov chain. Assume the system is currently in state j. We then simulate exponentially distributed random variables with rates given by row j of the transition rate matrix. The minimum of these samples gives us both the time spent in state j as well as the identity of the new state k to which we transition. We initialize the simulation using the most common 5-person unit for each team, and we terminate the simulation once it reaches 2880 seconds, corresponding to a regulation-length NBA game.

Using results for 29 of the 30 teams, the correlation between the simulated and true number of substitutions is 0.8634. For one team, the Boston Celtics, simulations predict 4627.57 substitutions in one season, while the true number is 1792. This is one indication that there are surely far better distributions than the exponential to model the time spent in one state before transitioning. We discuss ongoing work in this direction in Section 4.6.

In Fig. 2.1 and Fig. 2.2, we plot the true and simulated times played by each of the 5-man units across all 30 teams (Pearson correlation of 0.834), and the true and simulated times played by each of the 492 NBA players (Pearson correlation of 0.915). All times are in minutes. The data from the last panel has been plotted on log-scaled axes; the reported correlation is for the raw data.

Overall, the in-sample fit between true and simulated unit and player times indicate that our model is a reasonable starting point to account for substitutions and 5-person unit playing team. Clearly, further research is necessary to improve the fit and develop a more predictive model of 5-person unit time. An obvious area for improvement is to model the number of fouls committed by each player on a team. Because a player must leave the game immediately after committing a sixth foul, a player is more likely to be substituted out of the game as he accumulates more fouls. Another idea is to allow the Markov transition rates to depend on how many minutes remain in the game and the game score; towards the end of blowout games, where one team leads another by a large margin, we see teams rest their regular players in favor of bench players.

In what follows, we will show that the model developed here, despite its deficiencies and though it ignores which team actually won each regular-season game, is capable of prediction.

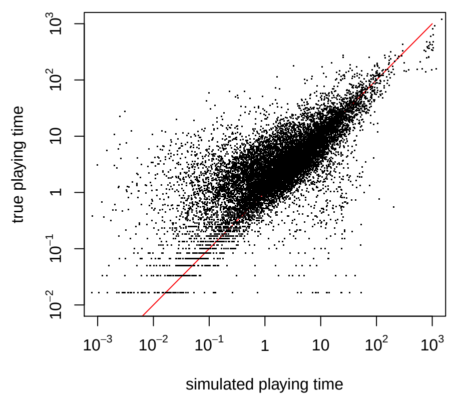


Figure 2.1: We plot true and simulated times played by each 5-man unit. We have plotted the line $y = x$ in red; deviations from this line constitute model error. Simulations are carried out using a continuous-time Markov chain model for substitutions inferred from play-by-play data. Note that the plot has log-scaled axes.

2.4 Scoring Models and Results

The second part of our model considers the change in point differential (or plus/minus) rate for each 5-person unit. We refer to this as our scoring model, even though the concept of point differential incorporates both offensive and defensive

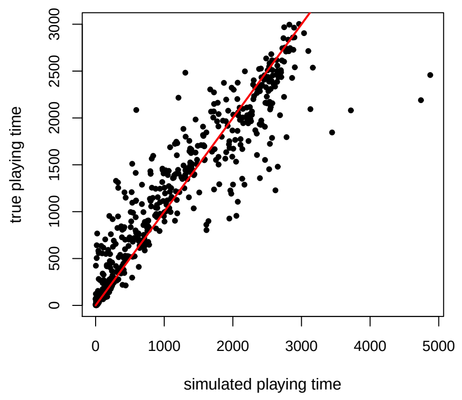


Figure 2.2: We plot true and simulated times played by each player. We have plotted the line $y = x$ in red; deviations from this line constitute model error. Simulations are carried out using a continuous-time Markov chain model for substitutions inferred from play-by-play data. Note that the plot has log-scaled axes.

performance, as described in Section 2.2.

2.4.1 Results for the 2014-15 NBA Regular Season

When simulating the continuous-time Markov chain substitution model, if the system spends τ units of time in state i, we multiply τ by the scoring rate associated with this state. This yields a change in point differential for a particular segment of game time. Summing these point differential changes across a 48-minute game, we obtain an aggregate point differential. Again, we initialize the system in the state corresponding to the lineup most often used by the team. To simulate a game between two teams, we simulate each team's aggregate point differential separately; the team with the larger value is then declared the winner.

In the most basic scoring model, we assign to each 5-person unit an average scoring rate. That is, across the entire training set, we sum the change in point differentials for a particular 5-person unit and divide by the total time this 5-

person unit spent on the court. Using this scoring rate, we simulate each of the 1230 regular season games 100 times and average the results for each game. We produce from this simulation three confusion matrices corresponding to true and predicted winners (H = home, V = visiting):

$$\begin{array}{cc} & \begin{array}{cc} H & V \end{array} \\ \begin{array}{c} H \\ V \end{array} & \begin{bmatrix} 506 & 202 \\ 200 & 322 \end{bmatrix} \end{array} \qquad \begin{array}{cc} & \begin{array}{cc} H & V \end{array} \\ \begin{array}{c} H \\ V \end{array} & \begin{bmatrix} 329 & 94 \\ 152 & 280 \end{bmatrix} \end{array} \qquad \begin{array}{cc} & \begin{array}{cc} H & V \end{array} \\ \begin{array}{c} H \\ V \end{array} & \begin{bmatrix} 220 & 54 \\ 90 & 186 \end{bmatrix} \end{array}$$

Rows correspond to predictions while columns correspond to the truth. From left to right, we show results on all games (overall accuracy of 0.67), games in which the predicted margin was ≥ 5 points (overall accuracy of 0.71), and games in which the predicted margin was ≥ 10 points (overall accuracy of 0.73).

2.4.2 Results for the 2014-15 NBA Playoffs

Because we used regular-season data to train the model, we must consider the above results to be training set results. To develop test set results, we consider the 2015 NBA playoffs. For each best-of-7 playoff series, we predict the winner, the expected margin of victory, and the probability of victory. Note that the margin here is in terms of the game score, i.e., if one team sweeps another, the margin is 4, whereas if the series goes to a seventh game, the margin will necessarily be 1. We present our predictions on the left and the truth on the right: Overall, our model correctly predicts 11 out of the 15 playoff series winners. Two of the erroneous predictions were made on series that were decided in a seventh and final game.

Ridge Regression. The next scoring model we present is built using ridge regression (Hastie *et al.* 2009). Each NBA team plays 82 games in a regular season. For team i, consider the $82 \times N_i$ matrix that indicates the number of seconds each 5-person unit played in each game. Let this matrix be X, and let $\vec{y}$ be the 82×1 vector giving the margin of victory or defeat for each game. The rough idea is to find $\vec{\beta}$ such that $X\vec{\beta} = \vec{y}$. In this case, $\vec{\beta}$ will contain a plus/minus rate for each 5-person unit.

Series	Winner	Margin	Probability	Winner	Margin
NO at GS	GS	1.43	0.75	GS	4
Dal at Hou	Hou	0.08	0.50	Hou	3
SA at LAC	SA	0.24	0.51	LAC	1
Mem at Por	Por	0.39	0.58	Mem	3
Mem at GS	GS	1.07	0.64	GS	2
LAC at Hou	LAC	0.72	0.64	Hou	1
Hou at GS	GS	1.63	0.77	GS	3
Bkn at Atl	Atl	2.05	0.82	Atl	2
Bos at Cle	Cle	2.38	0.86	Cle	4
Mil at Chi	Chi	0.92	0.66	Chi	2
Was at Tor	Tor	1.04	0.68	Was	4
Was at Atl	Atl	1.75	0.81	Atl	2
Chi at Cle	Cle	0.91	0.67	Cle	2
Cle at Atl	Cle	0.32	0.55	Cle	4
Cle at GS	GS	0.32	0.58	GS	2

Table 2.3: Predictions (left, with non-integer values of margin) and ground truth (right) for 15 NBA playoff series. The above results are test set results using the continuous-time Markov chain substitution model and the simple average scoring rate model. The model correctly predicts 11/15 of the winners.

There are two caveats. First, because $N_i > 82$ for all i, the linear system is underdetermined. We choose ridge regression over LASSO for this problem because we would like to determine a nonzero plus/minus rate for as many 5-person units as possible. If this rate happens to be close to zero, then that is acceptable, but we see no reason to promote sparsity as in LASSO. The second caveat is that while the usual ridge regression penalty is $||\vec{\beta}||_2^2$, in our case, following this procedure yields worse results than the average scoring rate model described above. Therefore, we change the penalty to $||\vec{\beta} - \vec{\beta}_0||_2^2$, where $\vec{\beta}_0$ is the vector of average scoring rates used in the earlier scoring model. We can implement this easily by considering

$\vec{\beta} = \vec{\beta}_0 + \vec{\beta}_1$. Then the ridge objective function is:

$$J_\lambda(\vec{\beta}_1) = \| \underbrace{(\vec{y} - X\vec{\beta}_0)}_{\vec{y}'} - X\vec{\beta}_1\|_2^2 + \frac{\lambda}{2}\|\vec{\beta}_1\|_2^2.$$

Passing $\vec{y}'$ and X to a ridge regression solver then yields, for a fixed value of λ, a minimizer $\vec{\beta}_1$. We use 10-fold cross-validation on the training set to determine an optimal value of λ; we then rerun the ridge regression on the entire training set using this optimal λ. This yields $\vec{\beta}_1$, which we add to $\vec{\beta}_0$ to obtain the scoring rate model. Of course, this procedure is repeated for each team.

Using ridge regression, we improve our training set performance, as displayed in the following confusion matrix: $\begin{bmatrix} 509 & 194 \\ 197 & 330 \end{bmatrix}$. The overall accuracy is now 0.682. We also see a slight improvement in test set performance as display in the left-most table in Table 2.4, as we are now correctly predicting 12/15 or 80% of the playoff winners. Among the models developed in this chapter, the ridge regression model is the best. Again, two of the incorrect predictions are for series that were decided in seven games.

Support Vector Regression. The next scoring model we consider is support vector regression (SVR) with a radial basis function kernel. For team i, we extract from the training data all rows and columns corresponding to 5-person units from team i. This yields, for each team, a training matrix with approximately 1500-2500 rows and exactly N_i columns. We fit one SVR model to each training matrix. Then, when simulating a game, we use this SVR model to predict the change in point differential generated by a particular 5-person unit over a particular stretch of time.

Test set results for the SVR model are given in the central table in Table 2.4. Because this model is more computationally intensive than the prior models, we simulated each NBA playoff series 10 times rather than 100 times. Overall, we see that only 7/15 or 46% of series winners have been predicted correctly.

Nearest Neighbor Regression. The final scoring model we consider is a k-nearest neighbor regression model with $k = 3$. We train this model on the same

Winner	Margin	Prob.	Winner	Margin	Prob.	Winner	Margin	Prob.
GS	1.74	0.78	GS	2.50	0.90	GS	3.40	1.00
Hou	0.44	0.57	Hou	3.20	0.90	Hou	2.60	0.90
SA	0.42	0.54	LAC	0.80	0.90	LAC	1.40	0.90
Por	0.29	0.56	Por	1.70	0.90	Por	1.00	0.70
GS	0.32	0.53	Mem	0.30	0.90	GS	0.80	0.50
Hou	0.01	0.53	Hou	2.10	0.90	Hou	1.50	0.70
GS	0.88	0.63	Hou	0.30	0.90	Hou	0.20	0.60
Atl	2.15	0.82	Bkn	2.50	0.90	Bkn	2.40	1.00
Cle	2.07	0.88	Cle	0.80	0.90	Cle	2.50	0.80
Chi	1.11	0.71	Mil	0.80	0.90	Mil	1.50	0.70
Tor	0.88	0.64	Was	1.80	0.90	Was	0.30	0.50
Atl	1.36	0.72	Was	3.20	0.90	Was	0.30	0.60
Cle	1.04	0.70	Chi	2.90	0.90	Cle	2.00	0.80
Cle	0.31	0.54	Atl	1.90	0.90	Cle	0.10	0.50
GS	0.16	0.51	GS	2.70	0.90	GS	0.10	0.50

Table 2.4: Test set results for ridge regression (left, 80% accuracy), support vector regression (center, 46% accuracy), and k-nearest neighbor regression (right, 66% accuracy). Note that the ridge regression scoring rate model results in a correct prediction for 12 out of the 15 playoff series; this is the best model considered in this chapter. For the order of the playoff series and true winners, please see Table 2.3.

set of matrices used to train the SVR model. Playoff predictions are given in the right-most table in Table 2.4. In situations where both teams won 5 of the 10 simulated series, we chose the team whose expected margin was positive. Overall, we see that 10/15 or 66% of series winners have been predicted correctly.

2.4.3 Additional Model Evaluation and Usage

To assess whether our test set prediction accuracy is meaningful, we have built three "box score" models. These models select—as a playoff series winner—the team that has (i) scored the most points in the regular season, (ii) achieved the best regular season winning percentage, and (iii) achieved the highest playoff seeding. Respectively, these models correctly predict 8/15, 11/15, and 12/15 of the playoff series' winners. Of course, our model is more complex than these box score models; naturally, we should expect our model to be capable of answering more complex questions than a box score model is capable of answering.

Our model is particularly well suited to answer "What if?" questions involving player/lineup usage. For example, the model can be used to assess the impact of a player being injured. Atlanta's Kyle Korver, one of the best three-point shooters in the NBA, was injured and did not play after the first two games of the playoff series against the Cleveland Cavaliers. From the next to the last row of Table 2.4, we see that the continuous-time Markov chain with ridge regression scoring rate predicts that Cleveland should win the series against Atlanta with a probability of 0.54 and a margin of less than one game (specifically, 0.31 games). These results assume that the usage of players mirrors that of the regular season, i.e., that Korver is healthy and able to play. As a test, we have removed from the Atlanta Hawks' transition matrix any 5-person lineup that involves Korver. Rerunning the simulation, we now find that Cleveland should win the series with a probability of 0.79 and a margin of almost 2 games (specifically, 1.72 games). This is closer to the real result, a 4-game series sweep by Cleveland.

While we have simulated the effect of a player not being to play at all, we note that we can also simulate more subtle scenarios such as (i) a player only being able to play a limited number of minutes per game, or (ii) a coach making a conscious decision to use particular lineups more often against a given opponent.

We view our model as a modular component to be incorporated into (rather than to replace) models that involve traditional predictors such as those used in the box score models above. Our best model uses ridge regression to infer the scoring rate for each 5-person unit, but completely ignores informative data such as who actually won each regular season game. In future work, we seek to use this information to generate improved predictions for the outcomes of games.

2.5 Conclusion

Given the simplicity of the model employed, our results are encouraging. There are several clear directions in which the model can be generalized and improved. First, at the moment, we are using a basic frequentist procedure to infer the transition rates of the continuous-time Markov chain. In ongoing work, we seek to compare this procedure against more sophisticated techniques such as variational Bayes and particle-based Monte Carlo inference (Opper and Sanguinetti 2007; Hajiaghayi *et al.* 2014). Second, the continuous-time Markov chain assumes that the holding time in each state has an exponential distribution. We seek to generalize this to a distribution that more accurately models the data; this will yield a semi-Markov process as in Cervone *et al.* (2016). While we have tested nonlinear regression models such as SVR, we have not conducted extensive cross-validation studies to find more optimal values of parameters for these models. For these nonlinear models, it may be beneficial to consider several years worth of training data. Finally, we expect that our scoring model can be improved by incorporating the effect of the opposing 5-person unit on the court.

Chapter 3

Modeling Basketball Substitutions from Play-by-Play Data

3.1 Introduction

Before basketball data became widely available through the Internet, people who were interested in understanding basketball had to rely on summary statistics to determine the performance of a player and a team lineup. Summary statistics are usually based on individual players, rather than on 5-person units (lineups). If one watches an NBA game, one would notice that a player's ability to score consists of not only the player's shooting accuracy, but also the lineup's effort to create a scoring opportunity.

Motivated by our previous work for the 2014-15 season, we use NBA play-by-play data for the 2015-16 season to approach the following questions:

1. How can we model time played by lineups for all 30 NBA teams based on substitutions?

2. How can we use such models to help predict game outcomes?

3. What methods can we use to increase our model's flexibility and accuracy?

We seek continuous-time stochastic models to account for lineup substitutions for all NBA teams, and we create team-specific scoring models to simulate games.

3.2 Related work

In this section we discuss work that has been done by others on basketball data and modeling.

Shirley (2007) modeled NBA basketball games using play-by-play data and discrete-time Markov chains with 30 states. Each state represents how a home team or away team gains possession of the ball. Their main goals were to compute in-game win probabilities for a home team and the change in win probability as the number of possessions changes. They found that, regardless of number of transitions, an average home team has a 61-65% chance of winning the home game. However, the win probability changes when there are fewer transitions left in a game, and the probability changes the most when the home team is leading. Their results also showed a good estimate of the teams' win percentage ($R^2 = 0.935$) for the 2003-04 season. However, the results were based only on in-sample prediction due to insufficient data.

Štrumbelj and Vračar (2012) repeated Shirley's model for the 2007-08 season and obtained good estimates on teams' actual win percentages for both in-sample and out-of-sample data ($R^2 = 0.85$). However, the results suggested the model overestimated weaker teams. To improve upon Shirley's model, Štrumbelj included teams' summary statistics such as effective field goal percentage and compared the performance of his forecaster to others, such as Shirley's model, bookmaker odds mark, and an ELO rating system. Štrumbelj's Markov model obtained out-of-sample accuracies of 0.6896 and 0.7053 for the 2007-08 and 2008-09 seasons, respectively. He noted that in both seasons, bookmaker odds mark has better predictive accuracies of 0.7042 and 0.7106 while Shirley's method obtained accuracies of 0.6698 and 0.6853.

Shi *et al.* (2013), on the other hand, applied several classifier learners on box-score type data to predict NCAAB individual game outcomes. Their results showed that for the seasons between 2009 and 2013, artificial neural networks and naive Bayes yielded 70-74% out-out-sample accuracy. The authors observed that more training data did not result in a better model and that Naive Bayes, one of the simplest classifiers, performs well. Continuous-time stochastic models have been considered by Peuter (2013). He also built models based on possessions but team lineups are ignored. His results showed the competitiveness between his models and Vegas forecasts for the 2012-13 season. Liu (2007) used a Bayesian social relations model to investigate interactions between team players in relation to offense, defense and how they contributed to the outcomes of the 2004 and 2005 NBA finals. The results of her model gave fair descriptions of player performance and recognized some players' contributions in offense and defense not visible in traditional statistics.

On a finer scale, Shortridge *et al.* (2014) used spatial data to evaluate players' shooting effectiveness. They developed metrics such as spatial shooting effectiveness and points above league average, allowing them to distinguish two players who have nearly identical traditional statistics such as effective field goal percentage. They also determine a player's shooting effectiveness at different spots of the half-court. Cervone *et al.* (2016) used multi-scale semi-Markov models on spatio-temporal data to further investigate expected possession value (EPV) of players. Based on EPV, they measure shot satisfaction of players who could either pass the ball or attempt a shot. They compare EPVs of multiple players when they all start a possession in the same situation. The methods allow one to see how and why EPV changes.

Xin *et al.* (2017), treating a game as a series of networks, used a continuous-time stochastic block model to help cluster player types for NBA games and obtain estimates of the effectiveness of players at possession-related action, e.g., rebounding. Their model can reveal differences in team offense strategies. Fewell *et al.* (2012), in similar spirit, developed metrics to measure ball movement efficiency (ball consistently moves to better shooters) and the predictability of ball movement of a team. D'Amour *et al.* (2015) tried to quantify ball movement based on possessions modeled using discrete states, continuous-time Markov chain.

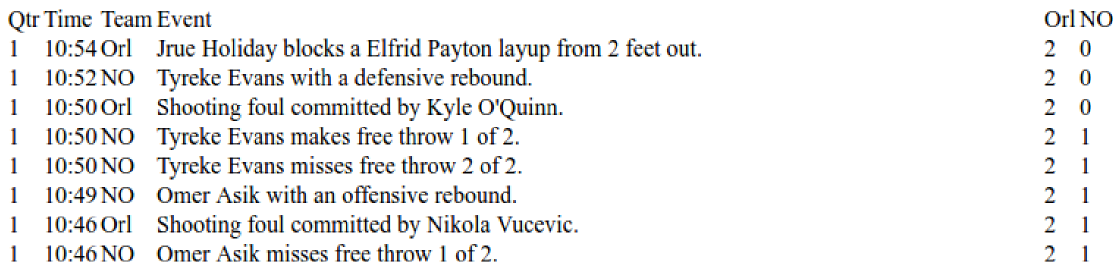
Using stepwise multiple linear regression models, Gómez *et al.* (2016) explored effects of player substitutions on point differences based on variables such as coach's decisions, player time in and out of the court, timeout, fouls, player field goal effectiveness, game location, and quality of opposition according to 1, 5, and 10 ball possessions during each quarter. They found that the impact of substitutions based on these factors was higher during the first and third quarters and lower during the fourth quarter. For all teams they studied, away teams made more substitutions than home teams, particularly when they were trailing. The home teams made significantly more substitutions when they were leading. Interestingly, this was not the case for the away teams.

Finally, the work closest to ours is by (Oh *et al.* 2015). Using probabilistic graphical models, they simulated games using features such as possessions and lineups and obtained good estimates for true teams' win percentage ($R^2 = 0.87$ on testing set of 369 games in the 2013-14 season). Using their model, one can answer question such as, "What is the predicted match outcome if I use these lineups against the opposing team?" One main conclusion is that changes in team lineup by either home or visiting team affects the dynamics and outcome of a game. Our work uses this as a staring point for modeling.

3.3 Data collection

Although sources such as NBA.com and ESPN provide volumnious statistics for teams and individual players, we found it difficult to obtain from such sites comprehensive statistics on the performance of 5-person units or lineups. To obtain this information, we mined 2015-16 regular season NBA play-by-play data from `knbr.stats.com`, supplemented by data on substitutions taking place between quarters from `Basketball-Reference.com`. For each play-by-play HTML page, we used `Beautiful Soup`, a Python package, to scrape the information needed from the text descriptions of particular plays. This is a more detailed description of how we mined NBA data.

To give an example of how the data appears after processing, we present Table



Qtr	Time	Team	Event	Orl	NO
1	10:54	Orl	Jrue Holiday blocks a Elfrid Payton layup from 2 feet out.	2	0
1	10:52	NO	Tyreke Evans with a defensive rebound.	2	0
1	10:50	Orl	Shooting foul committed by Kyle O'Quinn.	2	0
1	10:50	NO	Tyreke Evans makes free throw 1 of 2.	2	1
1	10:50	NO	Tyreke Evans misses free throw 2 of 2.	2	1
1	10:49	NO	Omer Asik with an offensive rebound.	2	1
1	10:46	Orl	Shooting foul committed by Nikola Vucevic.	2	1
1	10:46	NO	Omer Asik misses free throw 1 of 2.	2	1

Figure 3.1: Example of a play-by-play game data

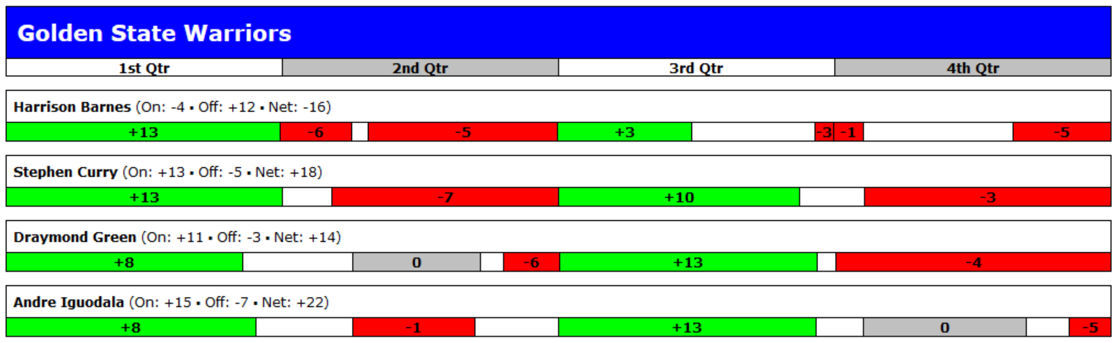


Figure 3.2: Example of plus/minus of players for four quarters of a game.

3.1. Each row of this data set corresponds to a group of 10 players who played on the court for a positive amount of time before at least one substitution was made by either team. Columns 1-3 record the date of the game and the identities of the home and visiting teams. Columns 4-8 record the identities of the five players on the court for the home team, while columns 9-13 record the same information for the visiting team. Column 14 contains the number of seconds this group of 10 players (5 from each team) played just before one substitution was made by either team. Columns 15-17 record the number of play-by-play events that have occurred for the home, visiting, and both teams since the last substitution. Columns 18-19 record the home and visiting scores at the time just before the substitution was made[1].

Column 20, the last column, records the change in point differential. Let the current home and visiting scores (recorded in columns 18-19) be H_i and V_i, respec-

[1]The table actually consists of more columns than what is presented. For presentation purposes, we omit the columns that do not contribute to our results.

1 (Date)	2 (Home Team)	3 (Visiting Team)	4 (Home Player 1)	5 (Home Player 2)	6 (Home Player 3)	7 (Home Player 4)	8 (Home Player 5)	9 (Visiting Player 1)	10 (Visiting Player 2)	11 (Visiting Player 3)	12 (Visiting Player 4)	13 (Visiting Player 5)	14 (Seconds Played)	15 (Home Events)	16 (Visiting Events)	17 (Total Events)	18 (Home Score)	19 (Visiting Score)	20 (Δ_i—see Eq. (3.1))
20151027	GS	NO	425	105	34	166	54	160	376	107	102	353	227	13	13	26	10	9	1
20151027	GS	NO	425	105	34	166	54	160	401	107	102	353	199	26	11	24	25	18	6
20151027	GS	NO	425	105	34	166	214	160	401	107	102	353	0	0	11	0	25	18	0
20151027	GS	NO	425	105	34	166	214	160	401	107	102	5	71	5	6	11	32	20	5
20151027	GS	NO	31	105	34	166	214	160	401	107	102	5	70	10	10	14	34	23	-1

Table 3.1: Sample rows of data frame produced by scraping play-by-play data.

tively. Then the change in point differential Δ_i is

$$\Delta_i = (H_i - V_i) - (H_{i-1} - V_{i-1}), \tag{3.1}$$

with the understanding that the initial scores are $H_0 = V_0 = 0$. This quantity is the "plus/minus" of the two 5-person units on the court. If we start with the first row of Table 3.1, we see that at the time the first substitution is made, $\Delta_1 = 1$, corresponding to home and visiting scores of 10 and 9, respectively. At the time the next substitution is made, the score is 25 to 18 in favor of the home team. The differential is $(25 - 18) - (10 - 9)$, and so the *change* in differential is six, i.e., $\Delta_2 = 6$.

Viewed from the point of view of the home 5-person unit, this means that even though the unit gave up 9 points on defense, the unit was able to score 15 and thereby increase their team's lead. We see that the Δ_i value encapsulates both the offensive and defensive performance of a particular 5-person unit. It is better to score only 3 points on offense and yield 0 points on defense than it is to score 20 points on offense while yielding 25.

3.4 Model

Our model consists of two parts: (i) a model for substituting one 5-person unit by another and (ii) a model for how each 5-person unit contributes to the overall score of the game.

3.4.1 Substitutions

At any fixed instant of time, a basketball game is played between 5 players from the home team and 5 players from the away team. Each game is 48 minutes, consisting of four 12-minute quarters. The match begins with a jump ball, in which the two tallest players from both teams try to help their team gain possession of the ball. The player who obtains the ball either dribbles past the half-court or passes the ball to his teammate. The goal for both teams is to put the ball into the basket within 24 seconds. Each basket made has a value of one, two, or three points, depending on different situations. Once a team scores a basket, the other team has possession of the ball. The team with the largest sum from the baskets made at the end of 48 minutes is the winner of the game. Figures 3.3 - 3.5 shows examples of substitutions and how a team scores.

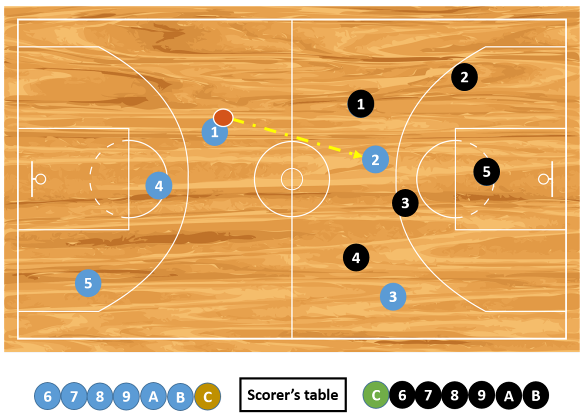


Figure 3.3: Basketball game between two teams: 5 vs. 5

During the game, a team can have multiple possessions of the ball. We consider the following scenario as two consecutive ball possessions by a team: A player misses a layup, and his teammate rebounds the ball and scores the basket. Unlike other team sports, multiple player substitutions of players are allowed in basketball. Coaches use substitution to rest players, avoid player foul trouble, put players who play with synergy together, devise new offense and defense strategy, etc. A substitution can occur at many possible times: after fouls, after timeouts, after out-of-bounds play, etc. Hence we build our substitution model using discrete-state,

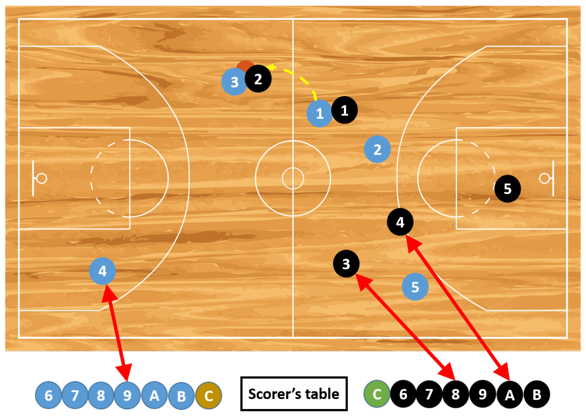


Figure 3.4: Example of players substitutions

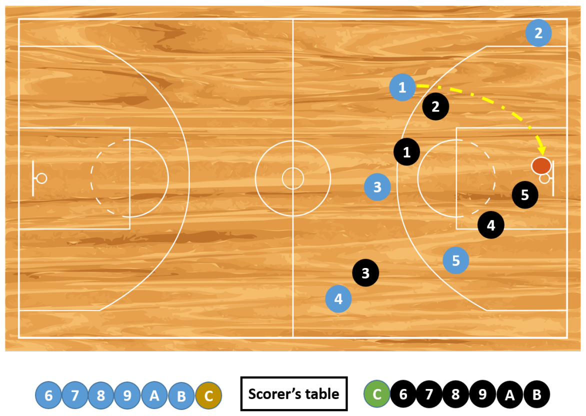


Figure 3.5: Example of how a team scores

continuous-time Markov chains (CTMC). Each state of the chains corresponds to a different 5-person unit/lineup. Table 3.2 is an example of different 5-person lineups. Figure 3.6 shows the flowchart of our basketball data modeling. Before continuing our discussion of basketball modeling, let us review some theory of the discrete-state, continuous-time Markov chain.

Theory of continuous-time Markov chains.

Consider a state space S and a process X_t satisfying the Markov property,

$$P\{X_t = y | X_r, 0 \le r \le s\} = P\{X_t = y | X_s\}.$$

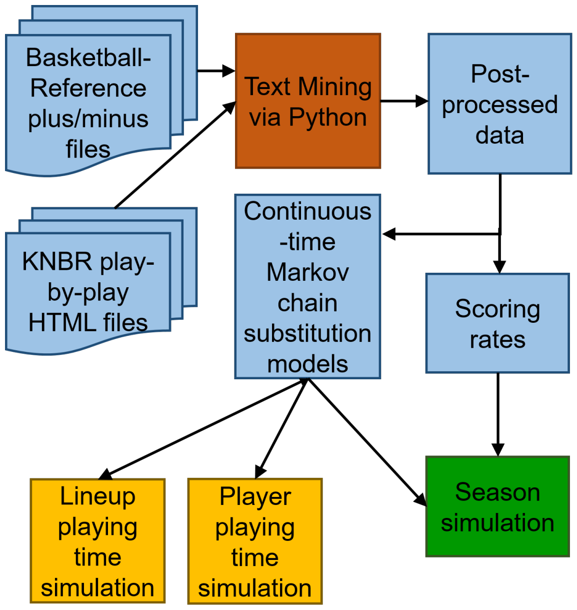

Basketball-Reference plus/minus files
KNBR play-by-play HTML files
Text Mining via Python
Post-processed data
Continuous-time Markov chain substitution models
Scoring rates
Lineup playing time simulation
Player playing time simulation
Season simulation

Figure 3.6: Flowchart of our basketball data modeling

State	Player 1	Player 2	Player 3	Player 4	Player 5
1	425	105	34	166	54
2	425	105	34	166	214

Table 3.2: Example of two different lineups/states. Even though lineup 1 and lineup 2 differs by one player, we treat them as different states in our model.

Suppose X_t is time-homogeneous,

$$P\{X_t = y | X_r, 0 \leq r \leq s\} = P\{X_{t-s} = y | X_0 = x\}.$$

For each x, $y \in S$, $x \neq y$ we define $\alpha(x, y)$ as the rate at which x jumps or moves to y Lawler (2006). The total rate for which the chain changes from state x is

$$\sum_{y \neq x} \alpha(x, y).$$

A time-homogeneous continuous-time Markov chain with rates α is a stochastic process X_t taking values in S satisfying

$$P\{X_{t+\Delta t} = x | X_t = x\} = 1 - \alpha(x)\Delta t + o(\Delta t) \tag{3.2}$$

$$P\{X_{t+\Delta t} = x | X_t = y\} = \alpha(y,x)\Delta t + o(\Delta t), \quad y \neq x. \tag{3.3}$$

Let $p_x(t) = P\{X_t = x\}$. Then, we can write 3.2 and 3.3 as a system of linear differential equations,

$$p'_x(t) = -\alpha(x)p_x(t) + \sum_{y \neq x} \alpha(y,x)p_y(t).$$

Suppose we have an initial condition, $p_x(0), x \in S$ and let A be the matrix whose (x,y) entry equals $\alpha(x,y)$ if $x \neq y$ and equals $-\alpha(x)$ if $x = y$.

If $\vec{p}(t)$ denotes the vector of probabilities, we can write the system as

$$\vec{p}'(t) = \vec{p}(t)A,$$

which has solution

$$\vec{p}(t) = \vec{p}(0)e^{tA},$$

assuming the initial condition $\vec{p}(0)$. Now let $p_t(x,y) = P\{X_t = y | X_0 = x\}$ and let P_t be the matrix whose (x,y) entry is $p_t(x,y)$. The system of differential equations can be written as a single matrix equation:

$$\frac{d}{dt}P_t = P_tA, \quad P_0 = I.$$

The matrix P_t is given by

$$P_t = e^{tA}.$$

We can use exponential waiting times to give an alternative description of the Markov chain. Suppose rates $\alpha(x,y)$ have been given. Suppose $X_0 = x$. Let

$$T = \inf\{t : X_t \neq x\},$$

be the (holding) time at which the process first changes state. The Markov property can be used to see that T must have the loss of memory property, and hence T must have an exponential distribution. By 3.2,

$$P\{T \leq \Delta t\} = \alpha(x)\Delta t + o(\Delta t).$$

For this to be true, T must be exponential with parameter $\alpha(x)$. The infinitesimal characterization of 3.3 can be used to check the probability that the state changes to y is exactly $\alpha(x,y)/\alpha(x)$.

To see T is exponential, consider $P(T > s + t | T > t)$. By definition

$$P(T > s + t | T > t) = \frac{P(T > s + t, T > t)}{P(T > t)}.$$

By the Markov property and time homogeneity, we can write the equation above as

$$\begin{aligned} P(T > s + t | T > t) &= \frac{P(T > s + t, T > t)}{P(T > t)}, \\ P(T > s | T > 0) &= \frac{P(T > s + t)}{P(T > t)}, \\ P(T > s) &= \frac{P(T > s + t)}{P(T > t)}, \end{aligned}$$

which simplifies to

$$P(T > s)P(T > t) = P(T > s + t).$$

The equation is only satisfied by an exponential function $e^{-\lambda t}$, where λ is the parameter. In Figure 3.7, we plot the empirical CDF and true CDF of our NBA data for all team lineups and observe that lineup substitutions, for the most part, follow the exponential distribution.

3.4.2 Simulation of CTMC: Alarm clocks

Now if we are given the rates, we want to know how to construct the discrete-state, continuous-time Markov chain. Let $\lambda_i = \sum_{j \neq i} q(i, j)$ be the rate at which the process leaves state i. Moreover, suppose $\lambda_i > 0$ and let $r(i, j) = q(i, j)/\lambda_i$ be the probability the chain goes to state j once it leaves i.

Computing the transition probability of the chain from the transition rates q. Consider the Chapman-Kolmogorov equations:

$$\begin{aligned} p_{t+h}(i, j) - p_t(i, j) &= \left(\sum_k p_h(i, k) p_t(k, j) \right) - p_t(i, j), \\ &= \sum_{k \neq i} p_h(i, k) p_t(k, j) + p_h(i, i) p_t(i, j) - p_t(i, j). \end{aligned}$$

$$p_{t+h}(i, j) - p_t(i, j) = \sum_{k \neq i} p_h(i, k) p_t(k, j) - (1 - p_h(i, i)) p_t(i, j). \tag{3.4}$$

Let $q(i,j) = \lim_{h\to 0} \frac{p_h(i,j)}{h}$ for $i \neq j$. Then, dividing both sides of 3.4 by h and letting $h \to 0$, we obtain

$$\begin{aligned}
p_t'(i,j) &= \lim_{h\to 0} \frac{p_{t+h}(i,j) - p_t(i,j)}{h} = \lim_{h\to 0} \frac{1}{h}\left[\left(\sum_k p_h(i,k)p_t(k,j)\right) - p_t(i,j)\right] \\
&= \lim_{h\to 0} \frac{1}{h}\left[\sum_{k\neq i} p_h(i,k)p_t(k,j) + p_h(i,i)p_t(i,j) - p_t(i,j)\right] \\
&= \lim_{h\to 0} \frac{1}{h}\left[\sum_{k\neq i} p_h(i,k)p_t(k,j) - (1 - p_h(i,i))\, p_t(i,j)\right] \\
&= \lim_{h\to 0} \frac{1}{h}\left[\sum_{k\neq i} p_h(i,k)p_t(k,j) - \sum_{k\neq i} p_h(i,k)p_t(i,j)\right] \\
&= \left[\sum_{k\neq i} \lim_{h\to 0} \frac{p_h(i,k)}{h} p_t(k,j) - \sum_{k\neq i} \lim_{h\to 0} \frac{p_h(i,k)}{h} p_t(i,j)\right] \\
&= \sum_{k\neq i} q(i,k)p_t(k,j) - \sum_{k\neq i} q(i,k)p_t(i,j) \\
p_t'(i,j) &= \sum_{k\neq i} q(i,k)p_t(k,j) - \lambda_i p_t(i,j)
\end{aligned}$$

Let

$$Q(i,j) = \begin{cases} q(i,j) & \text{if } j \neq i \\ -\lambda_i & \text{if } j = i. \end{cases}$$

Then, we can write the differential equation as $p_t' = Qp_t$, which has solution $p_t = e^{Qt}$.

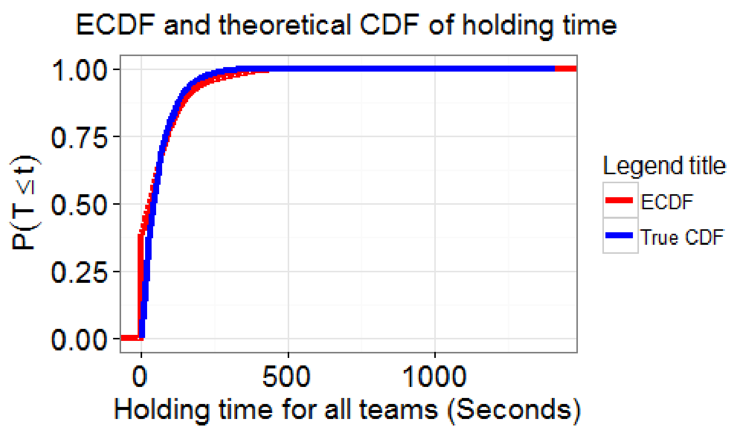


Figure 3.7: Holding times for all NBA lineups for the 2015-15 regular season. Notice the empirical CDF and the theoretical CDF are similar. Note the holding times for each individual team's lineups might not be exponential.

We apply this knowledge and construct one Markov chain for each of the 30 NBA teams; let M_i denote the transition rate matrix of the Markov chain for team i. The Markov chain for team i is completely specified by M_i. Each state of M_i is a different 5-person unit that appears in the training data for team i. Let N_i be the number of states for M_i; using the entire 2015-16 regular season and playoffs, we obtain the following lineup counts in Table 3.3.

In Figure 3.8 we show an example of different 5-person units/states used by the Golden State Warriors on October 27, 2015 against the New Orleans Pelicans:

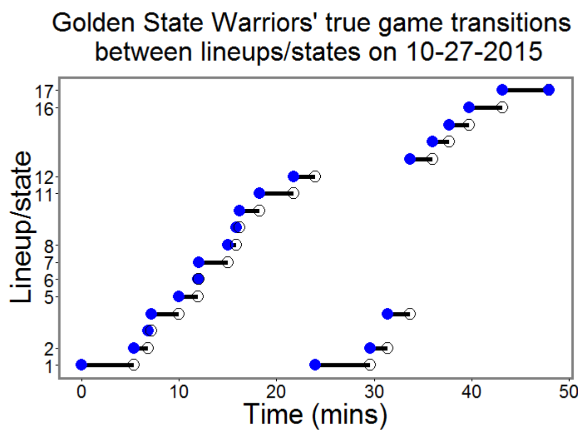


Figure 3.8: Transition of lineups/states of Golden State Warriors in true match against New Orleans Pelicans on 10/27/15. Most lineups only played once.

Team	**Total number of lineups**	**Team**	**Total number of lineups**
ATL	414	TOR	336
CHI	460	MEM	625
GS	446	IND	420
BOS	440	LAC	423
DET	229	NYK	373
HOU	442	CLE	436
LAL	359	DEN	451
MIA	434	PHI	645
MIL	404	SA	565
BKN	452	NOP	554
ORL	437	WAS	495
PHX	588	CHA	323
POR	282	MIN	342
SAC	432	DAL	540
OKC	419	UTA	482

Table 3.3: Total number of lineups used by each team. Lineups for playoff teams are included. Since each team has about 18 players, there are $\binom{18}{5} = 8,568$ lineup choices if we do not use the data.

For each team i, we infer the $N_i \times N_i$ transition rate matrix M_i using the MLE (maximum likelihood estimate) as derived in Section 3.7 and also in Guttorp (1995); Metzner *et al.* (2007a); Konstantopoulos (2006):

$$\widehat{M_i^{j,k}} = \frac{\#(j \to k)}{\alpha(j)}. \tag{3.5}$$

Here $\widehat{M_i^{j,k}}$ is the estimate for the (j,k)-th entry of M_i, $\#(j \to k)$ denotes the number of observations of a transition from state j to state k, and $\alpha(j)$ denotes the total time spent in state j. All these values can be computed using play-by-play data.

To validate this model's performance on the training set of 2015-16 NBA games, we simulate 8200 games for each team. We count the number of substitutions made by each team and divide by 100 to obtain a Monte Carlo estimate for the total number of substitutions made by each team in one full season of play. The simulation follows standard algorithms for sampling from a continuous-time Markov chain. Assume the system is currently in state j. We then simulate exponentially distributed random variables with rates given by row j of the transition rate matrix. The minimum of these samples gives us both the time spent in state j as well as the identity of the new state k to which we transition. We initialize the simulation using the most common 5-person unit for each team, and we terminate the simulation once it reaches 2880 seconds, corresponding to a regulation-length NBA game.

In Fig. 3.9, we plot the true and simulated times played by each of the 5-man units across all 30 teams (left panel, Pearson correlation of 0.9809) and the true and simulated times played by NBA players (right panel, Pearson correlation of 0.9955). All times are in seconds. The reported correlation is for the raw data.

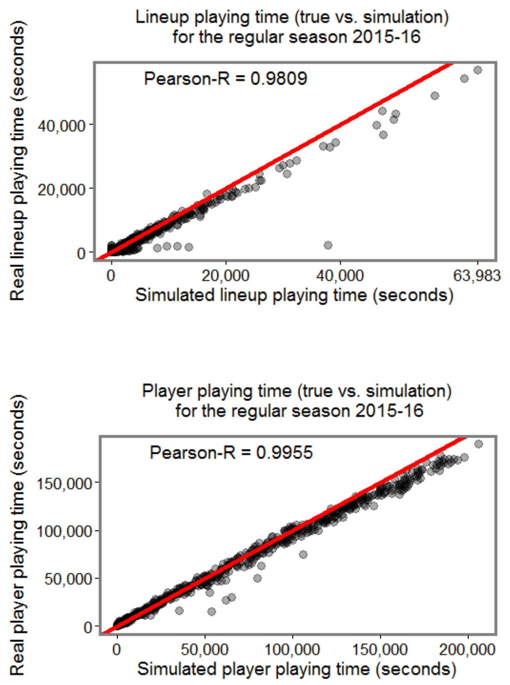


Figure 3.9: True vs. simulated lineup and player playing time for the 2015-16 regular season. Although there is a good in-sample fit between the simulation and true data, we see the model over-simulates many lineups and players in both plots. See Table 3.4 for an example of a problem using all states.

Overall, the in-sample fits between true and simulated unit and player times indicate that our model is a reasonable start for modeling substitutions and 5-person unit playing times. Clearly, further research is necessary to improve the fit and develop a more predictive model of 5-person unit times. An obvious area for improvement is to model the number of fouls committed by each player on a team. Because a player must leave the game immediately after committing a sixth

Lineup	**Player1**	**Player 2**	**Player 3**	**Player 4**	**Player 5**
1	Harrison Barnes	Stephen Curry	Festus Ezeli	Draymond Green	Klay Thompson
2	Harrison Barnes	Stephen Curry	Draymond Green	Andre Iguodala	Klay Thompson
3	Leandro Barbosa	Ian Clark	Kevon Looney	Brandon Rush	Jason Thompson
4	Harrison Barnes	Stephen Curry	Draymond Green	Andre Iguodala	Klay Thompson
5	Andrew Bogut	Stephen Curry	Draymond Green	Brandon Rush	Klay Thompson
6	Andrew Bogut	Stephen Curry	Draymond Green	Andre Iguodala	Klay Thompson

Table 3.4: Only 6 lineups are used in one simulated game. Note: Kevon Looney and Jason Thompson played 5 and 28 games, respectively, in season 2015-16. Thompson was traded to the Toronto Raptors in March. The appearance of states involving Jason Thompson and Kevon Looney is unavoidable if we use the entire regular-season to build the model.

foul, a player is more likely to be substituted out of the game as he accumulates more fouls. This affects how the player plays defensively, which in turn affects the dynamics of his lineup. Another idea is to allow the Markov transition rates to depend on how many minutes remain in the game and the game score; towards the end of blowout games, where one team leads another by a large margin, we see teams rest their regular players in favor of bench players.

3.4.3 Scoring rates

In the second part of our model, we consider changes in point differential (plus/minus) rate for each 5-person unit. We refer to this as our scoring model, even though point differential incorporates offensive and defensive performance, as described in Section 3.3. When simulating the CTMC substitution model, we multiply the time spent in a state by its associated scoring rate. Summing the results across a 48-minute simulated game, we obtain an aggregate point differential. We initialize the system with a starting lineup, which we sample from the starting lineup distribution. To simulate a matchup, we simulate each team's aggregate point differential independently. The team with the larger value is the winner of the game. We also incorporate home-court advantage for the home team.

There are many ways to model scoring rates. The following are the methods we have attempted.

Singular value decomposition (SVD). Consider the scoring matrix S whose

$(i, j)-$th entry corresponds to time played in game i by the jth lineup. In addition, consider Δ the vector of point differentials for all games the team played. Since there are more columns (lineups) than rows (games), the linear system $S\vec{y} = \Delta$ is underdetermined. To obtain the scoring rate $\vec{y}$, we apply the SVD to find the pseudoinverse $S^{\dagger}$. Multiplying $S^{\dagger}$ and Δ gives us the scoring rate $\vec{y} = S^{\dagger}\Delta$.

Weighted observations. Intuitively, most recent games have more effect on how two teams play against each other. To reflect this in our scoring rate model, we consider weighting observations by dates. We first determine the day gap between the date we would like to simulate the game and older game dates. This yields a range of day differences greater than one. Most recent game dates will receive the most weight. For example, we can assign a weight of 1 to the games played in the last 30 days, and a weight of 0.25 to games played past a month. Incorporating the weight matrix W and using the same scoring matrix and point differentials vector $\vec{\Delta}$ above, we obtain a linear system

$$\begin{aligned} S_{\text{time}}\, \vec{y}_{\text{rates to be determined}} &= \Delta_{\text{actual plus/minus rates}}, \\ S^T W_{\text{weights}} S\vec{y} &= S^T W_{\text{weights}}\Delta, \\ \vec{y} &= (S^T W S)^{\dagger} S^T W \Delta, \end{aligned}$$

where $(S^T W S)^{\dagger}$ is the pseudoinverse of $S^T W S$.

Exponentially weighting observation. Instead of choosing weights to weigh observations based on dates, we can weigh observation based on an exponential function. For example, choosing a date, we choose to weigh observations within the 30-day period by 1 and others by the exponential function $e^{-(\text{other days}-30)/30}$. This method will generate tapered weights if we have observations from multiple seasons or if we have many games much older than 30 days.

3.5 Simulation and Results

While constructing the CTMC matrices for all teams, we observed that not all states transition to other states. In basketball this occurs, for example, if a lineup is played till the end of the game. We call these states “dead-end” states or absorbing states. See Figure 3.10 for an example. If M is a CTMC matrix, a dead-end state corresponds to a row i for which

$$\sum_{j \neq i}^{n} M_{ij} = 0.$$

In a simulation, an absorbing state is forced to play for the remaining of the game once it begins to play. This situation is unrealistic in a true basketball game. To solve this problem, we develop three strategies: reverse gear, reduced states, and dead-end states only at the end of the game. We briefly discuss the strategies in the following subsections.

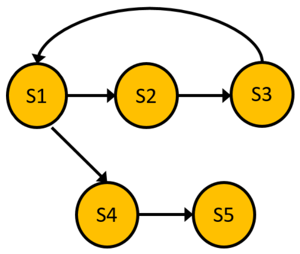


Figure 3.10: Example of state transitions containing an absorbing state $S5$. If we start with state $S1$, then either it proceeds to states 2 and 3 before returning or it jumps to state 4 and ends at state 5, which leads to nowhere.

3.5.1 Dead-end strategy 1: Reverse gear

In simulation, when we run into a dead-end state, we back track. This means we can either return to a previous lineup, or use one of the starting lineups sampling from its distribution.

3.5.2 Dead-end strategy 2: Reduced state and dead-end state removal

Earlier plots of simulated and real lineup playing times show over-simulated playing time for many team lineups. The scenario reflects a lineup that played less than T minutes in a season but now plays more minutes in simulation. To implement a reduced state strategy, we set a threshold time. Then, we proceed to remove rows and columns corresponding to states/lineups that played less minutes than the threshold during the season. This results in many states whose rows sum to 0. We also remove these state rows and their corresponding columns. In Table 3.5, we show lineup counts using reduced state models. Two minutes is our threshold. In Figure 3.11, we show an example of reduced-state strategy applied on a 5-state chain.

Team	ATL	CHI	GS	BOS	DET	HOU	LAL	MIA	MIL	BKN
Count	203	265	200	208	146	249	217	228	248	242
Team	ORL	PHX	POR	SAC	OKC	TOR	MEM	IND	LAC	NYK
Count	247	289	148	253	216	169	277	210	178	238
Team	CLE	DEN	PHI	SA	NOP	WAS	CHA	MIN	DAL	UTA
Count	227	252	353	247	334	250	184	189	273	275

Table 3.5: Number of lineups remaining after 2-minute threshold filter. Observe the reduced number of lineups compared to the counts in Table 3.3

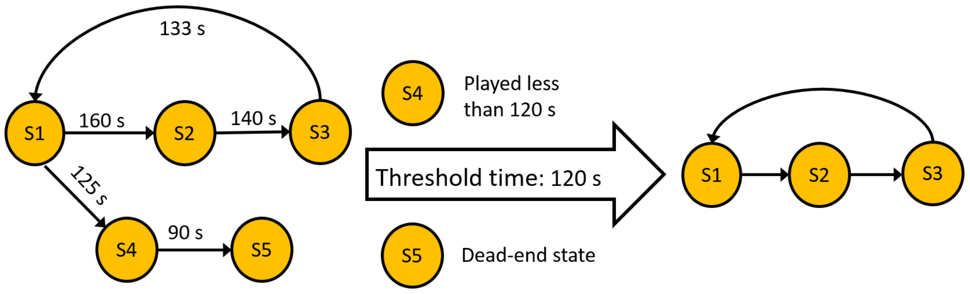


Figure 3.11: An example of reduced-state method: We remove states $S5$ and $S4$ because $S5$ is a dead-end state and $S4$ played less than the threshold time: 120 seconds.

3.5.3 Dead-end strategy 3: Stay in dead-end state

Every basketball game ends with a lineup/state that never transitions to another state. To capture this pheonomenon of the game, we allow a dead-end state/lineup to play if it was the last lineup played before the end of the game. Otherwise, we transition to a staring lineup.

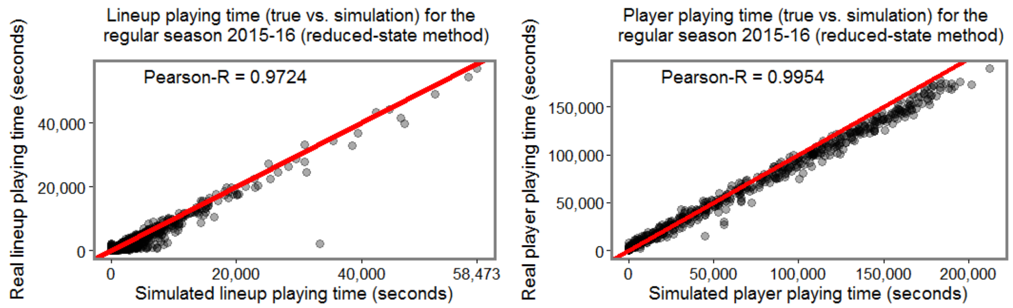


Figure 3.12: In-sample lineup and player simulated and true playing time for reduced-state method. The threshold time T is 2 minutes.

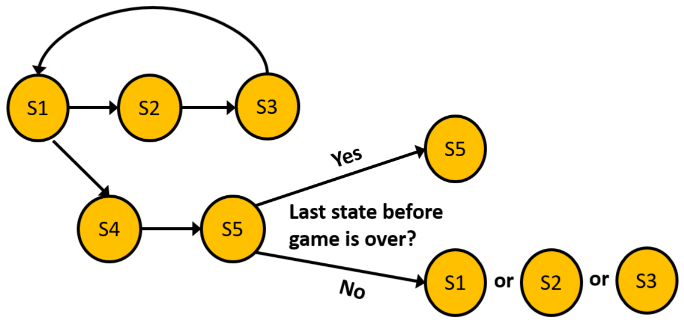


Figure 3.13: An example of dead-end strategy. Allow simulation to stay in the dead-end state if it is the last state transition, or else, transition from the dead-end state to one of the starting states.

3.5.4 Results

We train using the first 75 NBA regular season games and test using the remaining games 75-82. We implement each of the strategies for dead-end states, and we also implement various scoring rate models. Table 3.6 shows different combinations of strategies, together with in-sample and out-of sample accuracies. Overall, our results are as predictive as other work described in Section 3.2. We do not add home-court advantage in the training set but do so for the test set to obtain out-of-sample accuracy. We do so by taking a range of possible home-court advantages and adding each one at a time to the home teams' scores, to determine which value yields the best out-of-sample accuracy. Notice that if the simulated results are far from the true outcomes, in particular for the home teams, large values of home-court advantage are needed for best out-of-sample accuracy. Because only a single value of home-court advantage is picked, it is the league home-court advantage. In Table 3.6, we see unrealistic values of home-court advantage, though each yields best out-of-sample accuracy for the particular strategy we used. In fact, Ribeiro *et al.* (2016) observed over the seasons 2002-2014 play-by-play data that the home-court advantage is about 3.3 ± 0.1 points, averaged over all seasons.

In Figure 3.14 we find the absolute difference between margins of victory between simulated and true games of the regular season (82 games). In Figure 3.15, we show the optimal home-court advantage value that yields the best in-sample accuracy, which reflects improved matching between simulated and true margins of victory.

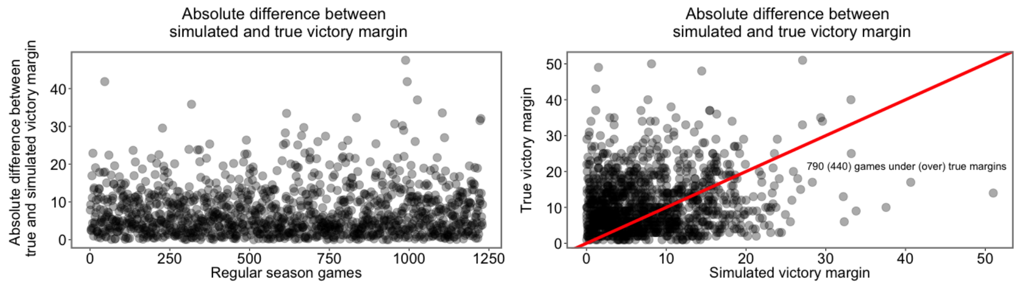


Figure 3.14: Margin of victory for all 1,230 season games. On the left, we plot the absolute difference between simulated and true margin of victory for each game. On the right, we plot simulated and true margins of victory.

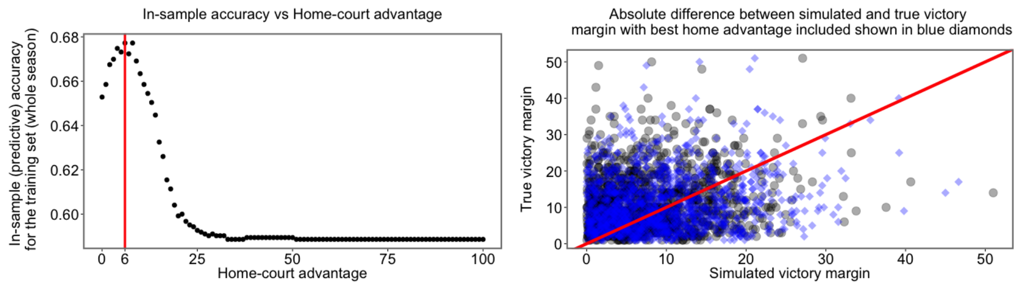


Figure 3.15: On the left, we plot in-sample (training) accuracy as function of home-court advantage. On the right, we plot simulated and true margin of victory with and without optimal home-court advantage shown in the left plot.

We also explore ways to improve in-sample-accuracy based on scoring methods. In Figure 3.16, we plot in-sample accuracy as a function of home-court advantage based on different scoring methods used to simulate the NBA season.

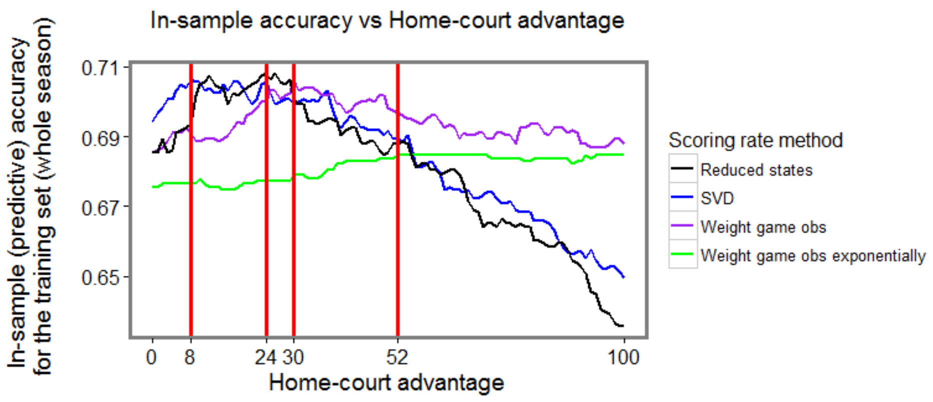


Figure 3.16: In-sample (training) accuracy of whole regular seasons based on different scoring-rate methods. Best home-court advantages that yield the best in-sample accuracies for the methods are plotted in the vertical lines.

Simulation Strategies				**Results**		
Scoring rates against particular opponent	Weighting by dates	scoring rates computed based on games	Reduced states	Home-court advantage	In-sample accuracy	Out-sample accuracy
				26	0.71	0.67
✓				5	0.61	0.51
		✓		46	0.71	0.64
✓		✓		6	0.72	0.65
✓	✓	✓		20	0.70	0.61
✓		✓	✓	23	0.75	0.70

Table 3.6: Check marks correspond to strategies used. Home court advantage is chosen to minimize in-sample error; For each team, training set is the first 75 games, test set is the remaining 7 games. 30 days is used in ‘weighting by days’ strategy.

We also experiment with various training and test splits on playing time of team lineups and players to see our model’s performance. In Figure 3.18, we plot average relative error of all 30 NBA teams for different training and test set splits. Given predicted and true values, we compute relative error using the 2-norm:

$$\text{Relative Error}_2 = \frac{\|\hat{Y} - Y\|_2}{\|Y\|_2}.$$

The blue and red curves represent training and test relative error, respectively. The plots show that increasing the size of the training set for simulating player and lineup playing time does not improve predictive accuracy. The results suggest that we should cap our training set size at 12 games. We also have observed that teams in the 2015-16 season did not consistently use the same lineups throughout the season. This makes exploring other ways to split data into training and test sets a worthwhile effort.

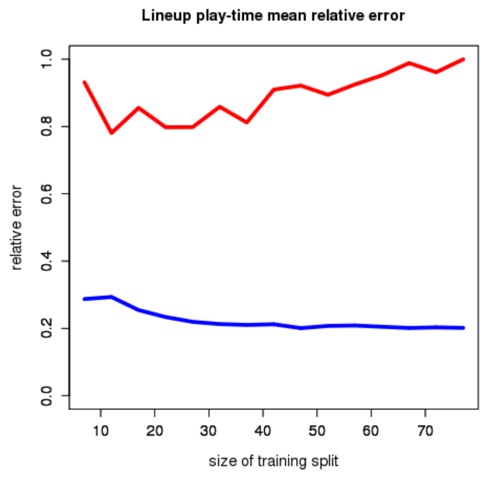


Figure 3.17: Average relative errors for all 30 NBA teams for different size of training sets for simulating playing time for both team lineups

In Figure 3.19, we plot in-sample and out-of-sample accuracy for an 82-game season based on various training and test set splits. The idea is that we train on the first seven games, predict games 8-12, train on the first 12 games and predict on the next set of five games, and so on. The last training set consists of the first 77 games; here we predict the last five games of the season. We do not see steady improvement in test accuracy as we increase the size of the training set as there are as many peaks as there are valleys.

3.6 Conclusion and future work

Compared to our previous work for the 2014-15 season (Bhat *et al.* 2015), we have better captured the dynamics of a basketball game. We make better predictions using various strategies in simulation and scoring models. Our results are similar to those mentioned in Section 3.2. This is very encouraging given the simplicity of our continuous-time Markov model. Moreover, our model can answer hypothetical questions such as, “What is a game outcome if certain players are unable to play due to injury or penalty from the league?” and “What is

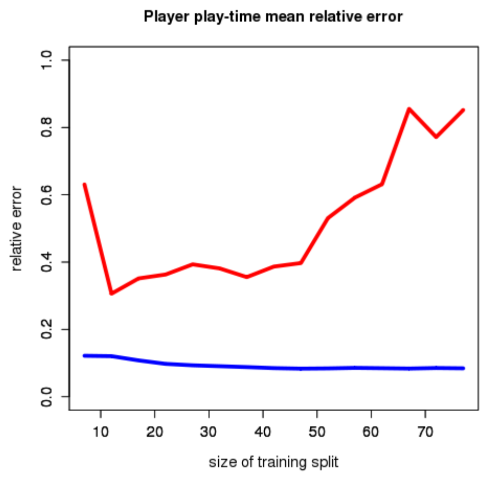


Figure 3.18: Average relative errors for all 30 NBA teams for different size of training sets for simulating playing time for both team lineups and players. The blue and red curves correspond to relative errors of training and test sets.

the winning probability of a team if it were to play another team 100 times?" To make our model more robust, we can apply the model to various seasons, assess predictive accuracies, and update our methods in addition to the options we described in Section 3.4.1. Another improvement to our model would be to find optimal parameters for values such as reasonable home-court advantage and weights on observation to maximize out-of-sample accuracy.

Several future goals to address some of the issues we have observed are the following:

- **Improve the fit between predicted and actual lineup times.** In Figures 3.7 and 3.9, there are significant discrepancies between the predicted and true lineup playing times despite the appearance of an exponential tail. One potential solution is to generalize our Markov model by considering a semi-Markov model, allowing an arbitrary holding time distribution (Barbu and Limnios 2009). Another possibility is a hidden Markov model. In a hidden Markov model, a sequence of observed data is mapped to a sequence of labels. Then, a probability distribution is computed over possible sequence labels. The best label sequence is chosen (Rabiner 1989). Note that this will require developing new methodology. Current methodology in R (using

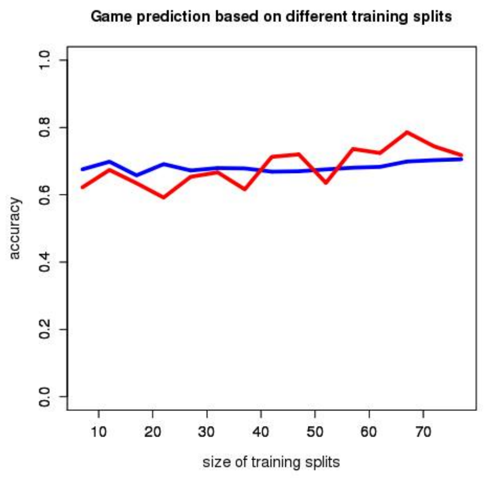


Figure 3.19: Season game prediction. Observe our predictive accuracy is as competitive as other models. The best out-of-sample accuracy seems to be when the training set consists of the first 68 games. However, the out-of-sample accuracy has several peaks and valleys.

available CRAN packages) only allows holding times to come from a set of parametric distributions. For example, in the package `SemiMarkov`, the set of parametric distributions where the holding times can come from include exponential, Weibull or exponentiated Weibull.

- **Improve overall basketball fidelity of the model.**
  - **Substitutions should depend on various factors**. Substitutions are made based on how much time remains in the game, score differential (i.e., whether we are in garbage time), and how many fouls each player has. An example is Deshpande and Jensen (2016), who modeled team winning probability as a function of its lead and time remaining in the game based on the impact of team players.
  - **Plus/minus rates should be stochastic**. Every lineup has good days and bad days, and basketball has fundamental randomness. Gabel *et al.* (2012); Ribeiro *et al.* (2016) observed that scoring rates are not constant and are decreasing over time. Gabel showed that the distribution of scoring time intervals has an exponential tail with little correlation to

scoring events. Moreover, scoring is also subjected to anti-persistence and restoring force, i.e., a team is likely to score after the opposition has just scored, and a team tends to coast after achieving a potentially certain lead, which allows the opponent to score more.

  - **Lineup scoring rates are nonlinear**. In a simulation we initialize the scoring rate of a lineup by randomly sampling from its scoring rate distribution. If the lineup is substituted out and plays later in the game, we randomly sample again from the lineup's scoring rate distribution. Our model should reflect that a poorly rested lineup will play at a different level than the same lineup at the start of a game.
  - **Home-court advantage should vary among teams**. Jones *et al.* (2007) has shown home-court advantage is not fixed. On average, home teams are expected to win 60% of the time; home advantage is front loaded in the first quarter and less in the remaining quarters. In the 2002-2004 seasons, Jones observed that home teams who won but trailed after the first quarter lost their home advantage. These teams regained it after the second quarter despite trailing. However, Ribeiro *et al.* (2016), taking a microscopic approach similar to Jones, found a diminishing effect of home-court advantage over the season. This suggests two approaches we can incorporate in our model:
    * Adopt the value of league average home-court advantage, rather than picking the best home-court advantage that maximizes our test accuracy.
    * Simulate games without home-court advantage to reflect reduced importance of home-court advantage.

- **Re-do the model starting from a possession-based framework**. Instead of simulating both teams independently, which is a key feature of our current approach, we could choose to simulate the entire game possession by possession. Kubatko *et al.* (2007) discussed events that allow a team to hold on to the ball. This can be a starting point for us to account for lineup substitutions between possessions. In addition, we can figure out all the different ways a possession can end for each team, and simulate these outcomes one possession at a time. In this way, we can use more aspects of the play-by-play data such as rebounds, blocks, out-of-bounds, steals, and fouls.

- **Find a method to quantify the change in intensity from regular season to playoff games**. During almost every postseason, basketball enthusiasts tend to notice that teams play with a different level of intensity during the playoffs as compared with the regular season. A method to describe playoff intensity has been developed by Swartz *et al.* (2011). They studied all possible standings of a playoff series and found that teams who are close to being eliminated tend to play better, unless they give up. Simple metrics of measuring game intensity can be the number of player fouls and final score difference between two teams. These metrics indirectly reflect team defense: a tight defense results in an intense game and more fouls. A finer measure would be to see the difference between total ball possessions between two teams.

3.7 Maximum likelihood estimate derivation

Suppose we have $X_1, X_2, \ldots, X_N$ independent and identically distributed random variables with exponential distribution with parameter/rate λ. The probability density function of the exponential distribution is

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases}$$

The likelihood function is

$$\mathcal{L}(\lambda; x_1, \ldots, x_N) = \prod_{i=1}^{N} f(x_i;\lambda) = \prod_{i=1}^{N} \lambda e^{-\lambda x_i} = \lambda^N \text{Exp}\left(-\lambda \sum_{i=1}^{N} x_i\right).$$

To calculate the maximum likelihood estimator, we solve the equation

$$\frac{d \log \mathcal{L}(\lambda; x_1, \ldots, x_N)}{d\lambda} = 0$$

for λ.

$$\begin{aligned}\frac{d\log\mathcal{L}(\lambda;x_1,\ldots,x_N)}{d\lambda} &= \frac{d\log(\lambda^N e^{-\lambda(x_1+\ldots+x_N)})}{d\lambda}\\ &= \frac{N}{\lambda}-\sum_{i=1}^{N}x_i\end{aligned}$$

Solving for λ, we obtain $\lambda = \frac{N}{\sum_{i=1}^{N}x_i}$. To see if this λ is the maximum, we compute $\frac{d^2\mathcal{L}(\lambda;x_1,\ldots,x_N)}{d\lambda^2}$. Since $\frac{d\log\mathcal{L}(\lambda;x_1,\ldots,x_N)}{d\lambda} = \frac{N}{\lambda}-\sum_{i=1}^{N}x_i$, the second derivative with respect to λ will yield

$$\frac{d^2\mathcal{L}(\lambda;x_1,\ldots,x_N)}{d\lambda^2} = -\frac{N}{\lambda^2} < 0.$$

This shows the rate $\lambda = \frac{N}{\sum_{i=1}^{N}x_i}$ captures most observed data.

An important task when modeling basketball data with continuous-time Markov chains is estimating parameters, i.e., transition rates. Suppose we have a data sequence $\{t_1,t_2,\ldots,t_N\}$. We want to infer the rate λ for exponential distribution. We do this using MLE. First, we write down the likelihood function:

$$p(t_1,t_2,\ldots,t_N|\lambda) = p([t_1-t_0,t_2-t_1,\ldots,t_N-t_{N-1}]|\lambda).$$

Since each time segment $T_i = t_{i+1}-t_i$, for $i = 0,1,2,\ldots,N-1$, is independent of each other and follows an exponential distribution with rate λ,

$$\begin{aligned}L = p(t_1,t_2,\ldots,t_N|\lambda) &= p([t_1-t_0,t_2-t_1,\ldots,t_N-t_{N-1}]|\lambda)\\ &= p(t_1-t_0|\lambda)p(t_2-t_1|\lambda)\ldots p(t_N-t_{N-1}|\lambda)\\ &= \lambda e^{-\lambda(t_1-t_0)}\lambda e^{-\lambda(t_2-t_1)}\ldots\lambda e^{-\lambda(t_N-t_{N-1})}\\ L &= \lambda^N e^{-\lambda t_N}\end{aligned}$$

Maximize the log likelihood function, we obtain

$$\begin{aligned}\log L &= N\log\lambda-\lambda t_N,\\ \frac{\partial(\log L)}{\partial\lambda} &= \frac{N}{\lambda}-t_N = 0.\end{aligned}$$

Hence, $\lambda = \frac{N}{t_N}$.

Here we derive MLEs for DTMC and CTMC models.

3.7.1 DTMC

Consider data consisting of a state time series $\{s_0, s_1, s_2, \ldots, s_N\}$. Define $p_{i,j}$ to be the probability of transitioning from state i to state j, i.e., $P(j|i)$. Then the likelihood function is:

$$L = p_{s_0,s_1} p_{s_1,s_2} \cdots p_{s_{N-1},s_N} = \prod_{i,j} p_{i,j}^{N(i,j)}$$

where $N(i,j)$ is the number of times that the pattern (i,j) occurs in the state time series. Note that $p_{i,i} = 1 - \sum_{j \neq i} p_{i,j}$. Therefore,

$$L = \prod_{i,j:i \neq j} p_{i,j}^{N(i,j)} \prod_i \left(1 - \sum_{j \neq i} p_{i,j}\right)^{N(i,i)}.$$

Fix states i', j' and maximize the log likelihood function over the parameter $p_{i',j'}$:

$$\frac{\partial \log L}{\partial p_{i',j'}} = \frac{N(i',j')}{p_{i',j'}} - \frac{N(i',i')}{1 - \sum_{j \neq i'} p_{i',j}} = 0.$$

Suppose there are $M - 1$ states in total. Then for fixed i', the above equation gives us $M - 1$ equations in $M - 1$ unknowns. Let $\vec{1}$ denote the $(M-1) \times 1$ vector of all ones. Then the $M - 1$ equations can be summarized by the matrix-vector equation

$$\left(N(i',i')I + \vec{N}\vec{1}^T\right)\vec{p} = \vec{N}.$$

where $\vec{N}$ is the $(M-1) \times 1$ vector

$$\vec{N} = (N(i',1), \ldots, N(i',i'-1), N(i',i'+1), \ldots, N(i',M))$$

and $\vec{p}$ is the $(M-1) \times 1$ vector

$$\vec{p} = (p_{i',1}, \ldots, p_{i',i'-1}, p_{i',i'+1}, \ldots, p_{i',M}).$$

Note that $\vec{N}\vec{1}^T$ is a rank-1 matrix; the matrix multiplying $\vec{p}$ is therefore a rank-1 perturbation of a multiple of the identity. We can then solve for $\vec{p}$ using the Sherman-Morrison-Woodbury matrix inversion formula:

$$\left(N(i',i')I + \vec{N}\vec{1}^T\right)^{-1} = \frac{1}{N(i',i')}I - \frac{1}{N(i',i')}I\vec{N}\left(1 + \frac{\vec{1}^T\vec{N}}{N(i',i')}\right)^{-1}\vec{1}^T\frac{1}{N(i',i')}I$$

Hence $\vec{p} = \vec{N}/\sum_{j=1}^{M} N(i',j)$. Since i' and j' were arbitrary, we see that the MLE for the (i,j)-th entry of the transition matrix is $\widehat{p}_{i,j} = N(i,j)/\sum_{k=1}^{M} N(i,k)$. In words, this is the number of times we observe the pattern (i,j) divided by the total number of times we observe any of the patterns $(i,1),(i,2),\ldots,(i,M)$. We have derived this formula for $i \neq j$. Because $\widehat{p}_{i,i} = 1 - \sum_{j\neq i}\widehat{p}_{i,j}$, it is valid for $i = j$ as well.

3.7.2 CTMC

Consider data consisting of times $\{0 = t_0, t_1, t_2, \ldots, t_N\}$ and state time series $\{s_0, s_1, s_2, \ldots, s_N\}$. Define $\alpha(x,y)$ as the rate that state x jumps to state y. Then the transition rate out of state x is $\alpha(x) = \sum_{y\neq x}\alpha(x,y)$. For $i = 0,\ldots,N-1$, define $T_i = t_{i+1} - t_i$. Then the likelihood function is (Guttorp 1995):

$$\begin{aligned}
L &= \alpha(s_0)e^{-\alpha(s_0)T_0}\frac{\alpha(s_0,s_1)}{\alpha(s_0)}\ \alpha(s_1)e^{-\alpha(s_1)T_1}\frac{\alpha(s_1,s_2)}{\alpha(s_1)}\ \cdots \\
&= e^{-\alpha(s_0)T_0}\alpha(s_0,s_1)\ e^{-\alpha(s_1)T_1}\alpha(s_1,s_2)\ \cdots \\
&= e^{-\sum_x \alpha(x)W(x)}\prod_{x,y:x\neq y}\alpha(x,y)^{N(x,y)},
\end{aligned}$$

where $W(x) = \sum_{i=0}^{N-1} T_i \cdot \mathbf{I}(s_i = x)$ and $N(x,y) = \sum_{i=0}^{N-1}\mathbf{I}(s_i = x, s_{i+1} = y)$. In words, $W(x)$ is the total time spent in state x, and $N(x,y)$ is the total number of times that the pattern (x,y) is observed. Then

$$\log L = \sum_{x,y:x\neq y}[-W(x)\alpha(x,y) + N(x,y)\log\alpha(x,y)].$$

Fix states x', y' and maximize the log likelihood function over the parameter $\alpha(x',y')$:

$$\frac{\partial \log L}{\partial\alpha(x',y')} = -W(x') + \frac{N(x',y')}{\alpha(x',y')} = 0.$$

We obtain the MLE $\widehat{\alpha}(x', y') = N(x', y')/W(x')$. This holds for all $x' \neq y'$. Hence the MLE for the (x, y) entry of the transition rate matrix is the total number of transitions from state x to state y, divided by the total time spent in state x.

Chapter 4

Removing Absorbing States from Markov Chain Models

4.1 Introduction

We study the problem of estimating Markov chain models for discrete-state systems. Suppose we observe such a system continuously in time as it transitions from one state to the next. Further suppose that there exists a set of one or more states (say, S) from which the system never transitions to another state. Maximum likelihood estimation will then yield a Markov chain model in which the states in S are absorbing. Consequently, the equilibrium or stationary distribution associated with the Markov chain will be supported only on S. The estimated model will assign a probability of zero to the long-term probability of spending time in states other than S. For several systems of interest, this will destroy the predictive power of the estimated model.

Our motivating example arises in the area of basketball analytics. To model substitutions and minutes played by each player for real NBA (National Basketball Association) games, we have built continuous-time Markov chain models. For a given team, each state corresponds to a distinct unit of five players on the court. Transitions from one state to another correspond to the substitution of one or more players. Given a time-stamped, play-by-play transcript of a game, we know exactly how long each five-person unit played on the court and exactly when substitutions occurred—in this sense, the system is observed continuously in time. In many regular-season NBA games, when one team is ahead by a large margin towards the end of the game, there is a high probability that one team (or both teams) will substitute in rookies or non-starting players. Hence it is likely that the last observed state (say, σ) of the system is a state that has been reached for the first time in the game (and possibly for the first time in the entire season). As described above, in the estimated model, σ will then be an absorbing state. The equilibrium distribution for the estimated model will then have nothing to do with the empirical fraction of time actually spent in each state.

The main contribution of this chapter is a a pair of algorithms for removing absorbing states from both continuous-time Markov chain (CTMC) and discrete-time Markov chain (DTMC) models. In both the CTMC and DTMC cases, the model is specified by an $M \times M$ matrix $\widehat{p}$, where M is the number of states. Starting from the maximum likelihood estimate (MLE) of $\widehat{p}$, we ask: what is the most sparse perturbation ε that yields an absorbing-state-free Markov chain $\widehat{p} + \varepsilon$ whose equilibrium vector exactly matches the observed fraction of time spent in each state? We formulate this question as an optimization problem and show how to solve it efficiently using linear programming.

While we have not found published work that solves the problems outlined above, we can certainly identify subsets of the literature that help contextualize the present work. Perhaps the most relevant subset is that dealing with inverse eigenvector problems for nonnegative matrices—see Chu and Guo (1998); Chu and Golub (2005); Bai *et al.* (2012). Here we find techniques to construct nonnegative matrices with prescribed eigenvalues and eigenvectors. Once such a nonnegative matrix has been constructed, it can be transformed into a stochastic matrix, i.e., a transition matrix for a DTMC. It is entirely possible to use these methods to construct a DTMC whose equilibrium vector matches data, i.e., matches the empirical fraction of time spent in each state. Special algorithms to deal with this particular case of the inverse eigenvector problem have been developed (Kumar *et al.* 2015; Maystre and Grossglauser 2015). However, there is no link between these techniques and short-term trends in the data—for each $i \neq j$, the (i, j) entry of the resulting stochastic matrix does not necessarily have anything to do with the empirical frequency of transitions from state i to state j. In contrast, our method *constrains* the estimated model to be close to these frequencies, i.e., the MLE estimates $\widehat{p}_{i,j}$.

Note that inverse eigenvector methods can be used to construct Markov Chain Monte Carlo (MCMC) methods with optimal properties (Wu and Chu 2015). We plan to explore in future work whether the methods from the present chapter can be used in the MCMC context. Note also that the inverse eigenvector problem—in which a desired eigenvalue-eigenvector pair is prescribed—is different from inverse problems for eigenvalues only (Laurie 1991; Chen and Liu 2011; Yao *et al.* 2016).

The next subset of papers deals with PageRank, a fundamental algorithm used to rank web pages (Langville and Meyer 2006a). Consider a collection $\mathcal{C}$ of M interlinked web pages and treat each web page as a Markov chain state. If there are n links on a particular web page, we assign a probability of $1/n$ to transition from that particular web page to each of the linked pages. This gives us a Markov chain corresponding to a random walk on the graph corresponding to $\mathcal{C}$. Also consider the $M \times M$ stochastic matrix in which each entry is $1/M$—this corresponds to a Markov chain in which all states communicate uniformly with all other states. The PageRank DTMC model consists of a linear combination of the random walk and uniform transition matrices; if the weights in the linear combination sum to 1, the resulting matrix is itself a valid Markov chain. The equilibrium vector of the PageRank DTMC model can then be used to rank web pages.

There has been significant development in estimating/analyzing how this equilibrium vector changes when links between web pages are changed (Ng *et al.* 2001; Langville and Meyer 2006b; Chartier *et al.* 2011), and also in optimizing the PageRank vector according to various criteria and methods (Fercoq *et al.* 2013; Fercoq 2014; Csáji *et al.* 2014). Given a desired PageRank equilibrium vector w and a PageRank DTMC model $\widehat{p}$ that has already been estimated, our algorithm computes the most sparse perturbation ε that yields a DTMC model $\widehat{p} + \varepsilon$ with equilibrium w. Suppose we treat the web graph as weighted, as in weighted

PageRank (Gleich 2015) or certain link recommendation models (Backstrom and Leskovec 2011). Then the ε developed by our algorithm tells us how to adjust weights to achieve a desired ranking of pages.

Finally, we should mention that the problem of estimating CTMC models from *temporally discrete* observations is entirely different from the problem considered here. If we only observe the CTMC process at discrete times, then we do not necessarily know that those times correspond to transitions, nor can we rule out transitions occurring at times in between the times at which we have data. This leads to a more complex estimation problem that has attracted much recent interest—see Metzner *et al.* (2007b); Rao and Teh (2013); Hajiaghayi *et al.* (2014); Crawford *et al.* (2014).

4.2 Background

For the purposes of this chapter, we describe Markov chains via the methods we use to generate sample trajectories. For more mathematical definitions, consult Lawler (2006).

A DTMC with M states is determined by an $M \times M$ transition matrix p. Suppose the DTMC is currently in state i. We examine p_i, row i of the matrix p. To be a valid transition matrix, each p_i must be a probability mass function over the state space. To generate the next sample in the trajectory, we sample from the distribution p_i; this yields a state $i' \in \{1, 2, \ldots, M\}$.

To continue, we repeat the above procedure, starting in state i'. In this way, we can generate trajectories of arbitrary length, consisting of sequences of states.

A CTMC with M states is determined by an $M \times M$ transition rate matrix p. Suppose the CTMC is currently in state i. We examine p_i, row i of the matrix p. The entries of p_i off the diagonal must be nonnegative—we treat these entries as parameters of $M - 1$ independent, exponentially distributed random variables. We draw one sample from each exponential random variable. The minimum of these samples represents how long we spend in state i before transitioning. The argmin of these samples represents the new state i' to which we transition.

To continue, we repeat the above procedure, starting in state i'. In this way, we can generate trajectories of arbitrary length, consisting of sequences of states and corresponding transition times.

4.3 Mathematical Methods

4.3.1 DTMC Optimization Problem and Solution

Let us first consider the problem for DTMC models. Suppose we have an $M \times M$ transition matrix $\widehat{p}$, estimated using MLE, that has absorbing states. We seek a perturbation ε such that $\widehat{p} + \varepsilon$ has no absorbing states. We still want $\widehat{p} + \varepsilon$ to be a valid Markov transition matrix, so we require that

$$\sum_j (\widehat{p}_{i,j} + \varepsilon_{i,j}) = 1. \tag{4.1}$$

Of course, $\sum_j \widehat{p}_{i,j} = 1$ already, implying

$$\sum_j \varepsilon_{i,j} = 0. \tag{4.2}$$

Because of this constraint, we only need to solve for the off-diagonal part of ε, a total of $M^2 - M$ unknowns. We set

$$\varepsilon_{i,i} = -\sum_{j \neq i} \varepsilon_{i,j}. \tag{4.3}$$

Assume that w is a desired equilibrium vector. This yields an additional constraint:

$$w^T(\widehat{p} + \varepsilon) = w^T.$$

In coordinates, this reduces to

$$\sum_{i \neq j} w_i \varepsilon_{i,j} - \sum_{i \neq j} w_j \varepsilon_{j,i} = w_j - \sum_i w_i \widehat{p}_{i,j}.$$

The entries of the perturbed matrix must also be valid probabilities:

$$0 \leq \widehat{p}_{i,j} + \varepsilon_{i,j} \leq 1$$

for all i and all j. For $i \neq j$, we can enforce this constraint verbatim. For $i = j$, we use (4.3) to rewrite the constraint as:

$$0 \leq \widehat{p}_{i,i} - \sum_{j \neq i} \varepsilon_{i,j} \leq 1.$$

For each i, we enforce the lower bound of 0:

$$\sum_{j \neq i} \varepsilon_{i,j} \leq \widehat{p}_{i,i}.$$

However, for the upper bound, we replace 1 by w_i and require, for each i,

$$-\sum_{j \neq i} \varepsilon_{i,j} + \widehat{p}_{i,i} \leq w_i. \tag{4.4}$$

In this chapter, we will take w to be a candidate equilibrium distribution with no absorbing states, i.e., $0 < w_i < 1$ for each i. Hence (4.4) and (4.3) guarantee that $\varepsilon_{ii} + \widehat{p}_{ii} \leq \max_i w_i < 1$. In short, the diagonal entries of the perturbed transition matrix $\widehat{p} + \varepsilon$ are bounded away from 1, implying that the perturbed system cannot have any absorbing states.

We also enforce (4.4) because we know that the Markov transition matrix given by

$$\widehat{p} + \varepsilon = \begin{bmatrix} \text{—}w\text{—} \\ \text{—}w\text{—} \\ \vdots \\ \text{—}w\text{—} \end{bmatrix} \tag{4.5}$$

automatically has equilibrium distribution w. Hence we know there exists at least one feasible solution where the diagonal elements satisfy $\widehat{p}_{ii} + \varepsilon_{ii} = w_i$.

We now consider the choice of objective function. A natural first choice might be the sum of the squares of the off-diagonal elements of ε, essentially a form of the 2-norm. This objective function does not promote sparsity. It is likely that the resulting solution will cause each MLE $\widehat{p}_{i,j}$ to be perturbed. We view this as undesirable.

Instead, we would like to find a new transition matrix $\widehat{p} + \varepsilon$ that retains as many of the original MLE entries $\widehat{p}$ as possible. In other words, we want to maximize the number of zero entries of ε. Consequently, we set our objective function equal to the sparsity-promoting 1-norm of the off-diagonal elements of ε:

$$J(\varepsilon) = \sum_{i,j:i\neq j} |\varepsilon_{i,j}|.$$

Putting the objective and constraints together, we are led to the following optimization problem:

$$\begin{aligned}
\min_{\varepsilon} \quad & J(\varepsilon) \\
\text{s.t.} \quad & 0 \leq \widehat{p}_{i,j} + \varepsilon_{i,j} \leq 1, \ \forall i \neq j \\
& 0 \leq -\sum_{j\neq i} \varepsilon_{i,j} + \widehat{p}_{i,i} \leq w_i, \ \forall i \\
& \sum_{i\neq j} w_i \varepsilon_{i,j} - \sum_{i\neq j} w_j \varepsilon_{j,i} = w_j - \sum_i w_i \widehat{p}_{i,j}, \ \forall j
\end{aligned} \tag{4.6}$$

To handle the 1-norm minimization problem, we employ the standard technique of introducing decision variables $t_{i,j}$ and additional constraints. The resulting optimization problem, equivalent to the one above, is:

$$\begin{aligned}
\min_{\varepsilon,t} \quad & \sum_{i\neq j} t_{i,j} \\
\text{s.t.} \quad & -t_{i,j} \leq \varepsilon_{i,j} \leq t_{i,j}, \ \forall i \neq j \\
& t_{i,j} \geq 0, \ \forall i \neq j \\
& 0 \leq \widehat{p}_{i,j} + \varepsilon_{i,j} \leq 1, \ \forall i \neq j \\
& 0 \leq -\sum_{j\neq i} \varepsilon_{i,j} + \widehat{p}_{i,i} \leq w_i, \ \forall i \\
& \sum_{i\neq j} w_i \varepsilon_{i,j} - \sum_{i\neq j} w_j \varepsilon_{j,i} = w_j - \sum_i w_i \widehat{p}_{i,j}, \ \forall j.
\end{aligned} \tag{4.7}$$

This optimization problem is a linear program (Nocedal and Wright 2006). Modern linear programming algorithms enable the solution of this problem efficiently and accurately for systems with hundreds of states M, i.e., when the number of decision variables is roughly 10^5. We use CVXOPT (Andersen *et al.* 2018) and Mosek (ApS 2018) to solve all linear programming problems described in this chapter. Source code is available upon request.

4.3.2 CTMC Optimization Problem and Solution

Let us now consider the problem for CTMC models. Suppose we have an $M \times M$ transition rate matrix $\widehat{\alpha}$ estimated using MLE. Suppose that $\widehat{\alpha}$ has absorbing states.

We seek a perturbation ε such that $\widehat{\alpha}$ has no absorbing states. For the continuous-time Markov chain, this means that for each i, we must have

$$\sum_{j \neq i} (\widehat{\alpha} + \varepsilon)_{i,j} > 0.$$

If this sum were to be zero, then state i would be an absorbing state for the continuous-time Markov chain. Note, however, that numerical optimizers do not distinguish between strict and non-strict inequalities. To guarantee positivity, we instead use the constraint

$$\sum_{j \neq i} (\widehat{\alpha} + \varepsilon)_{i,j} \geq \delta > 0$$

for some tolerance δ. Note that for a continuous-time Markov chain, we always choose the diagonal elements of the transition rate matrix to guarantee that the total row sum equals zero. For this reason, we only need to solve for the off-diagonal elements of ε, just as in the discrete-time case.

Assume that w is a desired equilibrium vector, again with $0 < w_i < 1$ for all i. For the perturbed system to have w as its equilibrium, we must have

$$w^T (\widehat{\alpha} + \varepsilon) = 0.$$

Let $\mathcal{Y} = \{y_1, \ldots, y_M\}$ denote all row sums—not including the diagonal element—of the estimated transition rate matrix $\widehat{\alpha}$. We set $\delta_1 = \min \mathcal{Y} \cap \{r \ : \ r > 0\}$, the minimum positive entry from $\mathcal{Y}$. We also set $\delta_2 = \min_{1 \leq i \leq M} (1 - w_i)$. Finally, we set $\delta = \min\{\delta_1, \delta_2\}$. By setting δ in this way, we ensure that a system that has *no* absorbing states and that already achieves the desired equilibrium vector w will result in a solution of $\varepsilon = 0$. We also ensure that there always exists a feasible solution—with the desired equilibrium vector w—given by

$$\widehat{\alpha} + \varepsilon = \begin{bmatrix} \text{—}w\text{—} \\ \text{—}w\text{—} \\ \vdots \\ \text{—}w\text{—} \end{bmatrix} - I, \tag{4.8}$$

where I is the $M \times M$ identity matrix. Additionally, we require that the perturbed transition rates are nonnegative: for all $i \neq j$,

$$\widehat{\alpha}_{i,j} + \varepsilon_{i,j} \geq 0.$$

Because we are interested in retaining as many of the MLE entries of $\widehat{\alpha}$ as possible, we again seek a perturbation ε that is as sparse as possible. Therefore, we use the 1-norm objective function J defined above. Putting the objective and constraints together, we are led to the following optimization problem:

$$\begin{aligned} \min_{\varepsilon} \quad & J(\varepsilon) \\ \text{s.t.} \quad & \widehat{\alpha}_{i,j} + \varepsilon_{i,j} \geq 0, \ \forall i \neq j \\ & \sum_{j \neq i} (\widehat{\alpha} + \varepsilon)_{i,j} \geq \delta > 0, \ \forall i \\ & - w_j \sum_{i \neq j} (\widehat{\alpha} + \varepsilon)_{j,i} + \sum_{i \neq j} w_i (\widehat{\alpha} + \varepsilon)_{i,j} = 0, \ \forall j. \end{aligned} \tag{4.9}$$

To handle the 1-norm minimization problem, we again employ the standard technique of introducing decision variables $t_{i,j}$ and additional constraints. The resulting optimization problem, equivalent to the one above, is:

$$\begin{aligned} \min_{\varepsilon, t} \quad & \sum_{i \neq j} t_{i,j} \\ \text{s.t.} \quad & - t_{i,j} \leq \varepsilon_{i,j} \leq t_{i,j}, \ \forall i \neq j \\ & t_{i,j} \geq 0, \ \forall i \neq j \\ & \widehat{\alpha}_{i,j} + \varepsilon_{i,j} \geq 0, \ \forall i \neq j \\ & \sum_{j \neq i} (\widehat{\alpha} + \varepsilon)_{i,j} \geq \delta > 0, \ \forall i \\ & - w_j \sum_{i \neq j} (\widehat{\alpha} + \varepsilon)_{j,i} + \sum_{i \neq j} w_i (\widehat{\alpha} + \varepsilon)_{i,j} = 0, \ \forall j. \end{aligned} \tag{4.10}$$

Again, this is a linear program (Nocedal and Wright 2006); we solve it using the same software described above.

4.4 Simulated Data Tests

We have subjected the methods described in Section 4.3 to a battery of tests with simulated data. For each test, we generate data from a known discrete- or continuous-time Markov chain. When generating the data, we ensure that there is one state from which the system does not exit. When we apply MLE to this data, we obtain a Markov chain model $\widehat{p}$ that we refer to as the *naive* model.

We then apply the optimization method from Section 4.3 to remove the absorbing state while simultaneously driving the Markov chain's equilibrium vector to equal the empirical fraction of time spent in each state. In this way, we obtain $\widehat{p} + \varepsilon$, which we refer to as the *fixed* model.

4.4.1 Metrics

There are two primary metrics in which we compare the naive and fixed models. The first, which we call *long-term error*, relates to the ability of the model to predict well in a distributional sense. Specifically, given a time series, let π^{emp} denote the empirical fraction of time spent in each state. For the DTMC model, π_i^{emp} is the number of times we observe state i normalized by the length L of the time series. Given a DTMC model with transition matrix P, we compute π^{mod} by solving $\pi^{\text{mod}} P = \pi^{\text{mod}}$, i.e., we find the left eigenvector with eigenvalue 1.

For the CTMC model, π_i^{emp} is the total amount of time spent in state i normalized by the total time T spanned by the time series. Given a CTMC model with transition rate matrix P, we compute π^{mod} by solving $\pi^{\text{mod}} P = 0$, i.e., we find the left eigenvector with eigenvalue 0.

For both the DTMC and CTMC cases, the long-term error is the L^1-norm distance between the empirical and model distributions:

$$\mathcal{E}_{\text{LT}} = \|\pi^{\text{mod}} - \pi^{\text{emp}}\|_1 = \sum_i \left|\pi_i^{\text{mod}} - \pi_i^{\text{emp}}\right|. \tag{4.11}$$

Since both π^{mod} and π^{emp} are probability mass functions, we can show that $\mathcal{E}_{\text{LT}} \leq 2$.

The second metric, *short-term error*, conveys the ability of the model to predict the very next element in the time series. Suppose we have a Markov chain model and data consisting of a sequences of states $\{s_0, s_1, s_2, \ldots, s_L\}$. For each $i \in \{0, 1, \ldots, L-1\}$, we use the Markov chain to compute a one-step prediction $\widehat{s}_{i+1}$. We then compute the average short-term error via

$$\mathcal{E}_{\text{ST}} = \frac{1}{L} \sum_{i=0}^{L-1} 1_{\widehat{s}_{i+1} \neq s_{i+1}} \tag{4.12}$$

Here 1_X is the indicator function that equals 1 if condition X is true and 0 otherwise. For both the DTMC and CTMC models, we compute the prediction $\widehat{s}_{i+1}$ by sampling from the Markov chain conditional on currently being in state s_i. In both cases, this amounts to examining row s_i of the transition (or transition rate) matrix P and applying the appropriate sampling procedure from Section 4.2.

4.4.2 Data Generation and Optimization Test

Let us now describe how we generate data for all tests in this section. Let M be the number of states in both the data and the naive/fixed Markov chain.

For DTMC tests, we begin by randomly generating a transition matrix over $M-1$ states. To do this, we first create an $(M-1) \times (M-1)$ matrix P of integers drawn uniformly from $\{0, 1, \ldots, 500\}$—we think of the unnormalized P_{ij} as a simulated count of transitions from state i to state j. We then normalize each row of P to sum to 1.

In the CTMC case, we again create an $(M-1) \times (M-1)$ unnormalized count matrix P of integers drawn uniformly from $\{0, 1, \ldots, 500\}$. We then create an $M - 1 \times 1$ vector of integers $\vec{u}$ drawn uniformly from $\{50, 51, \ldots, 500\}$—we think of this as the total time spent in each of the $M-1$ states. We divide each row of P by the corresponding entry of $\vec{u}$. We then reset the diagonal entries of P so that each row sums to zero, i.e., we set $P_{ii} = -\sum_{j \neq i} P_{ij}$. The final P is a valid transition rate matrix over $M-1$ states.

Equipped with a randomly generated Markov model, we apply the sampling procedure described in Section 4.2 to create both training and test time series. To the end of the training time series, we then tack on one final transition into state M; as this is the first entry into state M, there is clearly no exit from state M. We then train one naive model $\widehat{p}$ by applying MLE to the training set—for this naive model, state M is an absorbing state. Using the training set, we compute the empirical fraction of time spent in each state, π^{emp}. We then produce a fixed model by applying the optimization procedure from Section 4.3 to $\widehat{p}$, setting the desired equilibrium vector w equal to π^{emp}.

Each time we apply the optimization procedures from Section 4.3, the optimizer returns to us both ε and t. The optimizer also reports the maximum (or ∞-norm) violation of all constraints. These constraints, as described in Section 4.3, relate to guaranteeing that $\widehat{p} + \varepsilon$ is a valid Markov chain with no absorbing states and with equilibrium vector $w = \pi^{\text{emp}}$.

	Constraint Violation	
States	CTMC	DTMC
4	9.958124e-10	6.519809e-17
8	3.405134e-16	6.172510e-17
16	7.585014e-16	6.195263e-17
32	2.110672e-15	2.029552e-16
64	4.193528e-15	2.527282e-16
128	4.848341e-15	1.058269e-16
256	3.958157e-15	1.411389e-09

Table 4.1: We record the average constraint violation for fixed CTMC and DTMC models as a function of the number of states. These results have been averaged across 1000 simulations.

In Table 4.1, we report the average constraint violation for fixed DTMC and CTMC models. For each indicated number of states M, one simulation consists of applying the above procedure to generate a training time series of length 1000, computing naive and fixed models, and recording the constraint violation. The values reported in Table 4.1 have been averaged over 1000 simulations. The small values that we see indicate that, for practical purposes, the optimizer does achieve the desired constraints.

4.4.3 Long-Term Error Test

The goal of the procedure described in Section 4.3 is to reduce the long-term prediction error of estimated Markov chains. To show that our procedure meets this goal, we carry out the following test. We fix the number of states M and, as described above, generate a random, $(M-1)$-state Markov chain that we then sample to create both a training time series of length 10^4 and a test time series of length $L \in \{5 \times 10^2, 10^3, 10^4, 10^5, 10^6\}$. For the training time series, we tack on a final transition into state M.

Let $\widehat{p}$ be the naive MLE model estimated from the training set. As above, we also use the training set to compute the empirical fraction of time spent in each state, π^{emp}. We then produce a fixed model by applying the optimiation procedure from Section 4.3 to $\widehat{p}$, setting the desired equilibrium vector w equal to π^{emp}.

Using both training and test time series, we compute the long-term error $\mathcal{E}_{\text{LT}}$—see (4.11)—for all fixed models. In Table 4.2, we display long-term training (top) and test (bottom) errors for the fixed CTMC model. In Table 4.3, we display long-term training (top) and test (bottom) errors for the fixed DTMC model. We see that the training error is practically zero for each run, indicating that the optimizer succeeds in creating fixed Markov models with the desired equilibrium vector.

We see from Tables 4.2 and 4.3 that, for each fixed number of states M, as the length of the test set increases, the long-term test error decreases. This matches our expectation from theory. Over any finite-length test set, it is possible for the empirical fraction of time spent in each state to deviate from the equilibrium vector for the data-generating Markov chain. By the law of large numbers, these *sampling error* deviations must decay to zero as the length of the test set increases. Note that the equilibrium vector for the data-generating, original, $(M-1)$-state Markov chain does in fact differ from the training set's π^{emp} vector. However, these differences are negligible compared to the sampling error.

If we were to create tables analogous to Tables 4.2 and 4.3 for the naive CTMC and DTMC models, every entry would be close to 2.0, the maximum possible value of $\mathcal{E}_{\text{LT}}$. This is because state M is the unique absorbing state for the naive model; therefore, $\pi^{\text{mod}} = (0, \ldots, 0, 1)$. Meanwhile, the M-th entry of the π^{emp} vector is either nearly zero (for the training set) or identically zero (for the test set). This reflects the fact that in our training data, state M is reached once and only once.

Length	5×10^2	10^3	10^4	10^5	10^6
States					
4	5.022e-10	6.589e-13	5.896e-09	3.424e-11	1.103e-11
8	2.081e-16	3.400e-16	2.775e-16	2.081e-16	4.996e-16
16	2.567e-16	3.469e-16	2.255e-16	1.769e-16	1.838e-16
32	1.847e-16	4.909e-16	3.382e-16	2.818e-16	2.324e-16
64	3.087e-16	4.401e-16	2.697e-16	3.313e-16	3.456e-16
128	7.942e-16	6.338e-16	7.218e-16	5.126e-16	4.898e-16
256	6.372e-16	7.938e-16	4.592e-16	5.651e-16	4.406e-16

Length	5×10^2	10^3	10^4	10^5	10^6
States					
4	0.051	0.058	0.006	0.035	0.004
8	0.123	0.068	0.029	0.022	0.023
16	0.177	0.137	0.053	0.058	0.032
32	0.279	0.160	0.091	0.065	0.069
64	0.383	0.289	0.103	0.096	0.079
128	0.526	0.400	0.180	0.136	0.122
256	0.748	0.622	0.258	0.189	0.170

Table 4.2: Long-term training (top) and test (bottom) errors for the fixed CTMC procedure at each of the test time series lengths and different number of states. The training time series is of length 10000.

4.4.4 Short-Term Error Test

While it is clear from the above tests that our procedure dramatically reduces long-term prediction error, we want to be sure that the fixed Markov models retain the short-term predictive accuracy of the naive models. Given a DTMC or CTMC system, the MLE entries of the naive model are already optimized for short-term predictive power. Because we have used the sparsity promoting 1-norm objective function for ε, we expect that the optimizer will set most entries of ε to be zero. If this is the case, the fixed model will retain many of the MLE entries from the naive model, and therefore the fixed model's short-term predictive power should not differ greatly from that of the naive model.

To test this expectation, we run the following simulation procedure in both the DTMC and CTMC cases: for fixed M, we follow the procedure described above to generate a random, $(M-1)$-state Markov chain that we then sample to create both a training time series of length 10^3 and a test time series of length 3×10^3. For the training time series, we append a final transition into state M. We then compute the naive and fixed models using the training set only, in the same way as for the long-term test. For both the naive and fixed models, we then use the test set to compute the short-term errors $\mathcal{E}_{\text{ST}}$—see (4.12). We record the absolute difference between the short-term error of the naive and fixed models.

We repeat the above simulation procedure $S = 1000$ times. We have chosen this number of simulations by simulating repeatedly until all short-term errors averaged over the first $s \leq S$ simulations stabilize to within 10^{-2} of the average over all S simulations. The results, displayed in Table 4.4, show that each fixed model's short-term predictive power does not differ greatly from that of the corresponding naive model. We see that, along each row, errors do not differ by more than 10^{-2}. As we proceed down the table, we see smaller differences between the short-term errors of the naive and fixed models. This is due to larger state spaces yielding more sparse solutions for ε. For the largest state space ($M = 256$), the dimension of ε is $M(M-1) = 65280$; for systems of this size, it is typical that less than one percent of the entries returned by the optimizer are nonzero.

Let us note that even in the idealized setting where we have perfect knowledge of the Markov chain that generates data, we will still not achieve zero short-term error. For instance, consider a two-state DTMC model where every entry of the transition matrix is 1/2—the best possible short-term error will be 1/2. This is another reason to consider the absolute difference between naive and fixed models, rather than the raw short-term errors produced by both.

4.5 Real Data Tests

In this section, we apply the optimization procedures from Section 4.3 to Markov models estimated from real data sets. As our primary motivating example arises in basketball analytics, we begin with CTMC models for regular-season NBA games. We then briefly examine the performance of our procedure on DTMC models for biomedical data.

4.5.1 NBA Data

We begin with a description of the raw data itself. For each regular-season game from the 2015-16 NBA season, we obtained play-by-play files (in HTML form) from public web sites. These files contain time-stamped textual markers that we mined to determine who was playing on the court at all times for all games. Each team plays 82 games per season. Starting with all 1230 regular-season games, we omitted games that went to overtime. Hence all games in the data set considered here lasted 48 minutes (2880 seconds). For each team, we assigned a state number to each unique 5-person unit that played at any time for that team. Using these state numbers and the time stamps at which substitutions occur, we then extracted from each game a trajectory $\{(t_i, s_i)\}_{i=0}^{N}$, where $t_0 = 0$ and $t_N = 2880$. At each time t_i, the system transitions from state s_{i-1} to state s_i, corresponding to a substitution of one or more players on the court for one team only.

Our goal here is to use this data to build CTMC models that predict how long each 5-person unit plays on the court. Once we fix the sizes of the training and test sets, we build one naive CTMC model for each team. We find that all naive models—including those trained on all non-overtime games—features at least one absorbing state; in many cases, several absorbing states exist. Using the empirical fraction of time spent in each state, computed using the corresponding training set, we applied the optimization procedure from Section 4.3 to generate fixed CTMC models for each team.

In Figure 4.1, we plot training set result for the naive (top) and fixed (bottom) CTMC models. For these results only, the training set consists of all non-overtime games from the entire regular season. Each point on each plot corresponds to one 5-person unit for one team. We see that the optimization procedure results in a fixed model in which the naive model's training error has been reduced to nearly zero.

For the remainder of this section, we consider naive and fixed CTMC models trained on proper subsets of the regular season. We build naive and fixed CTMC models using a training set of either the first 40 or first 60 non-overtime games played by each team. The corresponding test sets consist of the remaining non-overtime games in the season, less than or equal to 42 or 22 games, respectively.

In Table 4.5, we examine the performance of our optimization procedure applied to CTMC models trained on, respectively, the first 40 and 60 non-overtime games. The tables show that the optimizer succeeds in finding highly sparse solutions that satisfy all constraints—removing absorbing states and achieving the desired equilibrium. The sparsity is indicated by the small number of nonzero entries of the computed ε compared to its dimension, which equals $M(M-1)$. Note also that M, the total number of unique 5-person units that appear for each team in each training set, is always in the hundreds.

In Figure 4.2 (40-game) and Figure 4.3 (60-game), we plot test set results for naive (left) and fixed (right) CTMC models with differing training set sizes. Each point on each plot corresponds to a 5-person unit for a given team. For each team, the predicted playing times for each 5-person unit correspond to the entries in the equilibrium vector for the CTMC model for that team, scaled by the number of seconds in the test set for that team. We plot the real time played by each 5-person unit versus the predicted playing time. Note that there is a massive concentration of points near $(0,0)$; we have decided against log-scaled axes in order to give a sense of the range of the predictions and true times in natural units. For the naive 40-game model, the RMSE (root mean-squared error) between all predicted and real times is 2.979. For the fixed 40-game model, the RMSE is 1.178, approximately 60.4% less than the naive model. This RMSE is measured in thousands of seconds across roughly 40 games; in minutes per game, the 40-game fixed model's average error is 0.49.

The naive 60-game model's RMSE is 1.584, while the fixed 60-game model's RMSE is 0.602, approximately 61.9% less. This RMSE is measured in thousands of seconds across roughly 20 games; in minutes per game, the 60-game fixed model's average error is 0.50.

Hence each CTMC predicts each 5-person unit's per-game playing time to within half a minute, on average. We consider this to be an excellent result.

4.5.2 Biomedical data

Holson and preproglucacon are discrete-time data sets from the R package `markovchain` (Spedicato *et al.* 2017). The Holson data set contains life history trajectories for 1000 unique patients, each measured at 11 points in time. The measurement at each time has value 1, 2, or 3. We split the data into training and test sets of size 500 each. We fit a 3-state DTMC to this data and compare the performance of naive and fixed models.

Preproglucacon data is the DNA sequence for the gene that encodes the protein preproglucacon. This data consists of 1572 observations with bases A, T, C, G coded numerically as 1, 4, 2, 3. We split this data into a training set of size 500 and a test set of size 1072. Using this data, we fit a 4-state DTMC and compare the performance of naive and fixed models.

For the sake of comparison, we also fit a hidden Markov model (HMM) to both data sets. The HMM is much more sophisticated than the DTMC and requires much more computational effort to train (Fraser 2008). When we trained HMM models, we explored hyperparameters such as the number of internal states and the random initialization of the model. We report results for the best (smallest long-term test error) hyperparameter choices we were able to find.

Tables 4.6 and 4.7 show long-term training and test errors for naive DTMC, fixed DTMC, and HMM models. The fixed DTMC models feature greatly reduced test set errors as compared to the naive DTMC models. Note also that for the Holson data set, the long-term test errors for the fixed DTMC and HMM models are comparable. For the preproglucacon data, the HMM's long-term test error is about 4 times less than that of the fixed DTMC.

While the HMM is able to achieve better test set results in some cases, we show in Table 4.8 that this achievement comes at great computational expense. We require about 50 times less computational time to produce the fixed DTMC compared to the HMM. This includes time spent by the linear programming solver.

Tables 4.6 and 4.7 show results of the DTMC models and HMM models. Tables 4.9 and 4.10 showed how we picked the optimal internal states to use for our HMM models. The long-term total error is computed as the one-norm difference between training equilibrium vector and the fraction of time spent in each state for the whole data. By whole data, we mean the union of the training and test sets.

4.6 Remarks

Both CTMC and DTMC models containing absorbing states do not capture well the observed fraction of time spent in each state. We remove absorbing states by finding a sparse perturbation to the transition (or transition rate) matrix such that the new matrix achieves a desired equilibrium distribution. We formulated this problem as a linear programming problem. Through extensive tests with simulated and real data, we have shown that our method improves long-term predictions without sacrificing much short-term accuracy. Furthermore, our method requires far less time to train than HMM methods.

One extension of our work is to see how well our models fit misspecified data. In a simulated data test with misspecified data, we generate data from some model that is totally different from the models considered in this chapter. Such model could be a hidden Markov model, a model with non-Markovian properties such as memory, or even a model like a differential equation whose output we then convert into a discrete state time series. Another direction would be to apply our methods to partially observed data. This arises commonly in medical data where a disease progression is not fully observed. Hidden Markov models are usually used for partially observed data. We would like to see how well our models perform in comparison to HMM models.

Length	5×10^2	10^3	10^4	10^5	10^6
States					
4	2.738e-16	4.106e-18	3.010e-16	1.388e-16	3.566e-16
8	3.222e-16	3.514e-16	1.572e-16	2.233e-16	2.399e-16
16	1.507e-16	1.896e-16	1.355e-16	1.841e-16	2.675e-16
32	2.263e-16	2.715e-16	2.736e-16	1.333e-16	4.137e-16
64	4.092e-16	3.647e-16	4.178e-08	3.592e-16	4.585e-16
128	1.082e-15	1.014e-15	7.295e-16	7.309e-16	8.954e-16
256	9.276e-16	6.937e-16	6.796e-16	7.195e-16	6.593e-16

Length	5×10^2	10^3	10^4	10^5	10^6
States					
4	0.019	0.040	0.009	0.008	0.016
8	0.069	0.090	0.034	0.015	0.024
16	0.086	0.086	0.050	0.026	0.028
32	0.185	0.179	0.053	0.048	0.061
64	0.286	0.214	0.080	0.051	0.061
128	0.447	0.285	0.130	0.099	0.086
256	0.585	0.422	0.172	0.136	0.135

Table 4.3: Long-term training (top) and test (bottom) errors for the fixed DTMC procedure at each of the test time series lengths and different number of states. The training time series is of length 10000.

Simulations	50	250	500	1000
States				
4	1.57e-02	1.91e-02	1.89e-02	2.05e-02
8	4.65e-03	3.15e-03	3.70e-03	3.62e-03
16	1.73e-03	6.74e-04	8.84e-04	8.94e-04
32	2.27e-04	2.60e-04	1.76e-04	8.70e-05
64	6.00e-05	2.41e-04	1.81e-04	1.38e-04
128	4.33e-04	3.06e-04	2.18e-04	8.30e-05
256	2.53e-04	1.95e-04	9.00e-05	7.80e-05

Simulations	50	250	500	1000
States				
4	5.60e-04	9.83e-04	1.03e-03	1.14e-03
8	1.81e-03	3.20e-04	6.38e-04	5.85e-04
16	6.80e-04	6.30e-05	1.40e-05	1.04e-04
32	1.32e-03	1.40e-04	1.84e-04	3.42e-04
64	6.33e-04	4.80e-05	6.00e-06	6.20e-05
128	3.30e-05	7.40e-05	5.80e-05	3.80e-05
256	2.00e-05	1.15e-04	1.13e-04	1.95e-04

Table 4.4: We tabulate absolute differences between fixed and naive DTMC (top) and CTMC (bottom) short-term test errors. For both types of models, we consider increasing state space dimension M and an increasing number of simulations. For all simulations, the training and test set lengths are, respectively, 1000 and 3000.

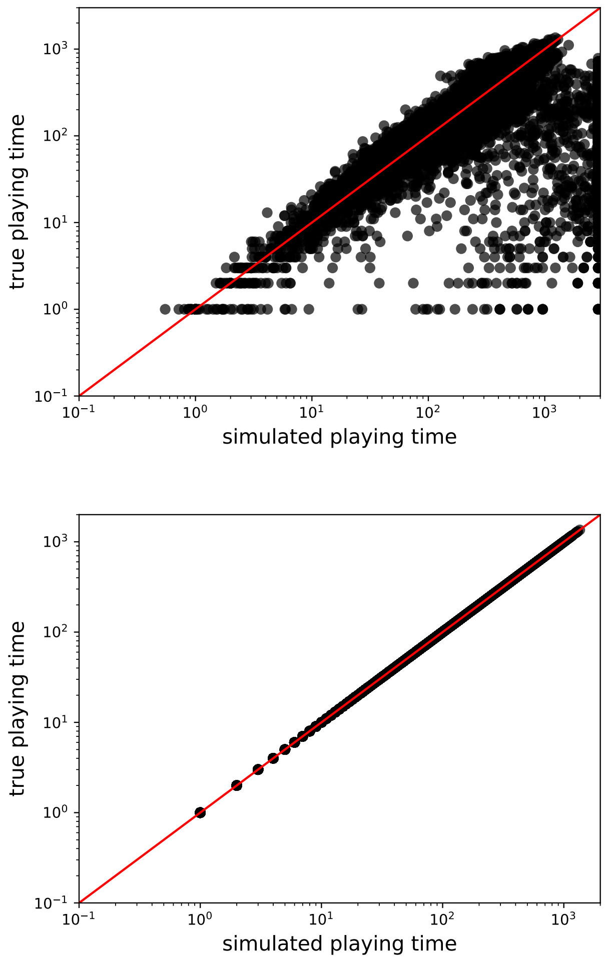


Figure 4.1: We plot simulated versus true (training set) playing times for naive (top) and fixed (bottom) CTMC models. Each point on each plot corresponds to one 5-person unit for one team. We see that the fixed model features practically zero training error, dramatically reducing the training error from the naive model.

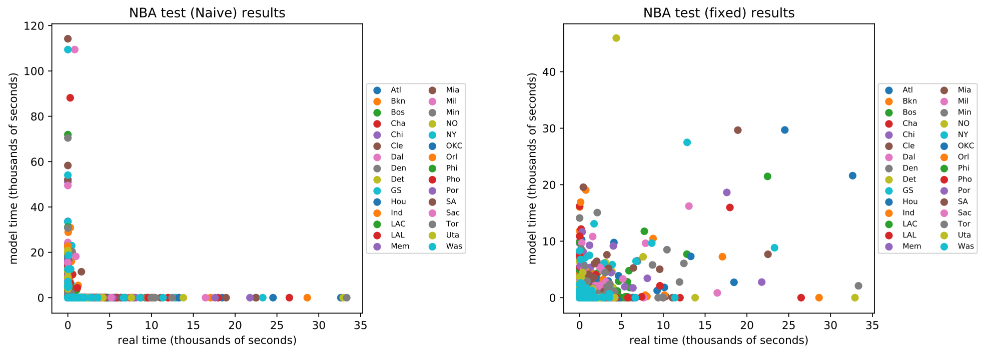


Figure 4.2: We plot naive (left) and fixed (right) CTMC test results using 40-game training sets. The fixed model decreases RMSE error by $\approx 60.4\%$. For further details, see Section 4.5.1.

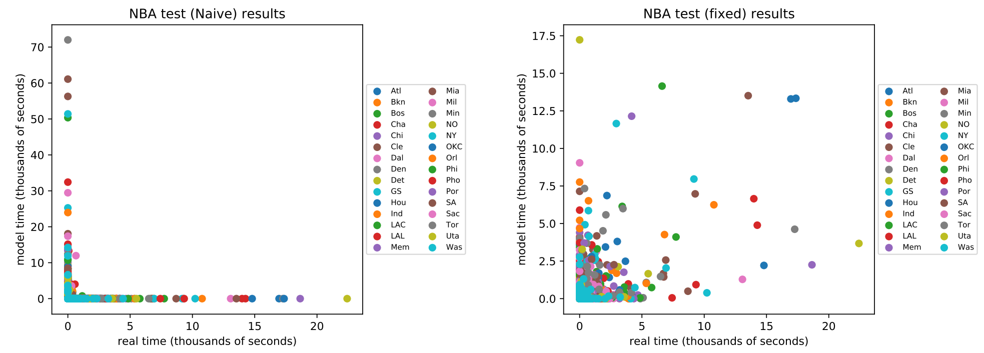


Figure 4.3: We plot naive (left) and fixed (right) CTMC test results using 60-game training sets. The fixed model decreases RMSE error by $\approx 61.9\%$. For further details, see Section 4.5.1.

	Dim	NNZ	CV	Dim	NNZ	CV
Team						
Atl	72092	240	2.167e-18	112560	274	4.445e-17
Bkn	67340	164	1.484e-18	105950	272	4.888e-18
Bos	90902	237	8.345e-19	114582	277	5.117e-18
Cha	35532	185	1.775e-18	68382	263	2.957e-18
Chi	54522	223	1.842e-18	151710	286	1.045e-17
Cle	72630	243	3.305e-18	119370	275	4.916e-18
Dal	110556	208	1.935e-18	213906	364	3.374e-18
Den	69432	144	8.410e-19	144020	344	2.740e-18
Det	13806	114	2.252e-18	35910	178	1.107e-18
GS	70490	170	2.479e-18	120756	286	8.207e-18
Hou	73170	263	2.690e-18	141000	2040	1.440e-17
Ind	58806	171	2.730e-18	118680	280	3.404e-17
LAC	55932	156	3.320e-18	97032	258	4.264e-18
LAL	43056	132	2.267e-18	68382	216	2.339e-18
Mem	59292	238	2.149e-18	181902	302	4.993e-18
Mia	63252	1256	6.223e-17	114582	1866	6.285e-17
Mil	73170	1241	2.534e-17	104652	237	3.334e-18
Min	57360	202	2.542e-18	93942	235	3.524e-18
NO	93942	193	2.229e-18	172640	2373	5.705e-17
NY	56882	183	3.909e-18	99540	267	4.578e-18
OKC	48180	190	1.604e-18	90300	234	1.641e-18
Orl	57840	235	1.326e-18	134322	306	8.248e-18
Phi	175980	280	9.187e-18	270920	389	4.804e-18
Pho	93330	171	2.320e-18	234740	328	3.836e-18
Por	34040	1046	7.235e-17	45582	236	1.085e-18
SA	69432	243	5.111e-18	141000	316	1.709e-17
Sac	66306	243	1.382e-18	102720	286	4.837e-18
Tor	30102	165	4.034e-18	46010	204	3.328e-18
Uta	113232	288	1.172e-17	199362	384	8.124e-18
Was	87912	259	2.558e-18	146306	317	4.125e-18

Table 4.5: For each team, we compute fixed CTMC models using training sets of either the first 40 (left of bar) or 60 (right of bar) non-overtime, regular season games. For each fixed model, we report Dim, the dimension of ε, equal to $M(M-1)$ where M is the number of states or unique 5-person units in that team's training set. We report NNZ, the number of nonzero entries in the computed ε—the small values of NNZ relative to Dim show that the computed solutions are highly sparse. Finally, we record CV, the maximum constraint violation reported by the optimizer—all values are close to zero.

Holson

1-norm Error	Naive	Fixed	HMM
Long-term Training	0.160545	1.665335e-16	0.000835
Long-term Test	0.235091	7.454545e-02	0.075380

Table 4.6: Long-term errors with training size 500 on Holson data.

Preproglucacon

1-norm Error	Naive	Fixed	HMM
Long-term Training	0.003426	1.942890e-16	0.000509
Long-term Test	0.113921	1.154179e-01	0.036629

Table 4.7: Long-term errors with training size 500 on preproglucacon data.

Training Time (seconds)

Data	Our models	HMM
Holson	0.71	54.68
Preproglucacon	0.05	28.40

Table 4.8: Training time comparison between our models and best-case HMM models on Holson and preproglucacon data sets

1-norm Error	2	3	4	5	6	7	8	9	10
Long-term Training	0.408995	0.464299	0.804931	0.827032	0.849197	0.933985	0.897961	0.832214	0.780222
Long-term Test	0.483541	0.538844	0.879477	0.901578	0.923742	1.004167	0.968143	0.902396	0.854767
Long-term Total	0.483527	0.538831	0.879463	0.901564	0.923729	1.004154	0.968130	0.902383	0.854754

Table 4.9: Long-term training, long-term test, and long-term total error for different internal states with 3 outputs. The HMM model is applied to the first 500 observations of Holson data with training 50 iterations. The best model of each internal state is chosen based on the maximum of negative log likelihood. The test set is the remaining data. Random seed is set at 7 for reproducibility. The training time was about 14,684 seconds, or 244.7 minutes.

1-norm Error	2	3	4	5	6	7	8	9	10
LT Training	0.000820	0.000795	0.000666	0.000681	0.000905	0.000650	0.000577	0.000723	0.000772
LT Test	0.035891	0.036242	0.036427	0.037149	0.036796	0.036576	0.036647	0.036425	0.036406
LT Total	0.035857	0.036208	0.036393	0.037115	0.036762	0.036541	0.036613	0.036391	0.036372

Table 4.10: Long-term training, long-term test and long-term total errors with with training size 500 on the preproglucacon data using HMM with parameter training size 100 and different internal state sizes. Best model of each internal state was chosen after training 10 times with different transition probability and emission probability matrix. Training time for all sizes of the internal states, including choosing the best models, was about 6,858 seconds (≈114 minutes).

Chapter 5

Conclusion

In Chapter 4, we showed that Markov chain optimization (MCO) achieved zero long-term training errors and greatly reduced long-term test errors compared to original maximum likelihood estimated (MLE) models for both discrete-time and continuous-time data. In the discrete-time data sets on patient life trajectories and protein DNA, we observed that MCO achieved reduced errors similar to more sophisticated hidden Markov models but with significantly reduced training time. In continuous-time data, we showed that we can predict game outcomes with roughly 70% accuracy using continuous-time Markov chain models (CTMC) for all National Basketball Association (NBA) teams. For each team we constructed a CTMC and inferred transition rates via maximum likelihood estimators (MLE) on observed data; that is, the rate at which state i transitions to state j is derived as the ratio of the total number of observed transitions from state i to state j and the total amount of time spent in state i. Note that the maximum likelihood estimates maximizes short-term accuracy.

The MLE derived transition rate matrix can then be used to simulate a game between two teams in turn. The team with the highest score in the simulation is the winner of the game. Consider a National Basketball Association (NBA) match. In a season, it is typical that a team plays a number of distinct 5-person units only once. Those units/lineups are called absorbing states in a CTMC. This results in a model equilibrium distribution supported on the set of absorbing states. In the long run, the equilibrium distribution will assign 48 minutes among these absorbing states. The absorbing state problem is resolved in our MCO work in Chapter 4.

Using the pre-MCO models we achieved a best model test accuracy of 73% by training on the 2014-15 NBA regular season data. Coupled with ad hoc strategies on absorbing Markov chains, we achieved a best model accuracy of 75% (training) and 70% (test). To achieve these results, we trained on the first 75 games of the 2015-16 regular season and tested on the rest of the regular season games.

The most recent work focused on capturing lineup playing time because maximum likelihood estimates give us transition rates that maximize observations of most seen outcomes. We consider this to be ideal for short-term game predictions; however, this accuracy comes at the expense of poorly capturing lineup playing time as seen in Chapter 3.

Using our models, we can predict a game outcome using the following two strategies:

1. Short-term based: Simulate a game between two teams. The team with highest end-game point differential is the winner of the game.

2. Long-term based: Multiply equilibrium distribution by 2880 seconds and point differential of each lineup contribution to obtain an estimate of game point differential. This strategy has not been explored in previous chapters but highlights the importance of capturing playing times of lineups accurately, which is accomplished in our MCO work in Chapter 4.

A natural question we can then ask is, "Can we further extract more information using our substitution models?"

On a fine scale, we can treat each 5-v-5 lineup configuration as a unique state to capture how two lineups fare against each other. Recall Table 3.1 in Chapter 3. We can view the first line of the table as a unique state. Currently our simulation does not account for which lineup plays against which lineup. We believe this strategy will yield better game prediction, e.g., game score differential and winning percentage. In the case where 5-v-5 state has not been observed in a match, accrue the lineup usage in the transition rate matrix. If we are interested in predicting the outcomes of games between two particular teams, we would train one CTMC model using past games between those two teams. The number of 5-v-5 states in this model will likely be in the hundreds. If there are absorbing states, we can apply MCO; as noted in Chapter 4, MCO can efficiently train CTMC models with hundreds of states. To actually perform the simulation, we would apply either the alarm clock or equilibrium vector methods described above. Either way, we would obtain an estimate of how much time each 5-v-5 lineup configuration spends on the court. Using plus/minus rates estimated for each 5-v-5 lineup, we can then determine the overall winner of the game.

One major problem with this approach is that, in a regular season, two teams play fewer than five games. The 5-v-5 state approach yields reduction in states but because so few games are played between two teams, we have few observations to infer true transition rates using MLE. Our current approach infers transition rates without regard to which teams a particular team has played. One potential fix is to include games of teams who are relatively similar to the opposing team. In this way, we obtain more reasonable transition rates, possibly yielding better game predictions.

For example, consider a matchup between the Golden State Warriors and Cleveland Cavaliers. In the 2017-18 season, the two teams only played twice against each other during the regular season prior to the championship series. Since the top 5 teams (including the Cavaliers) in the Eastern Conference had similar or better win-loss standings than the Cavaliers, we can include the games the Warriors played against those teams. In this way, we minimize the effect of lineups with outlier plus/minus rates that played little time on the court.

On a more subtle scale that can contribute to coaching decisions, we can ask, "What are the strengths among lineups?" Let us return to the current model in which lineups are treated as single units. We can ask, "Are there particular lineups that are greater than the sum of their individual players?" In other words, we can try to identify whether particular players—either superstars or bench players—lead to a synergistic effect with other players on the court. We can also ask questions such as: "If a particular lineup is unavailable, what is the best lineup to use in its place?” Again, the answers to such questions are of much greater potential interest to teams and coaches than they are to gamblers.

On a similar note, since basketball is a physical sport where players are prone to injury, we can ask what is the likelihood of a team winning a game if certain key players are unavailable to play. In Chapter 2, for example, we simulated playoff games when Blake Griffin and Chris Paul of the Los Angeles Clippers were injured. We can answer such hypothetical question and make game predictions of a team missing key players for team strength evaluation and/or ranking.

One natural extension of our current work is applying our MCO work from Chapter 4 to games in the women’s National Basketball Association (WNBA) and the National Collegiate Athletic Association (NCAA). Both have play-by-play data that can be found on public websites such as ESPN. Similar to our NBA models, a challenge is picking up player substitutions. In a WNBA game, the word “enters” is used instead of “substitute.” In an NCAA play-by-play game log on ESPN, the words “substitute” and “enters” cannot be found.

Aside from basketball, we can potentially extend our MCO work to the National Hockey League (NHL) using play-by-play data. In the NHL, a game is played by 5-v-5 lineups in three 20-minute quarters. The objective of a team in the game is to outscore its opponent by putting the puck into the opponent’s goal. Substitutions occur frequently; usually, scores by both teams are low because of strong defense. If the game is tied after 60 minutes, a 5-minute three-on-three match between the two teams ensues. The team to first score a goal is the winner of the game. A tied overtime match results in a shootout in which three players from each team take turns to take a penalty shot. The team with the most goals scored is the winner of the match.

Similar to the NBA, teams in the NHL also play 82 games; eight teams from each conference play in the playoffs. Each series is also best-of-seven games. A team with better standing is awarded home-ice advantage in playoffs where four games of a series are played at this team’s ice rink. In this way, strengths of lineups do play an important role in determining a team’s success. Though individual player substitution occurs in a game, it is more typical that two or more players are substituted in one setting.

Nevertheless, for a more accurate prediction of game score differential, we need to be able to capture playing times of lineups as well as how they score. In Chapter 2 and 3, we used plus/minus rates to help predict game outcomes assuming lineups scored linearly and players did not get tired. Consequently, we consistently observed large positive or negative score margins in a game simulation. A potential fix mentioned in Chapter 3 is to seek stochastic scoring models for all lineups of all teams. Another approach to game prediction involves training neural networks on data consisting of multiple features. This enables the neural networks to learn interactions among features without requiring any feature extraction or feature engineering. We can architect neural networks to output predictions as functions of quantities such as time played by players and lineups, points scored by those players and lineups, and the number of fouls committed by players.

Overall, if we include potential improvements as discussed in Chapter 4, we see great scope for this work to be usefully applied in many settings. This includes problems in sports analytics as well as other areas where Markov chains intersect with large, detailed data sets.

Bibliography

Andersen, M. S., Dahl, J., and Vandenberghe, L. (2018). CVXOPT: A Python package for convex optimization, version 1.2.0. http://cvxopt.org.

ApS, M. (2018). *The MOSEK optimization toolbox for Python manual. Version 8.1.0.53.*

Backstrom, L. and Leskovec, J. (2011). Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 635–644.

Bai, Z.-J., Serra-Capizzano, S., and Zhao, Z. (2012). Nonnegative inverse eigenvalue problems with partial eigendata. *Numerische Mathematik*, **120**(3), 387–431.

Barbu, V. S. and Limnios, N. (2009). *Semi-Markov chains and hidden semi-Markov models toward applications: their use in reliability and DNA analysis*, volume 191. Springer Science & Business Media.

Bhat, H. S., Huang, L.-H., and Rodriguez, S. (2015). Learning stochastic models for basketball substitutions from play-by-play data. In *MLSA Workshop at ECML/PKDD 2015.*

Cervone, D., DAmour, A., Bornn, L., and Goldsberry, K. (2016). A multiresolution stochastic process model for predicting basketball possession outcomes. *Journal of the American Statistical Association*, **111**(514), 585–599.

Chartier, T. P., Kreutzer, E., Langville, A. N., and Pedings, K. E. (2011). Sensitivity and stability of ranking vectors. *SIAM J. Sci. Comput.*, **33**(3), 1077–1102.

Chen, X. and Liu, D. (2011). Isospectral flow method for nonnegative inverse eigenvalue problem with prescribed structure. *Journal of Computational and Applied Mathematics*, **235**(14), 3990–4002.

Chu, M. and Golub, G. H. (2005). *Inverse Eigenvalue Problems: Theory, Algorithms, and Applications*, volume 13. Oxford University Press.

Chu, M. T. and Guo, Q. (1998). A numerical method for the inverse stochastic spectrum problem. *SIAM Journal on Matrix Analysis and Applications*, **19**(4), 1027–1039.

Crawford, F. W., Minin, V. N., and Suchard, M. A. (2014). Estimation for general birth-death processes. *Journal of the American Statistical Association*, **109**(506), 730–747.

Csáji, B. C., Jungers, R. M., and Blondel, V. D. (2014). PageRank optimization by edge selection. *Discrete Applied Mathematics*, **169**, 73–87.

D'Amour, A., Cervone, D., Bornn, L., and Goldsberry, K. (2015). Move or die: How ball movement creates open shots in the NBA. In *MIT Sloan Sports Analytics Conference*.

Deshpande, S. K. and Jensen, S. T. (2016). Estimating an NBA players impact on his teams chances of winning. *Journal of Quantitative Analysis in Sports*, **12**(2), 51–72.

Fercoq, O. (2014). Perron vector optimization applied to search engines. *Applied Numerical Mathematics*, **75**, 77–99.

Fercoq, O., Akian, M., Bouhtou, M., and Gaubert, S. (2013). Ergodic control and polyhedral approaches to PageRank optimization. *IEEE Transactions on Automatic Control*, **58**(1), 134–148.

Fewell, J., Armbruster, D., Ingraham, J., Petersen, A., Waters, J., and Boccaletti, S. (2012). Basketball teams as strategic networks. *PLoS ONE*, **7**(11), e47445.

Fraser, A. M. (2008). *Hidden Markov Models and Dynamical Systems*. SIAM, Philadelphia.

Gabel, A., Redner, S., *et al.* (2012). Random walk picture of basketball scoring. *Journal of Quantitative Analysis in Sports*, **8**(1), 1416.

Gleich, D. F. (2015). PageRank Beyond the Web. *SIAM Review*, **57**(3), 321–363.

Gómez, M.-Á., Silva, R., Lorenzo, A., Kreivyte, R., and Sampaio, J. (2016). Exploring the effects of substituting basketball players in high-level teams. *Journal of Sports Sciences*, pages 1–8.

Guttorp, P. (1995). *Stochastic Modeling of Scientific Data*. Chapman & Hall/CRC.

Hajiaghayi, M., Kirkpatrick, B., Wang, L., and Bouchard-Côté, A. (2014). Efficient continuous-time Markov chain estimation. In *Proceedings of ICML 2014*, pages 638–646.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, second edition.

Jones, M. *et al.* (2007). Home advantage in the NBA as a game-long process. *Journal of Quantitative Analysis in Sports*, **3**(4), 2.

Konstantopoulos, T. (2006). Notes on survival models. *Edinburgh: Heriot-Watt University*.

Kubatko, J., Oliver, D., Pelton, K., and Rosenbaum, D. T. (2007). A starting point for analyzing basketball statistics. *Journal of Quantitative Analysis in Sports*, **3**(3).

Kumar, R., Tomkins, A., Vassilvitskii, S., and Vee, E. (2015). Inverting a steady-state. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 359–368.

Kvam, P. and Sokol, J. S. (2006). A logistic regression/Markov chain model for NCAA basketball. *Naval Research Logistics*, **53**, 788–803.

Langville, A. N. and Meyer, C. D. (2006a). *Google's PageRank and Beyond: the Science of Search Engine Rankings*. Princeton University Press, Princeton, NJ.

Langville, A. N. and Meyer, C. D. (2006b). Updating Markov chains with an eye on Google's PageRank. *SIAM Journal on Matrix Analysis and Applications*, **27**(4), 968–987.

Laurie, D. P. (1991). Solving the inverse eigenvalue problem via the eigenvector matrix. *Journal of Computational and Applied Mathematics*, **35**, 277–289.

Lawler, G. F. (2006). *Introduction to Stochastic Processes*. Chapman & Hall/CRC, Boca Raton.

Liu, X. L. (2007). *Bayesian analysis of Dyadic data arising in basketball*. Master's thesis, Simon Fraser University.

Maystre, L. and Grossglauser, M. (2015). Fast and accurate inference of Plackett–Luce models. In *Advances in Neural Information Processing Systems 28*, pages 172–180.

Metzner, P., Dittmer, E., Jahnke, T., and Schütte, C. (2007a). Generator estimation of Markov jump processes. *Journal of Computational Physics*, **227**, 353–375.

Metzner, P., Dittmer, E., Jahnke, T., and Schütte, C. (2007b). Generator estimation of Markov jump processes. *Journal of Computational Physics*, **227**(1), 353–375.

Ng, A. Y., Zheng, A. X., and Jordan, M. I. (2001). Link analysis, eigenvectors and stability. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 903–910.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York.

Oh, M.-H., Keshri, S., and Iyengar, G. (2015). Graphical model for basketball match simulation. In *MIT Sloan Sports Analytics Conference*.

Opper, M. and Sanguinetti, G. (2007). Variational inference for Markov jump processes. In *Advances in NIPS 20*, pages 1105–1112.

Peuter, C. D. (2013). *Modeling Basketball Games as Alternating Renewal-Reward Processes and Predicting Match Outcomes*. Master's thesis, Duke University.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), 257–286.

Rao, V. and Teh, Y. (2013). Fast MCMC sampling for Markov jump processes and extensions. *Journal of Machine Learning Research*, **14**, 3295–3320.

Ribeiro, H. V., Mukherjee, S., and Zeng, X. H. T. (2016). The advantage of playing home in NBA: Microscopic, team-specific and evolving features. *PloS one*, **11**(3), e0152440.

Shi, Z., Moorthy, S., and Zimmermann, A. (2013). Predicting NCAAB match outcomes using ML techniques—some results and lessons learned. In *Proceedings of the MLSA Workshop at ECML/PKDD 2013*.

Shirley, K. (2007). A Markov model for basketball. In *New England Symposium for Statistics in Sports*, Boston, MA.

Shortridge, A., Goldsberry, K., and Adams, M. (2014). Creating space to shoot: quantifying spatial relative field goal efficiency in basketball. *Journal of Quantitative Analysis in Sports*, **10**(3), 303–313.

Spedicato, G. A., Kang, T. S., Yalamanchi, S. B., and Yadav, D. (2017). *The markovchain Package: A Package for Easily Handling Discrete Markov Chains in R*.

Swartz, T. B., Tennakoon, A., Nathoo, F., Tsao, M., Sarohia, P., *et al.* (2011). Ups and downs: team performance in best-of-seven playoff series. *Journal of Quantitative Analysis in Sports*, **7**(4), 1–17.

Štrumbelj, E. and Vračar, P. (2012). Simulating a basketball match with a homogeneous markov model and forecasting the outcome. *International Journal of Forecasting*, **28**(2), 532–542.

Wu, S.-J. and Chu, M. T. (2015). Constructing optimal transition matrix for Markov chain Monte Carlo. *Linear Algebra and its Applications*, **487**, 184–202.

Xin, L., Zhu, M., Chipman, H., *et al.* (2017). A continuous-time stochastic block model for basketball networks. *The Annals of Applied Statistics*, **11**(2), 553–597.

Yao, T.-T., Bai, Z.-J., Zhao, Z., and Ching, W.-K. (2016). A Riemannian Fletcher–Reeves conjugate gradient method for doubly stochastic inverse eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, **37**(1), 215–234.