Predicting NBA Games with Matrix Factorization

by
Tuan Tran
B.S., Massachusetts Institute of Technology (2015)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering in Computer Science and Engineering
at the
Massachusetts Institute of Technology
June 2016

Author ……………………………………………………………………………………………..
Department of Electrical Engineering and Computer Science
May 20, 2016

Certified by ………………………………………………………………………………………..
Regina Barzilay, Professor
Thesis Supervisor

Accepted by ……………………………………………………………………………………..
Dr. Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Predicting NBA Games with Matrix Factorization

by
Tuan Tran

Submitted to the
Department of Electrical Engineering and Computer Science
May 20, 2016
in partial fulfillment of the requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

In my thesis, I present the methods I use to predict NBA games using matrix factorization. Matrix factorization is popular through the Netflix recommendation problem, but in general, one can apply it to data that are best modeled as the result of pairwise interaction. My thesis contains three parts. First, I explain how I model NBA prediction as a matrix factorization problem and use the basic low-rank matrix factorization approach to discover structure in the data. I also explain some differences between using matrix factorization for NBA prediction versus that in the Netflix recommendation problem. Second, I use probabilistic matrix factorization (PMF) to incorporate the fact that when two teams play each other, the scores will be different each time. Lastly, I incorporate supplementary information such as the date of the game by combining multiple PMF problems using Gaussian process priors. I replace the scalar latent features with functions of this supplementary information to aid with prediction.

Acknowledgements

I would like to thank my advisor, Professor Regina Barzilay, for the support of my work. She was my departmental advisor throughout my undergraduate years at MIT and I also had the pleasant opportunity of working with her as a teaching assistant for Introduction to Machine Learning (6.036) this semester.

I want to express my utmost gratitude to my parents who have cared for me and raised me to be who am I today ever since we immigrated from Vietnam to the United States.

I also want to express my gratitude to my family members in California and Vietnam, and friends who supported me throughout life and my years of study. This accomplishment would never have been possible without all of them.

Contents

1. Introduction…………………………………………………………………….....9

1.1. Data Available……………………………………...………………...10

1.1.1. Box-score Data……………………………………………...............10

1.1.2. Play-by-play Data..12

1.1.3. Camera-tracked Data...14

1.2. Previous Work……………………………………..........………......14

1.3. Modeling with Matrix Factorization……………...............……………………15

1.4. Data Format...16

2. Basic Low-Rank Matrix Factorization…………………………………………..17

2.1. Matrix Factorization Review……..……………………………………......18

2.2. Adapting to NBA Games...………………………………………………....19

2.3. Results..………………………………………………….........................20

3. Probabilistic Matrix Factorization..…………………………………………......23

4. PMF Incorporating Supplementary Information………………………………......27

4.1. Summary of DPMF……...…………………………………………….........27

4.2. DPMF for NBA Prediction..…...……………………………………..…....31

5. Conclusion…..…………………..…………………………………………………..34

5.1. Future Work…...…………...……………………………………………………35

6. References…..…………………..………………………………………………..37

List of Figures and Tables

Figure 1: Box score data for a team. Source: NBA.com..12

Figure 2: Play-by-play data for two competing teams. Source: NBA.com..................................14

Table 1: Best training and test accuracy when using the training set with scores averaging........23

Figure 3: Graphical model for PMF..26

Table 2: Percentage errors of winner predictions using DPMFs..34

1. Introduction

The National Basketball Association (NBA) is the premier men's professional basketball league in North America. The NBA consists of players not only from North America, but also from different countries worldwide. Therefore, the NBA's influence permeates around the world. Since the NBA is a multi-billion dollar industry and involves millions of fans, many studies have been conducted to try to predict game outcomes based on the large amount of data and statistics available. Moreover, in general, predicting the results of sporting events is a natural application of machine learning. It is also particularly well-suited for the NBA because we can measure players' performances across different positions using the same set of statistics (ie. points, rebounds, steals, assists, blocks, etc.).

Trivially, by looking at teams' previous records of win to loss ratio, we can somewhat infer how good one team will be when playing against an opponent team. However, predicting basketball outcomes is very complicated and tricky because there are many factors that affect the results. These factors are, for example, whether a team has a superstar player like LeBron James, whether a team has an exceptional coach like Gregg Popovich, or whether a team has home court advantage. On December 9th, 2004, the Houston Rockets played at home versus the San Antonio Spurs. Down 76-68 with 42 seconds left, the Rockets seemed very unlikely to win. However, its superstar player Tracy McGrady miraculously scored 13 points in 33 seconds to allow the Rockets to defeat the Spurs 80-81 [1]. Situations like this make it tricky to predict the outcomes. Moreover, what makes it especially difficult is the fact that the score during a game can change very quickly. Even a fraction of a second makes a difference, as can be seen in game 5 of the 2004 Western Conference Semifinals when Derek Fisher scored a two-pointer with only 0.4

seconds remaining to allow the Los Angeles Lakers to defeat the San Antonio Spurs 74-73 [2]. This quick change in score is unlike other sports such as soccer, football, or baseball where the average time to score is much higher than that for basketball.

The majority of previous work on NBA prediction involves assembling features for each game and using these feature vectors as input into some machine learning algorithm like logistic regression, support vector machine (SVM), neural networks, or Naive Bayes. Most features are obtained from the readily available game statistics such as points or rebounds. In section 1.1, I describe the types of data available that can be used as features. In section 1.2, I summarize the majority of the previous work that has been done. These previous work mainly focus on engineering new features by combining different game statistics. In section 1.3, I discuss how matrix factorization is a more interesting and suitable approach to the problem of predicting game outcomes.

In chapter 2 of my thesis, I describe modeling the problem using the basic low-rank matrix factorization technique. In chapter 3, I describe using probabilistic matrix factorization (PMF) to incorporate the fact that when two teams play each other, the scores will be different each time. In chapter 4, I incorporate supplementary information such as the date of the game by combining multiple PMF problems using Gaussian process priors. This involves replacing the scalar latent features with functions of this supplementary information.

1.1 Data Available

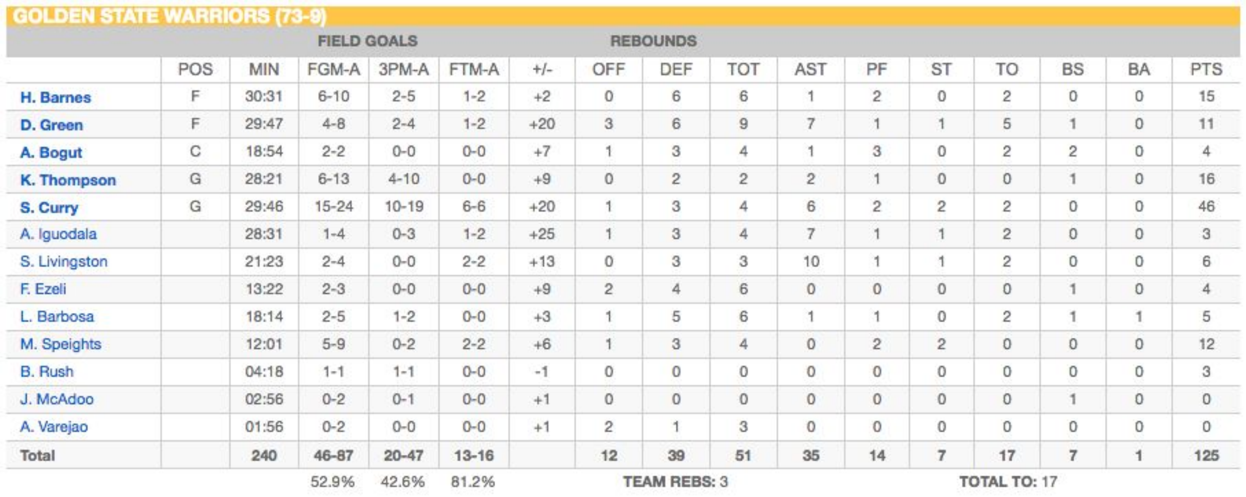
The NBA records 3 types of data:

- Box-score data
- Play-by-play data

- Camera-tracked data

1.1.1 Box-score Data

A box score contains the information about a game between two teams. Here is an example box score (from NBA.com) for one of the teams in a game. The opponent team's box score has the same format.



GOLDEN STATE WARRIORS (73-9)

			FIELD GOALS				REBOUNDS									
	POS	MIN	FGM-A	3PM-A	FTM-A	+/-	OFF	DEF	TOT	AST	PF	ST	TO	BS	BA	PTS
H. Barnes	F	30:31	6-10	2-5	1-2	+2	0	6	6	1	2	0	2	0	0	15
D. Green	F	29:47	4-8	2-4	1-2	+20	3	6	9	7	1	1	5	1	0	11
A. Bogut	C	18:54	2-2	0-0	0-0	+7	1	3	4	1	3	0	2	2	0	4
K. Thompson	G	28:21	6-13	4-10	0-0	+9	0	2	2	2	1	0	0	1	0	16
S. Curry	G	29:46	15-24	10-19	6-6	+20	1	3	4	6	2	2	2	0	0	46
A. Iguodala		28:31	1-4	0-3	1-2	+25	1	3	4	7	1	1	2	0	0	3
S. Livingston		21:23	2-4	0-0	2-2	+13	0	3	3	10	1	1	2	0	0	6
F. Ezeli		13:22	2-3	0-0	0-0	+9	2	4	6	0	0	0	0	1	0	4
L. Barbosa		18:14	2-5	1-2	0-0	+3	1	5	6	1	1	0	2	1	1	5
M. Speights		12:01	5-9	0-2	2-2	+6	1	3	4	0	2	2	0	0	0	12
B. Rush		04:18	1-1	1-1	0-0	-1	0	0	0	0	0	0	0	0	0	3
J. McAdoo		02:56	0-2	0-1	0-0	+1	0	0	0	0	0	0	0	1	0	0
A. Varejao		01:56	0-2	0-0	0-0	+1	2	1	3	0	0	0	0	0	0	0
Total		**240**	**46-87**	**20-47**	**13-16**		**12**	**39**	**51**	**35**	**14**	**7**	**17**	**7**	**1**	**125**
			52.9%	42.6%	81.2%			**TEAM REBS:** 3					**TOTAL TO:** 17			

Figure 1: Box score data for a team. Source: NBA.com

The box score records, for each player, the following statistics:

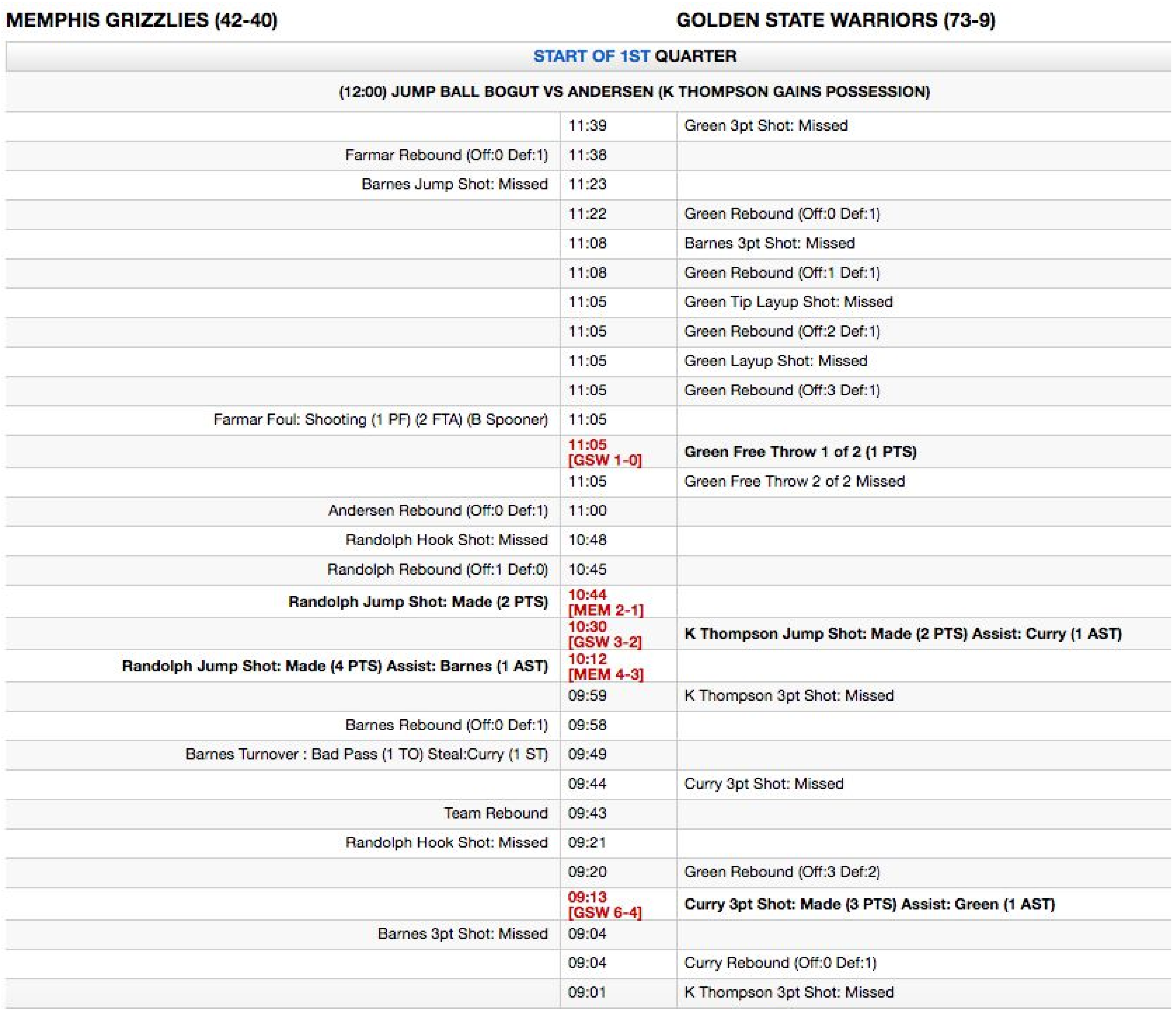
- The number of minutes played (MIN).
- The number of field goals made and attempted (FGM-A).
- The number of 3-point field goals made and attempted (3PM-A).
- The number of free throws made and attempted (FTM-A).
- The plus/minus score which determines how the team performs with that player on the floor (+/-).
- The number of offensive rebounds (OFF).
- The number of defensive rebounds (DEF).
- The number of total rebounds (TOT).

- The number of assists (AST).
- The number of personal fouls (PF).
- The number of steals (ST).
- The number of turnovers (TO).
- The number of times that player blocked a shot (BS).
- The number of times that player had a shot blocked against him (BA).
- The number of points (PTS).

The first five player names are bolded in blue to indicate that they are the starting players. The box score also includes information at the team level. We can see the total statistics, such as total number of points scored, the percentages for field goal, 3-point field goal, and free throw, and team rebounds, which are rebounds allocated to the team as a whole and not specifically to one player. The official site of the NBA [3] contains box score data from the 1946-1947 season to present, although it is very rare to use or require data that further back because the rules, teams, and overall league have changed and evolved over time. Another popular source is Basketball Reference [4], which contains box score data from the 1985-1986 season to present, albeit in a slightly different format.

1.1.2 Play-by-play Data

Play-by-play data is similar to box score data, except it is more detailed. Specifically, it tracks every play and timestamps it so that you know exactly when each play occurs.



MEMPHIS GRIZZLIES (42-40)		GOLDEN STATE WARRIORS (73-9)
	START OF 1ST QUARTER	
	(12:00) JUMP BALL BOGUT VS ANDERSEN (K THOMPSON GAINS POSSESSION)	
	11:39	Green 3pt Shot: Missed
Farmar Rebound (Off:0 Def:1)	11:38	
Barnes Jump Shot: Missed	11:23	
	11:22	Green Rebound (Off:0 Def:1)
	11:08	Barnes 3pt Shot: Missed
	11:08	Green Rebound (Off:1 Def:1)
	11:05	Green Tip Layup Shot: Missed
	11:05	Green Rebound (Off:2 Def:1)
	11:05	Green Layup Shot: Missed
	11:05	Green Rebound (Off:3 Def:1)
Farmar Foul: Shooting (1 PF) (2 FTA) (B Spooner)	11:05	
	11:05 [GSW 1-0]	**Green Free Throw 1 of 2 (1 PTS)**
	11:05	Green Free Throw 2 of 2 Missed
Andersen Rebound (Off:0 Def:1)	11:00	
Randolph Hook Shot: Missed	10:48	
Randolph Rebound (Off:1 Def:0)	10:45	
Randolph Jump Shot: Made (2 PTS)	10:44 [MEM 2-1]	
	10:30 [GSW 3-2]	**K Thompson Jump Shot: Made (2 PTS) Assist: Curry (1 AST)**
Randolph Jump Shot: Made (4 PTS) Assist: Barnes (1 AST)	10:12 [MEM 4-3]	
	09:59	K Thompson 3pt Shot: Missed
Barnes Rebound (Off:0 Def:1)	09:58	
Barnes Turnover : Bad Pass (1 TO) Steal:Curry (1 ST)	09:49	
	09:44	Curry 3pt Shot: Missed
Team Rebound	09:43	
Randolph Hook Shot: Missed	09:21	
	09:20	Green Rebound (Off:3 Def:2)
	09:13 [GSW 6-4]	**Curry 3pt Shot: Made (3 PTS) Assist: Green (1 AST)**
Barnes 3pt Shot: Missed	09:04	
	09:04	Curry Rebound (Off:0 Def:1)
	09:01	K Thompson 3pt Shot: Missed

Figure 2: Play-by-play data for two competing teams. Source: NBA.com

With play-by-play data, you know not only how many points or rebounds a player has from the box score data, but also when he scored those points or when he grabbed those rebounds. Other things we learn from the play-by-play data that we can not obtain from box score data are, for example, when player substitutions are made, when timeouts are called, who blocked whom or who assisted whom, the type of field goal attempted (three-pointer, layup, dunk, etc.), the fouls drawn, and the times when possessions change.

The most important characteristic of play-by-play data is that it provides context for events that occur. Knowing which players are on the court at different times allows us to determine the value of having different combinations of players or the value that individual players provide.

1.1.3 Camera-tracked Data

Camera-tracked data contains the (x, y, z) location of every player on the court and the ball relative to a fixed point. This information is updated 25 times a second. As of May 2013, 15 of the 30 NBA teams kept track of this data [5], but by now, every team is doing this.

This data offers much finer granularity than box score data or play-by-play data. The downside of using this data is that it only became available starting in 2010 and by 2013, only half of the teams in the NBA kept track of it. Therefore, it is incomplete and there is not much data to work with. Also, the data is very low-level; you have to convert the data into a higher level format in order to do something useful with it and doing this conversion is a challenge and another project by itself. Lastly, the data is not publicly available; you must purchase it [6].

1.2 Previous Work

The field of basketball analytics involves descriptive and predictive analytics. Most of traditional analysis are descriptive, focusing on describing what has already happened, for example, evaluating players or teams. There are many descriptive models, but the majority of them can be categorized as follows:

1. Team based analysis: analyzes factors such as points scored per possession and points allowed per possession that motivate winning at a team level.

2. Box score based analysis: looks at each player's box scores and assigns values to each of the individual box score statistics in order to judge a player's value on a per-minute basis.
3. Plus-minus based analysis: analyzes the number of points a player's team scores against the number of points the opposing team scores while that player is on the floor.

Most of the predictive models attempt to predict the outcome of one game because there are 1230 games in one season to work with. This is different from trying to predict, for example, the season's most valuable player (MVP) in which there is only one per year. Moreover, most models simply apply machine learning techniques such as logistic regression, SVM, neural networks, or Naive Bayes to statistics in the box score data. For example, one model uses linear regression on box-score data from 1992 to 1996 to predict 69% of the games correctly [7]. Some models also attempt to use methods for feature selections or come up with new, interesting features.

1.3 Modeling with Matrix Factorization

As mentioned, most of the approaches to NBA prediction involve designing some features for each game and then training some off the shelf binary classifier that outputs a label indicating whether the home team or the away team won. This approach is reasonable, but it tries to fit the problem into a typical format instead of trying to design a solution that suits the problem. An important thing to note about a basketball game is that it is an interaction between two teams. Therefore, it makes more sense to learn features for teams and directly model the interactions between teams rather than engineer or learn features of games.

Many data are best modeled as the result of pairwise interactions. These interactions are typically between items from different sets, but can be between items from the same set. The most noticeable aspect is that the observations are the result of interactions. For example, in the Netflix Prize example, you are given user/movie ratings and must predict the ratings of unseen pairs. We can model the pairwise interactions by treating the observations as a matrix and then using matrix factorization to discover structure in the data. Therefore, we can apply this technique for modeling interactions between two teams in an NBA game. In this context, each entry (i, j) in the matrix corresponds to team i's score when playing against team j and vice-versa for entry (j, i).

1.4 Data Format

In my project, I have access to regular season games from the 1985-1986 season to the 2014-2015 season as training data. For each game, I have the date (year, month, day, time), home team name, away team name, home team score, and away team score. Each team's full name (i.e. Los Angeles Lakers) is a combination of the team's city (Los Angeles) and the team's name (Lakers). Because team cities and names change throughout the years, I remove the team cities and keep only the team names to have the data in a consistent format. For example, I use only "Lakers" rather than "Los Angeles Lakers". Also, the same team today may have used different names in the past. I format my data so that every game lists the two teams' current names. Here are the important naming issues:

- New Jersey Nets changed to Brooklyn Nets (use Nets)
- Vancouver Grizzlies changed to Memphis Grizzlies (use Grizzlies)
- Washington Bullets changed to Washington Wizards (use Wizards)
- Toronto Raptors was formed in 1995
- Minnesota Timberwolves was formed in 1989

For the games in the following seasons, the team names are listed as follows, but use the current name (Pelicans):

- 1988-2002 Charlotte Hornets
- 2002-2005 New Orleans Hornets
- 2005-2007 New Orleans/Oklahoma City Hornets
- 2007-2013 New Orleans Hornets
- 2013-Present New Orleans Pelicans

For the games in the following seasons, the team names are listed as follows, but use the current name (Hornets):

- 2004-2014 Charlotte Bobcats
- 2014-Present Charlotte Hornets

2. Basic Low-Rank Matrix Factorization

The most popular application of matrix factorization is the Netflix Prize example in which there is a set of users and a set of movies. The data contains tuples (u, m, r) which specifies the rating r that user u gives to movie m and the goal is to predict the ratings for the movies that users have not yet rated. By placing this data in a matrix, where each row corresponds to one user and each column corresponds to one movie, we have that each entry corresponds to a user's rating of a movie. Although user-based or item-based collaborative filtering methods are simple and intuitive, matrix factorization techniques are usually more effective because they allow us to discover the latent features underlying the interactions between users and movies. The technique is simply a mathematical tool that can be used in scenarios where one would like to find out something hidden under the data.

2.1. Matrix Factorization Review

The basic idea behind matrix factorization is to find two matrices such that multiplying them gives you back the original matrix. The intuition behind using matrix factorization is that there should be some latent features that determine how a user rates a movie. For example, two users would rate a movie similarly if they both like the actors or actresses of the movie or if the movie is of a genre preferred by both users. Therefore, if we can discover these latent features, we should be able to use them to predict a user's rating of a movie that he has not yet rated because the features associated with the user should match with the features associated with the movie.

Mathematically, we have a set M of users, and a set N of movies. R is the matrix containing all ratings that the users have assigned to the movies and we have K latent features that we want to discover. The task is to find two matrices U (a $|M| \times K$ matrix) and V (a $|N| \times K$ matrix) such that their product approximates R.

$$R \approx U \times V^T = \hat{R}$$

Therefore, each row of U represents the strength of the associations between a user and the latent features and each row of V represents the strength of the associations between a movie and the latent features. To get the predicted rating of a movie n by user m, we calculate the dot product of the two vectors corresponding to m and n:

$$\hat{r}_{m,n} = u_m v_n^T = \sum_{k=1}^{K} u_{m,k} \cdot v_{n,k}$$

2.2. Adapting to NBA Games

When modeling NBA games with matrix factorization, the interactions are no longer between users and movies, but between one team and another. I adopt the following approach: I assume that each team has two latent vectors describing how good its offense and defense are in certain attributes [8]. For example, since there are five positions in the NBA: point guard, shooting guard, small forward, power forward, and center, one possible representation of the latent vectors is how good the players of each position on each team are in terms of offense and defense. For example, a team's offensive and defensive vector, listed in order of the positions mentioned, can be:

Offense: $\left[5, 3, 6, 7, 9\right]$

Defense: $\left[1, 3, 8, 4, 6\right]$

This indicates that the point guards have a score of 5 on offense and 1 on defense. In this model, I assume that higher numbers are better for offense and lower numbers are better for defense. For a game between team i and team j, team i's score is the dot product of team i's offensive vector and team j's defensive vector and vice versa. Team i scores more points if its offensive vector has higher values (better offense) and if team j's defensive vector has higher numbers (lower defense).

To learn these offensive and defensive vectors, I find latent matrices U (offense) and V (defense) that minimize the sum of squared error between the predicted scores and observed scores for all games in the training set. The total error for a game between team i and team j is:

$$e_{ij} = (s_i - \sum_{k=1}^{K} U_{ik} \cdot V_{jk})^2 + (s_j - \sum_{k=1}^{K} U_{jk} \cdot V_{ik})^2 + \lambda \cdot (\sum_{k=1}^{K} U_{ik}^2 + U_{jk}^2 + V_{ik}^2 + V_{jk}^2)$$

Let $\delta_i = (s_i - \sum_{k=1}^{K} U_{ik} \cdot V_{jk})^2$, $\delta_j = (s_j - \sum_{k=1}^{K} U_{jk} \cdot V_{ik})^2$

This error is equal to the sum of the squared difference between team i's actual score and predicted score, the squared difference between team j's actual score and predicted score and the regularized entries in both team's offensive and defensive vectors. Performing gradient descent results in the following updates to the offensive and defensive vectors:

$U_{ik}\prime = U_{ik} + 2 \cdot \alpha \cdot (\delta_i \cdot V_{jk} - \lambda \cdot U_{ik})$

$U_{jk}\prime = U_{jk} + 2 \cdot \alpha \cdot (\delta_j \cdot V_{ik} - \lambda \cdot U_{jk})$

$V_{ik}\prime = V_{ik} + 2 \cdot \alpha \cdot (\delta_j \cdot U_{jk} - \lambda \cdot V_{ik})$

$V_{jk}\prime = V_{jk} + 2 \cdot \alpha \cdot (\delta_i \cdot U_{ik} - \lambda \cdot V_{jk})$

2.3. Results

First, I use game results from the 1985-1986 season up until the 2014-2015 season as training data to learn the latent offensive vectors and defensive vectors. I do this for dimensions $K = 1, 2, .., 10$ for the latent vectors. I use the models to make predictions on the test set, the games for the current 2015-2016 season. The 4D model yields the best training accuracy at 0.5769 and the 3D and 5D models yields the best test accuracy at 0.5234. These accuracies, especially on the test set, are very low and not that much better than flipping a coin. This is because I use data all the way back from the 1985-1986 season. Between then and now, a lot of things have changed in terms of team composition, strengths, and weaknesses. Therefore, a team ranked very highly back then can be ranked poorly now. As the learning algorithm sees each game from the 1985-1986 season up until the 2014-2015 season, it makes adjustments to the corresponding two opposing teams' offensive and defensive vectors. This means that having outdated scores is not helpful in making predictions for the games in the current 2015-2016 season.

It makes sense that the most recent data is the most useful for predicting the games for the current 2015-2016 season because there are lower chances of changes to team compositions. So next, I use only the previous season, the 2014-2015 games as training data. The 10D model yields the best training accuracy at 0.7341 and the 1D model yields the best test accuracy at 0.6515. Restricting the training set improves the test accuracy by about 13%.

In the Netflix recommendation problem, each entry in the matrix corresponds to one user's rating of one movie. In my model for NBA games, entry (i, j) in the matrix corresponds to team i's score when team i plays against team j and entry (j, i) corresponds to team j's score. However, within one season, two teams can play each other more than once, and especially when considering games among multiple seasons, there are many matches between two teams. To handle this case so that entry (i, j) and entry (j, i) each has only one value, entry (i, j) holds team i's average number of points that team i scores against team j throughout the games they play in the training set used and vice versa for entry (j, i).

Since using the 2014-2015 games (most recent season) as the training set seems to give the best predictions for the 2015-2016 games, I first use the 2014-2015 games (with scores averaging) to learn the models. Then, I expand the training set to use seasons 2013-2015, 2012-2015, all the way back to 2000-2015, all with scores averaging. I stop at 2000 because

2000 is a nice number to use and the Toronto Raptors was formed in 1995 so using data after 1995 allows me to have every season include matches for all teams in the NBA. Here are my results:

Seasons (Training Data)	Best Training Accuracy & Model	Best Test Accuracy & Model
2000-2015	0.5732 (10D)	0.5531 (7D)
2001-2015	0.5744 (9D)	0.5647 (4D)
2002-2015	0.5783 (2D)	0.5734 (3D)
2003-2015	0.5863 (3D)	0.5743 (6D)
2004-2015	0.5908 (8D)	0.5830 (4D)
2005-2015	0.5912 (3D)	0.5898 (3D)
2006-2015	0.5955 (8D)	0.5994 (9D)
2007-2015	0.6002 (10D)	0.6062 (6D)
2008-2015	0.6075 (9D)	0.6178 (3D, 7D)
2009-2015	0.6194 (9D)	0.6264 (7D)
2010-2015	0.6290 (3D)	0.6274 (4D)
2011-2015	0.6392 (4D)	0.6226 (9D)
2012-2015	0.6450 (7D)	0.6313 (8D)
2013-2015	0.6737 (9D)	0.6409 (9D)
2014-2015	**0.7130** (9D)	**0.6535** (4D, 8D)

Table 1: Best training and test accuracy when using the training set with scores averaging.

Notice that the averaging process yields a slight improvement over non-averaging when using the 2014-2015 data as training examples. The accuracy on the test set increases from 0.6515 to 0.6535.

The next thing I implement is a rolling learning approach. I start with the 2014-2015 data as training set and then use this data to learn 1D - 5D models (as opposed to going up to 10D for the interest of time). Next, I use the models to predict the first game. Then, I use the 2014-2015 data plus the first game as a new training set to learn a new set of models and use those models to predict the second game. I repeat this for every game in the 2015-2016 season. The accuracy I obtain on the 2015-2016 test set is 0.7075 which is significantly higher than 0.6535. When implementing this method, I predicted games in order by date. However, from a prediction standpoint, it makes more sense to start with the 2014-2015 data as the training set, learn the models, and use the models to predict however many games there are on the first day of the 2015-2016 season since there is at least one game on each game day. Then, incorporate these games into the training set, learn the new set of models, predict the games on the next day and repeat. This yields a slightly higher accuracy with 0.7095.

3. Probabilistic Matrix Factorization

The probabilistic matrix factorization (PMF) method was designed to address two issues of existing approaches to collaborative filtering on the Netflix Prize dataset that contains 480,189 users, 17,770 movies, and over 100 million observations. The first issue is that existing approaches cannot scale well with large datasets and the second issue is that they cannot make accurate predictions for users with very few ratings. The dataset is large, sparse, and imbalanced in the sense that some users rate less than 5 movies whereas some rate over 10,000 movies. These two issues do not necessarily apply to my problem of NBA game prediction because the dataset I use is not that large and the matrix representing the scores is not sparse and imbalanced since every team plays every other team at least once. Using the basic matrix factorization approach in the previous section gives rise to the same scores for two teams playing against each other since the teams' learned offensive and defensive vectors are fixed. Therefore, the main purpose of using PMF is to incorporate the fact that when two teams play each other, the scores will be different each time.

In the paper describing the PMF model [9], the conditional distribution over observed entries is defined as

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{M}\prod_{j=1}^{N}[N(R_{ij}|U_i^T V_j,\sigma^2)]^{I_{ij}}$$

Where $M = |U|$, $N = |V|$, R_{ij} is user i's rating of movie j, U_i and V_j are user-specific and movie-specific latent feature vectors, respectively, $N(x|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij} is the indicator function that is equal to 1 if user i rated movie j and equal to 0 otherwise.

For modeling NBA games, $M = N$. We can break up the conditional distribution defined above and see that for a game between team i and team j

$$p(R_{ij}|U_i,V_j,\sigma^2) = N(R_{ij}|U_i^T V_j,\sigma^2)$$

$$p(R_{ji}|U_j,V_i,\sigma^2) = N(R_{ji}|U_j^T V_i,\sigma^2)$$

are the conditional distribution of team i's score when playing against team j and team j's score when playing against team i, respectively. U_i is team i's offensive vector and V_j is team j's defensive vector. The total conditional distribution over all observed entries is just the product over all pairs of team matchups. I use the scores averaging method from chapter 2 on the training data to ensure that every matchup between team i and team j corresponds to one observation for team i's score and one observation for team j's score. There are also zero-mean spherical Gaussian priors on offensive and defensive feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{M} N(U_i|0, \sigma_U^2\mathbf{I}), \quad p(V|\sigma_V^2) = \prod_{j=1}^{N} N(V_j|0, \sigma_V^2\mathbf{I})$$

The rows of U and V are independent draws from two K - dimensional Gaussian distributions.

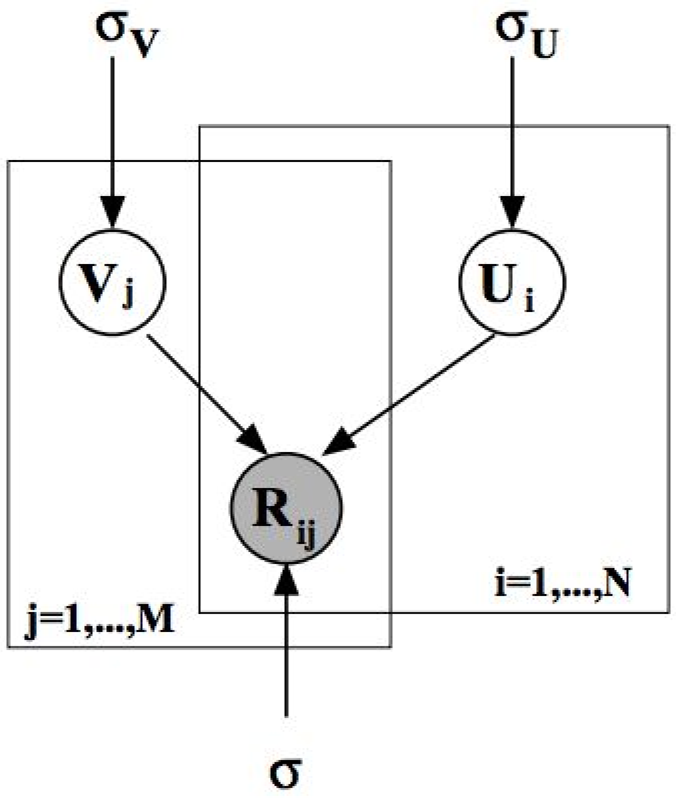


Figure 3: Graphical model for PMF.

The log of the posterior distribution over the offensive and defensive features is given by

$$ln\, p(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(R_{ij} - U_i^T V_j)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{M} U_i^T U_i -$$

$$\frac{1}{2\sigma_V^2}\sum_{j=1}^{N} V_j^T V_j - \frac{1}{2}\left(\left(\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}\right) ln\ \sigma^2 + MD\ ln\ \sigma_U^2 + ND\ ln\ \sigma_V^2\right) + C$$

where C is a constant that does not depend on the parameters. Maximizing the log-posterior over offensive and defensive features with hyperparameters ($\sigma^2, \sigma_U^2, \sigma_V^2$) fixed is equivalent to minimizing the sum-of-squared errors objective function with quadratic regularization terms:

$$E = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{M}\|U_i\|^2 + \frac{\lambda_V}{2}\sum_{j=1}^{N}\|V_j\|^2$$

where $\lambda_U = \frac{\sigma^2}{\sigma_U^2}$, $\lambda_V = \frac{\sigma^2}{\sigma_V^2}$. We can find a local minimum of the objective function by running gradient descent in U and V.

We can choose values for σ^2, σ_U^2, and σ_V^2 by using the soft weight-sharing methods [10], but for simplicity, I use point estimates obtained from the data. Minimizing the objective function gives a local minimum, which is a maximum a posteriori (MAP) estimate. I use the utilities built into PyMC3 [11] to find the MAP estimate with Powell optimization. To make predictions for team i's score when playing against team j using the latent offensive and defensive vectors U and V, I draw from $N(U_i^T V_j, \sigma^2)$ and average over $K = 10$ samples so that

$$P(R_{ij}|\sigma^2,\sigma_U^2,\sigma_V^2) \approx \frac{1}{K}\sum_{k=1}^{K} N(U_i^T V_j, \sigma^2)$$

Using the 2014-2015 season as training data and the point estimates to predict the 2015-2016 season games yields the best accuracy of 0.7102.

4. PMF Incorporating Supplementary Information

When using probabilistic matrix factorization to model data associated with pairwise relationships, knowing supplementary information about the events can help with prediction. For example, for Netflix movie ratings prediction, it is useful to know when the ratings occurred or what actors appear in the movies. However, it is difficult to directly incorporate this supplementary information into the PMF model and doing this for the low-rank matrix factorization model limits the effect to only simple, linear interactions.

In this chapter, I summarize the paper [12] that presents the Dependent Probabilistic Matrix Factorization (DPMF) model, a generalization of probabilistic matrix factorization that replaces scalar latent features with functions, whose inputs are the supplementary information, that vary over the covariate space. The model attempts to connect several related PMF problems and incorporate the supplementary information into the latent features. I adopt the model and techniques presented for my project.

4.1. Summary of DPMF

Let X be the space of supplementary information and $x \in X$ (x is a point in X). We replace latent feature vectors u_m and v_n with latent feature functions $u_m(x) : X \rightarrow \Re^K$ and $v_n(x) : X \rightarrow \Re^K$ to allow for dependence on the supplementary information. If R is the matrix of observations we wish to estimate, we can use a generative model for R that draws R from a distribution parameterized by $\hat{R}$, which is now a function $\hat{R}(x)$ of the supplementary information: $\hat{R}(x) = U(x)V^T(x)$. By instating this dependence on X, we can say that $R(x)$ is drawn from a distribution parameterized by $\hat{R}(x)$. We model each $R(x)$ as conditionally independent given $\hat{R}(x)$. This representation allows the latent features to vary according to x, rather than be constant, which illustrates that the supplementary information should be important to the problem. We construct a nonparametric Bayesian model of the latent features using multi-task Gaussian process (GP) priors for these vector functions.

The Gaussian process is a useful prior because it allows us to use

1. A positive-definite covariance kernel $C(x, x') : X \times X \to \Re$
2. A mean function $\mu(x) : X \to \Re$

to generically specify distributions on functions mapping $X \to \Re$. We use a multi-task Gaussian process for the K scalar component functions rather than use K independent Gaussian processes so that each component within a feature function $u_m(x)$, and analogously for $v_n(x)$, can have a structured prior. However, it is fine for one individual's $u_m(x)$ to be independent of another individual's $u'_m(x)$. We use a matrix L_{Σ_U}, the Cholesky decomposition of an intertask covariance matrix $\Sigma_U = L_{\Sigma_U} L_{\Sigma_U}^T$ to do a point wise linear transformation of the K independent latent functions. The members of M share the covariance functions $C_k^U(x, x')$ and hyperparameters θ_k^U and analogously, the members of N share $C_k^V(x, x')$ and θ_k^V. After the linear transformations, we add constant mean functions $\mu_U(x)$ and $\mu_V(x)$ to each function. The intra-feature sharing of covariance function, mean function, and hyperparameters (i.e. $C_k^U(x, x')$, $\mu_U(x)$, and θ_k^U) indicates that there is consistency among variations of features for members of the set. The inter-feature covariance matrix (i.e. Σ_U) indicates that some features have information (which can be shared) about other features.

Overall, the model relates several PMF problems indexed by X. There are two interesting behaviors in the limit of the length scales of the Gaussian process increasing or decreasing:

1. The length scales of the Gaussian process increase and x becomes uninformative. In the limit, this reduces to one PMF problem.
2. The length scales of the Gaussian process decrease and x becomes very informative. In the limit, each unique x has its own PMF model.

We see then that the length scales of the covariance functions on X indicate how relevant the supplementary information are. The standard choice is to use the automatic relevance determination (ARD) covariance function:

$$C_{ARD}(x, x') = exp\left\{ -\frac{1}{2}\sum_{d=1}^{D}(x_d - x'_d)^2/l_d^2 \right\}$$

where there are $2K$ sets of length scales corresponding to the covariance hyperparameters $\{\theta_k^U\}$ and $\{\theta_k^V\}$. Also, we let $\hat{R}(x) = U(x)\psi(V^T(x))$ where $\psi(x) = ln(1 + e^x)$ is a component wise transformation that restricts entries of V to be positive. This is done to avoid invalid modes in the posterior distribution caused by sign flips in the functions.

Here is a summary of the steps of the DPMF model:

1) Draw K Gaussian process hyperparameters for $\{\theta_k^U\}$ and also K for $\{\theta_k^V\}$ from top-hat priors.

2) Draw $K(M+N)$ functions from the $2K$ Gaussian processes:

   $f^U_{k,m} \sim GP(x_m, \theta^U_k)$ and $f^V_{k,n} \sim GP(x_n, \theta^V_k)$

3) Draw 2 K-dimensional mean vectors μ_U and μ_V from vague Gaussian priors.

4) Draw 2 $K \times K$ cross-covariance matrices Σ_U and Σ_V from uninformative priors on positive definite matrices.

5) Apply the corresponding Cholesky decomposition and add the mean vector.

   $u_m(x) = L_{\Sigma_U} f^U_m + \mu_U$ and $v_n(x) = L_{\Sigma_V} f^V_n + \mu_V$

6) Apply the transformation $\psi(x) = ln(1+e^x)$ elementwise to $\{v_n(x)\}$ to make them strictly positive.

7) $y_{m,n}(x) = u^T_m(x)\psi(v_n(x))$

8) $\hat{R}(x)$ parametrizes a model for the entries of $R(x)$.

   $\hat{R}(x) = U(x)\psi(V^T(x))$ and $R(x) \sim p(R|\hat{R}(x))$

We use Markov Chain Monte Carlo (MCMC) to sample the parameters and latent variables from the posterior distribution on $\hat{R}(x)$ and use the samples to construct a Monte Carlo estimate of the predictive distribution.

The state of the Markov chain is defined with:

1. $U(x)$ and $V(x)$ evaluated at the observations.
2. Hyperparameters $\{\theta^U_k\}$ and $\{\theta^V_k\}$.
3. Feature cross-covariances Σ_U and Σ_V.
4. Feature function means μ_U and μ_V.

5. Any parameters controlling the conditional likelihood of $R(x)$ given $\hat{R}(x)$.

We use elliptical slice sampling (ESS) rather than a transition operator such as Metropolis-Hastings or Hamiltonian Monte Carlo because they require extensive tuning. It is often difficult to sample from the posterior distribution over latent functions with Gaussian process priors so ESS enables efficient slice sampling without needing to tune or use gradients. ESS can make transitions that are never rejected by the Gaussian process prior by taking advantage of invariances in the Gaussian distribution. In depth details on getting slice sampling to work well when sampling the hyperparameters can be found in the paper.

4.2. DPMF for NBA Prediction

I use the DPMF model and techniques presented in the paper [12] to model the scores of games. The matrix $R(x)$ contains the actual scores of the games with supplementary information x.

- $R_{m,n}(x)$: points scored by team m against team n.
- $R_{n,m}(x)$: points scored by team n against team m.

There are two matrix entries (scores) for each observation so I model them with a bivariate Gaussian distribution by placing a joint distribution over them. I use one variance value for all observations, allowing for a correlation between the two team scores. $U(x)$ represents latent feature functions of offense and $V(x)$ represents latent feature functions of defense as before.

I set up a rolling censored-data problem using scores of games in the 2005-2006 to 2015-2016 season to determine the value of using supplementary information. I divide each season into four-week blocks and use the model to make predictions about the games for each block using only past information. For example, the model could only use data from 2005 to November 2006 as training data to make predictions for December 2006. I roll over each block over the entire dataset and retrain the model each time to make predictions.

I use a conditional likelihood function that parametrizes the distribution over the entries in $R(x)$ in terms of $\hat{R}(x)$ as:

$$\left[\begin{array}{c} R_{m,n}(x) \\ R_{n,m}(x) \end{array}\right] \sim N\left(\left[\begin{array}{c} \hat{R}_{m,n}(x) \\ \hat{R}_{n,m}(x) \end{array}\right], \begin{bmatrix} \sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 \end{bmatrix}\right)$$

where $\sigma \in \Re^+$ and $\rho \in (-1, 1)$ parameterize the bivariate Gaussian on scores.

When using temporal information (the time of games) in the models, the goal is to see how the latent features change, possibly due to changes in players or coaches on the teams. However, the idea of time scale is different depending on whether it is the off-season because there are 28 weeks between the end of a regular season and the start of another. To handle this, I use the effective number of weeks between seasons as another parameter. This number is expected to be smaller than the true number of weeks. I use this as a hyperparameter in the covariance functions and include it as part of inference, using the same elliptical slice sampling technique.

Overall, I construct DPMFs using temporal information, home/away information, and both of these together. For temporal information, the original paper only uses the date of the game, but I also include the time as I expect there to be a difference in latent features for a game during daytime versus during nighttime. I apply each of these model using numbers of latent features $K = 1, 2, 3, 4, 5$. I run 15 separate Markov chains to predict each censored interval. For a warm start within a year, I initialize the Markov state from the previous chain's ending state. In every chain, I run the cold start at the beginning of the year for 1200 burnin iterations and run the warm start for 150 iterations. I keep 100 samples of each predictive score from each chain after burning in and thinning by a factor of 4.

I only provide data from the current season and previous two seasons to the DPMF model to prevent it from being highly affected by old data. Also, it is costly to sample the covariance hyperparameters because it requires computing multiple Cholesky decompositions. To make it more efficient, I run an extensive Markov chain to burn in these hyperparameters and use them for all further sampling. Results from the evaluations are in the table below.

	K	2005	2006	2007	2008	2009	2010	2011	2012	2013	201	2015	2016	All
(T)	1	37.1	39.0	34.3	34.2	35.5	36.1	34.4	35.0	38.2	35.2	33.2	32.4	38.7
	2	38.8	39.9	34.0	33.5	37.9	34.3	35.0	34.2	36.3	34.8	31.7	30.6	37.9
	3	35.5	37.8	36.4	32.9	32.3	32.8	33.9	32.9	37.1	35.8	32.9	31.2	38.1
	4	34.9	38.8	35.5	35.1	35.5	33.1	34.1	33.6	35.4	35.8	30.6	33.9	37.7
	5	35.1	38.1	35.9	34.0	34.8	33.5	33.2	34.8	35.0	34.5	33.0	32.2	39.0
(H)	1	34.2	38.3	37.5	33.3	30.1	37.0	36.0	34.1	33.2	33.3	34.1	31.7	35.5
	2	35.6	37.7	37.7	33.9	31.1	35.6	35.6	35.7	34.1	31.7	32.2	33.2	36.8
	3	34.7	36.2	39.0	34.6	30.4	34.3	35.9	32.1	35.3	35.0	35.5	30.3	37.0
	4	35.9	38.0	37.6	34.7	32.1	36.1	36.1	33.8	32.9	31.2	34.7	29.9	36.2
	5	34.4	37.6	38.0	33.8	30.0	33.7	34.4	34.0	33.0	32.7	31.4	30.8	37.1
(TH)	1	34.8	38.3	33.5	31.9	37.6	32.5	32.2	33.3	33.5	33.1	30.2	29.7	34.3
	2	34.7	38.6	33.3	32.2	35.5	**31.9**	33.5	34.9	32.2	31.0	30.2	31.1	34.1
	3	**32.9**	36.8	34.4	31.6	**29.9**	32.8	**31.2**	33.1	32.6	30.3	29.2	28.7	33.9
	4	35.0	36.6	**32.8**	**31.5**	30.4	33.2	34.0	**32.0**	33.8	**29.0**	28.9	29.0	34.3
	5	33.1	**36.0**	33.0	31.6	31.8	33.9	33.9	32.7	**31.0**	29.3	**28.2**	**27.9**	**33.6**

Table 2: Percentage errors of winner predictions using DPMFs with different number of latent features. (T) uses only temporal (date and time of game) information, (H) uses only home/away information, and (TH) uses both.

We can see that for the 2015-2016 season, the DPMF with temporal and home/away information with $K = 5$ yields an error of 27.9% or an accuracy of 72.1%, which is great improvement over 71.02% with the standard PMF model.

5. Conclusion

In my thesis, I introduce basketball prediction as a natural application of machine learning. I present the different available data: box-score data, play-by-play data, and camera-tracked data. Box-score data and play-by-play data are publicly available and have been around for a while, but camera-tracked data has only been around since 2010 and it is not publicly available. Next, I discuss about most of the previous work in basketball analytics, which can be categorized as descriptive or predictive analytics. Descriptive analytics focuses on describing what has already happened, for example, evaluating players or teams, but my project is about predictive analytics. Most predictive models attempt to predict the outcome of one game. They mainly use features from box-score data and simply apply off the shelf algorithms such as logistic regression, SVM, neural networks, or Naive Bayes or engineer new features from box-score data and play-by-play data. Since many data are best modeled as the result of pairwise interactions and an NBA game is an interaction between two teams, I discuss how matrix factorization is a more interesting and suitable approach to the problem of predicting game outcomes. My goal is to predict game outcomes for this regular season (2015-2016). First, I use the basic low-rank matrix factorization approach and achieve an accuracy of 70.95%. Next, I use probabilistic matrix factorization (PMF) to incorporate the fact that when two teams play each other, the scores will be different each time. This yields an accuracy of 71.42%. Last, I incorporate supplementary information, the date and time of the game and information about home/away team into the PMF model. This model, named the Dependent Probabilistic Matrix Factorization (DPMF) combines multiply PMF problems using Gaussian process priors. This involves replacing the scalar latent features with functions of this supplementary information. Overall, this yields the best accuracy of 72.1%.

5.1. Future Work

For the basic low-rank matrix factorization approach, I use the scores averaging method so that entry (i, j) holds team i's average number of points that team i scores against team j throughout the games they play in the training set used and vice versa for entry (j, i). I do this so that although two teams can play each other more than once, entries (i, j) and (j, i) in the observed matrix each has only one value. Perhaps, it is more useful to use an exponential running average, which emphasizes more recent scores since those scores are more indicative of how the team's current strengths and weaknesses are.

In the probabilistic matrix factorization paper [9], the authors present an alternative version called the Constrained PMF. When applying to the Netflix Prize problem, the basic idea is that users who have seen the same or similar movies will have similar prior distributions for their feature vectors. Perhaps, I can apply something similar to NBA teams. Certain NBA teams with similar team dynamics or with coaches with similar styles will probably have similar prior distributions for their offensive and/or defensive feature vectors.

In the DPMF model, the obvious possible future work is to use other supplementary information, such as information about coaches, besides information about the date and time of the game and the home/away information. A suggested future work in the paper is that the

Gaussian process allows for only smooth variation in latent features. However, critical events like players being traded or getting injured are reflected better with a changepoint model.

6. References

[1] Jon Bois. *13 points, 33 seconds: The night Tracy McGrady was a basketball god*. http://www.sbnation.com/2013/8/27/4661546/tracy-mcgrady-13-points-33-seconds, 2013.

[2] Thomas Duffy. *We Remember: Derek Fisher Sinks Game-Winner vs. Spurs with 0.4 Seconds Left.* http://bleacherreport.com/articles/2463874-we-remember-derek-fisher-sinks-game-winner-vs-spurs-with-04-seconds-left, 2015.

[3] NBA. http://stats.nba.com/scores, 2016.

[4] Basketball-reference. http://www.basketball-reference.com/, 2016.

[5] Zach Lowe. *Lights, Cameras. Revolution*. http://grantland.com/features/the-toronto-raptors-sportvu-cameras-nba-analytical-revolution/, 2013.

[6] SportVU. http://www.stats.com/sportvu/sportvu-basketball-media/, 2016.

[7] Matthew Beckler and Hongfei Wang. *NBA Oracle*. http://www.mbeckler.org/coursework/2008-2009/10701_report.pdf

[8] Danny Tarlow. *Data driven march madness predictions*. http://blog.smellthedata.com/2009/03/data-driven-march-madness-predictions.html

[9] Ruslan Salakhutdinov and Andriy Mnih. *Probabilistic Matrix Factorization*. https://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf

[10] Steven J. Nowlan and Geoffrey E. Hinton. *Simplifying Neural Networks by Soft Weight-Sharing*. http://www.cs.toronto.edu/~fritz/absps/sunspots.pdf

[11] PyMC3. *Probabilistic Matrix Factorization for Making Personalized Recommendations*. https://pymc-devs.github.io/pymc3/pmf-pymc/#probabilistic-matrix-factorization

[12] Ryan Prescott Adams, George E. Dahl, and Iain Murray. *Incorporating Side Information in Probabilistic Matrix Factorization with Gaussian Processes*. http://www.cs.toronto.edu/~gdahl/papers/dpmfNBA.pdf