Applying Absorbing Markov Chains to simulate Tennis Matches

Siddharth Sharma

MATH 104, December 2022

1 Introduction

Tennis is a sport played by millions worldwide. With its unique scoring system and the alternating nature of point play, it is a discrete problem which can be modeled. In tennis, scoring follows a distinct pattern that allows for the easy tracking of the progress of a match. A game is won by the first player to reach four points, with the winner being the first to win two out of three games in a set. A match is typically comprised of multiple sets, with the winner being the first to win the majority of sets (typically two out of three or three out of five). Points in tennis are awarded based on the outcome of a rally, with the serving player receiving a point if the receiving player is unable to return the ball within the bounds of the court, or if the receiving player commits a fault (such as stepping on the boundary line while hitting the ball). There is an edge case regarding the case of deuce. In tennis, "deuce" is the term used to describe a situation in which both players have scored three points each in a game. This means that the score is tied at 40-40 and the next point will determine the winner of the game. In order to win the game, a player must win two consecutive points after the score is deuce. This is known as "advantage" for the winning player, as they only need one more point to win the game. If the player who wins the first point after deuce loses the next point, the score returns to deuce and the process is repeated until one player wins two consecutive points and thus wins the game.

Given the unique scoring system and opportunity to model points, this work aims to simulate the winner of a tennis match by simplifying the problem into a nested absorbing Markov chain.

2 Absorbing Markov Chains

In the field of mathematics, an absorbing Markov matrix is a type of square matrix that is used to model the transitions between different states in a stochastic system. An absorbing matrix is a square matrix that has the property that, once a state is entered, it is impossible to leave that state. A Markov matrix,

on the other hand, is a square matrix that satisfies the Markov property, which states that the probabilities of transitioning between different states in the system are constant over time.

An absorbing Markov matrix is a type of matrix that combines these two properties. It is a square matrix that satisfies the Markov property and also has one or more "absorbing" states, which are states that, once entered, cannot be left. This type of matrix is used to model systems in which certain states are considered "absorbing" or "terminal" states, such as in the modeling of Markov chains or Markov decision processes. The use of Markov absorbing matrices allows for the modeling of systems in which the transitions between different states are probabilistic and the outcome of the system is determined by the state that is ultimately reached. This type of matrix is commonly used in the study of systems involving random processes, such as in the fields of economics and engineering. An example of an absorbing Markov chain is given below:

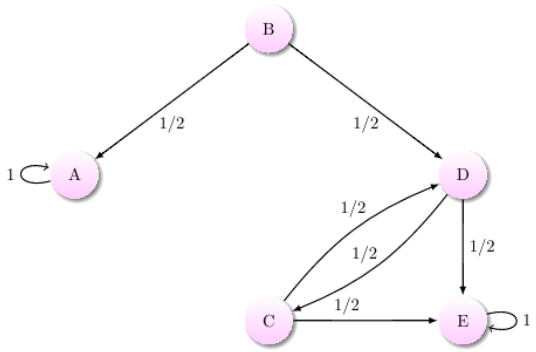

Figure 1: Example of absorbing Markov chain

In the example above, the `A` and `E` states are the absorbing states since when they are reached, they cannot be exited. Formally, a Markov chain is an absorbing chain if the following properties hold: (1) there is at least one absorbing sate, (2) it is possible to go from any state to an absorbing state in a finite number of steps. We can also show a Markov absorbing chain in its canonical form: let the absorbing chain have a transition matrix P with t transient states and r absorbing states. Let the rows represent the starting state and the column represent the next state. Moreover, let the r absorbing states be the last r states of P. Then, we can model M as follows:

$$P = \begin{bmatrix} Q & R \\ 0 & I_r \end{bmatrix} \tag{1}$$

In the canonical form expressed above, the matrix $R \in \mathbb{R}^{r \times r}$ is the probability of transitioning from a transient state to an absorbing state while the matrix $Q \in \mathbb{R}^{t \times t}$ is the probability of transitioning from a transient state to another

transient state. In this case, the identity matrix I_r represents the idea that once an absorbing state is reached, it cannot be changed or terminated. We will now apply the absorbing Markov chains to model tennis.

3 Simulating a Tennis Match

To simulate a tennis match, it will be necessary to apply multiple absorbing Markov chains within each other, forming a nested absorbing Markov chain. This is because a point (or rally) can be modeled with an absorbing Markov chain and similarly, a game is an absorbing Markov chain dependent on simulating points to transition between states. A set is then a set of trials modeling player's serving. This approach works since tennis matches are not time-dependent: over several shots, points, and games, discrete outcomes are reached. Please note that tennis is a complex sport with several hundred factors ranging from playing surface to player rankings to conditions; thus, this is only a *model* of predicted scores given a very set of statistics. A fundamental characteristic of our model is that we only model points and game from the server's perspective. This simplifies several aspects of the match.

3.1 Simulating a Point

Let us first formulate the absorbing Markov matrix for a point. In tennis literature, it is known that first serve percentage and unenforced error rates are among the most crucial set of statistics in determining if a player won a point. Moreover, a point in tennis is essentially a serve (either first, second, or double fault) followed by an exchange of shots. For simplicity, we will disregard drop shots, volleys, lobs, and other unique tactical shots. Thus, in this model, the shots following a serve are constrained to forehands and backhands. Note that a point in tennis ends with either a winner or an error (we do not separate forced errors from unforced errors for simplicity). Thus, the absorbing states for modeling a single point are the states of a winner and error. Let's formulate the transition matrix P_{point} with the following states (in order): {1st serve, 2nd serve, double fault, forehand, backhand, winner, error}. Now assume we are given the following probabilities: p_{fs}, probability of making a first serve; p_{ss}, probability of making a second serve; p_{df}, the probability of the serve being a double fault; p_{fh}, the probability of making a forehand to continue the rally; p_{bh}, the probability of hitting a backhand to continue the rally; p_w, the probability of hitting a winner (ending the point); p_e, the probability of hitting an error (ending the point). With the given ordering of states and aforementioned probabilities, we can now generate the matrix P_{point} that models the transition between states (shots) for a point in tennis:

$$\textbf{States} = \{\text{1st serve, 2nd serve, double fault, forehand, backhand, winner, error}\}. \tag{2}$$

$$P_{\text{point}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_{fh} & p_{fh} & 0 & p_{fh} & p_{fh} & 0 & 0 \\ p_{bh} & p_{bh} & 0 & p_{bh} & p_{bh} & 0 & 0 \\ p_w & p_w & 0 & p_w & p_w & \mathbf{1} & 0 \\ p_e & p_e & \mathbf{1} & p_e & p_e & 0 & \mathbf{1} \end{bmatrix} \tag{3}$$

In our above transition matrix, the absorbing states are bolded: they occur when there is a double fault, a winner, and an error. Note that we also assume that $p_{fh}+p_{bh}+p_w+p_e = 1$: in other words, the probabilities of a rally forehand, rally backhand, winner, and error sum to 1. To more accurately represent the structure of a tennis point, we can incorporate the offensive and defensive nature of shots. To accomplish this, I apply a probability boost in regard to first serves and forehands since they can considered more offensive shots (greater probability of winners). Following this, to maintain the probabilities adding to 1, we then offset the probabilities of second serves and backhands. Similarly, in defensive situations (following a backhand or second serve), there is no boost and the probability of an error is increased. Let the boost given to forehands and first serves be denoted as fh_{boost} and $fserve_{\text{boost}}$ respectively. Thus, our finalized transition matrix for simulating a point is:

$$P^{'}_{\text{point}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_{fh} + fserve_{\text{boost}} & p_{fh} & 0 & p_{fh} & p_{fh} & 0 & 0 \\ p_{bh} - fserve_{\text{boost}} & p_{bh} & 0 & p_{bh} & p_{bh} & 0 & 0 \\ p_w + fserve_{\text{boost}} & p_w & 0 & p_w + fh_{\text{boost}} & p_w - fh_{\text{boost}} & \mathbf{1} & 0 \\ p_e - fserve_{\text{boost}} & p_e & \mathbf{1} & p_e - fh_{\text{boost}} & p_e + fh_{\text{boost}} & 0 & \mathbf{1} \end{bmatrix} \tag{4}$$

The initial state for all points is $S_o = [p_{\text{fs}}, p_{\text{ss}}, p_{\text{df}}, 0, 0, 0, 0]$ since the first shot is some kind of service outcome: either a first serve is made, a second serve is made, or a double fault occurred. To add an element of randomness to shots, we also specify a temperature parameter that adjusts the various probabilities of the $P^{'}_{\text{point}}$ matrix on a point-by-point basis. To simulate a point with software, we can find the stable state matrix for very large but computationally reasonable n where n is the number of shots played:

$$v_{\text{point}} = \left(P^{'}_{\text{point}}\right)^n S_o \tag{5}$$

3.2 Simulating a Game

To simulate a game, we must create a transition matrix that transitions between scores based on the probability of winning a unique point each time. In the context of absorbing Markov chains, the absorbing states here are "holding serve" (win game) and "breaking serve" (lose game). Again, we are modeling the game simulation from the server's perspective. There are 17 states for scores (15 transient, 2 absorbing). They are denoted in order as follows: 0-0, 15-0, 0-15, 30-0, 0-30, 30-15, 15-30, 40-15, 15-40, D (deuce), AD-IN, AD-OUT, 40-0, 0-40, hold (win), break (lose). We must first note that the order of points in tennis scoring is critical. For an arbitrary score in the format $X - Y$, the server's points are denoted via X, the left-hand side, while the returner's points are denoted via Y, the right-hand side. Another important assumption of the above states is that the 30-30 score is the same concept as deuce since a player must win two points in a row to progress to an outcome; otherwise, this transient state will be repeated.

Now let the probability of winning a point be p_w and losing the same point be p_l as generated by the stable state of the absorbing Markov transition matrix of points. We can now present the the transition matrix to model a game, P_{game}, as follows:

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
p_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
p_w & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & p_w & p_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & p_w & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & p_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & p_w & p_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & p_l & 0 & p_w & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & p_w & 0 & 0 & 0 & 0 & 0 & 0 & p_l & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & p_l & 0 & 0 & 0 & 0 & 0 & 0 & p_w & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & p_l & p_w & 0 & 0 & 0 & p_l & p_w & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & p_l & 0 & p_w & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_w & p_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & p_w & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & p_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_w & 0 & 0 & p_w & 0 & p_w & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_l & 0 & 0 & p_l & 0 & p_l & 0 & \mathbf{1}
\end{bmatrix} \tag{6}$$

As seen in the last two columns, the absorbing states are hold (win game) and break (lose game). The initial state T_o for all games is [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] since all games start with the state 0-0. To find the probability of a server winning their game, we can compute the stable state matrix for very large but computationally reasonable m where m is the number

of points in the game:

$$v_{\text{game}} = (P'_{\text{game}})^{m} T_o \tag{7}$$

Note that the p_w and p_l will be different each time the transition matrix is applied to the existing state since each point has an associated randomness as determined by the temperature parameter.

4 Experiments

To simulate a set and more specifically a match, I performed experiments using a Google Colab environment. A set can be simulated by having both players attempt to hold serve in a fixed order as determined by a coin toss until a player reaches a 6 games. For simplicity, tiebreaks were not accounted for. To then simulate multiple sets (a whole match), one could predict each set at a time until a player wins best 2 out of 3 sets or best 3 out of 5 sets depending on the match format.

To find statistics for the point transition probabilities, I used data that roughly matched Roger Federer's career statistics. For example, Federer has a career average first service percentage of 65% and a double fault percentage of 2.5%. I then used the `NumPy` library to store my transition matrices and initial states. Following this, I computed the stable state probabilities per point and per-game for an $n = 10,000$ and $m = 10,000$ respectively. I then tied it all together in a wrapper function that shows the score in realtime as points are predicted. An example of the interface is provided below:

```
Based on the result of the coin toss,  Gene Kim  is to serve first
Beginning Match ...
Gene Kim  to serve:
Gene Kim  holds serve!
Sid  to serve:
Sid  holds serve!
---------------------------------
Sid :  1
Gene Kim :  1
---------------------------------


Gene Kim  to serve:
Gene Kim  holds serve!
Sid  to serve:
Sid  holds serve!
---------------------------------
Sid :  2
Gene Kim :  2
---------------------------------


Gene Kim  to serve:
Gene Kim  holds serve!
Sid  to serve:
Sid  holds serve!
---------------------------------
Sid :  3
Gene Kim :  3
---------------------------------
```

Figure 2: Match simulation (only first three service holds for each player)

Overall, the results are favorable given holds of serve are more common to breaks of serve while one or two breaks of serve still typically occur in a set. Moreover, as the probabilities of forehands, backhands, winners, and errors change (across different players) alongside probabilities of first service versus second service and double fault, the results make sense. Interestingly, we can glean insights into ideas like optimal first service percentage, whether players should hit forehands or backhands more.

5 Future Work

In the future, it may be relevant to consider more tactical shots (drop shots, lobs, volleys) as well as overall strategies (cross-court rallies, down the line shots, change of pace, approaching the net). Adding a probability offset in regard to surface may also be interesting since players like Rafa Nadal highly benefit from slower clay courts to control points while big servers like John Isner benefit from fast grass courts to hit aces and winners. It would also be of interest to customize skillsets by player. For example, Novak Djokovic and Stan Wawrinka are known to be more offensive on their backhands so the point absorbing markov chain would need to take this into account. Moreover, it may also be worthwhile to add a strategic temperature parameter per game that affects how the point is played. This is because a Deuce (40-40) point is typically played in a different style (more pressure and less offense) than a 40-0 or 40-15 point. Another idea is that as a match goes on, a fatigue parameter could be introduced to enable higher error rates. Lastly, in the future, it may be worthwhile to potentially add in another nested absorbing Markov chain that accounts for which region of the court the shot was hit into.

Overall, the applications of this work are exciting and with more development, a more complex representation that accounts for several player stats and conditions may be possible. Furthermore, absorbing Markov chains enable us to even simulate from an existing point in a match and this has unique properties.

6 Acknowledgements

I'm a huge fan of tennis, having played for over 14 years and reaching the top 200 juniors in the Under-18 division for California when I graduated high school. At Stanford, I also serve as a manager for our NCAA/Varsity Men's Tennis Team. Given I'm a computer science major, I've made multiple attempts at simulating tennis matches digitally before this project and this experience in particular has only helped me to further appreciate the nuances in mathematical prediction. I'd like to thank Prof. Gene B. Kim and Stanford for making a lasting impression on my treatment of linear algebra. I'd also like to thank my family, friends, and of course, my idol, Roger Federer (the GOAT). If you'd like to learn more about my passion for tennis and admiration for Roger Federer, this article I recently

wrote captures the essence pretty well.

7 References

will add