Modelling Optimal Tennis Decisions with Reinforcement Learning

Stylianos Rousoglou
Computer Science MSCS
Stanford Unviersity
steliosr@stanford.edu

Abstract

Modelling sports can be especially challenging given the large variety and high complexity of factors that human professional athletes take into account simultaneously when making decisions during gameplay. In this paper, we first develop a parametrized simulation of the game of Tennis, using all modern rules of the game. Subsequently, we employ Reinforcement Learning to train sophisticated agents and evaluate their success in learning obvious as well as empirical rules about how to optimize their play, given their different capabilities. Using a null action-selection model as a baseline, we compare the performance of agents trained with Q-Learning to see how they perform against equally skilled players, as well as how they might adapt to exploit their personal strengths and mitigate their weaknesses.

I. Introduction

Tennis is a sport that is as proactive as it is reactive. Positioning oneself correctly in the court after hitting the ball is essential in gaining an advantage in subsequent ball hits. Of course, the choice of a player's positioning depends on how the opponent returns the ball; however, in modern tennis, the ball travels at speeds of 100 mph, so waiting until the opponent has hit the ball to position oneself is a recipe for defeat. Instead, players depend on their *best estimate* of where the ball is likely to be returned, and make quick positioning choices based on their experience, skill, and likelihood of outcome. We have increasingly seen sports analytics utilized in training athletes for individual sports (e.g. track & field) as well as teams for sports like basketball and baseball; one might postulate that tennis players also stand to gain from a quantitative analysis of their game.

Although in theory we can observe specific patterns of play employed by specific players, in practice there are also external variables to be accounted for in such patterns, such as the skills and gameplay style of the opponent. Professional players developing their own "style of play" refers to the adoption (or rejection) of specific strategies that have proven to work well for them, as well as the adaptation to specific opponents' gameplays at times of competition. Therefore, we can model a player's decisions on the court at a high-level as a function of their skillset, their trained intuition, and their receptiveness to the style of play of respective opponents.

To make these theoretical claims concrete, we first need to develop a Tennis game simulation, and then pitch different players against each other under varying conditions. After making simplifying assumptions, we develop a semi-stochastic simulation of a game between two players, parametrized for various abilities of the two players (in this paper *accuracy, speed,* and *serve.)* To model the process of player training and experience, we need a learning algorithm that is flexible enough to learn both intuitive and non-obvious action decisions without requiring too accurate of a state-space definition (because that, after all, would be near impossible in a highly-complex sport.) We will then run simulations with players utilizing different policies, to verify that repetition and reinforcement can actually lead to more nuanced policies being learned and employed to gain a competitive advantage. We will also look closely at some sample actions of such learned policies and try to decompose them in order to understand the extent to which the developed models have learned to perform optimal state-action decisions.

II. Related work

There have been several attempts to use Machine Learning Methods and data, historical, demographic, and otherwise, to predict the outcome of tennis matches. However, they largely depend on maximum likelihood probability distributions drawn from players' past game data and aim to predict individual match outcomes for the purposes of sports betting; therefore, none of the data used in such work was relevant to my approach, which models and examines matches on a microscopic level (individual point.)

In "Probability of Winning at Tennis" [2], Keller and Newton model individual points as independently identically distributed random variables and use match-level data to simulate tennis tournaments. In "Optimal Strategy in Tennis: A Simple Probabilistic Model" [3], George uses data analysis to describe an optimal service strategy. More recently, Ferrante, Fonseca, and Pontarollo [4] develop the law of "random duration of a game" in a more general setting than the Markovian

Figure 1: Court quadrants for both players

model previously assumed.

Q-Learning is a model-free Reinforcement Learning algorithm. Reinforcement Learning is widely used in contexts where agents need to develop a state-action policy to maximize some long-term reward accumulation. Being model-free, Q-Learning does not require a precisely defined state transition model; more conveniently, it utilizes state, action, and reward samples drawn directly from repeated gameplay in order to learn the policy that accumulates maximal reward; the state-action policy is learned by applying incremental estimation to the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

where s, a, t are the state, action, and reward observed in a single example, $\alpha \in [0,1]$ is the learning rate, and $\gamma \in [0,1]$ is a discount factor.

III. Approach

A. Rules of the Game

Tennis is a 2-player sport in which each player occupies one side of the court (symmetry around vertical axis in the middle, which is "the Net" – see Figure 1). The smallest fundamental unit of play is a "point", which is defined as a continuous exchange of the ball between the two players, where the players *must* hit the ball after it has bounced *at most* once. Each player must aim to hit the ball within the opponent's court side. If player k's hit misses the opponent's court (ball hitting the net or out of bounds), or if a player k allows the opponent's ball to bounce more than once (the first time being within their own court), player k loses the point. Each player's side of the court is thought of as being split in 4 quadrants, which we will refer to by their topological abbreviations in *Figure 1* (Bottom Right (BR), Top Left (TL), etc.)

The first player to win 4 points is awarded one "game".[1] The first player to win 6 games by a margin of at least 2, is awarded the "set". In case of a 6-6 game tie, a "tie-break" is played for the set to be awarded to the player who wins at least 7 points with a margin of at least 2 points. The first player to win 2 "sets", wins the match. Note that in alternating games, the players alternate "serve" (first hit of each "point"); players have 2 chances to serve within bounds (on each point) before they lose the point.

B. Tennis Match Simulation

In order to develop a reasonably simple tennis match simulation, simplifying assumptions about the conditions and outcomes of play had be made. There was also a need to encode expertise into the model; since real-life tennis gameplay data are not available, this was done in terms of approximate probability distributions of stochastic outcomes of different actions. There prior distributions were estimated with top tennis players in mind and would not represent the ability or stochasticity of outcome of an average tennis player. For simplicity, the different types of ball hits are classified into two distinct categories: Top Spin (*TS)* and Slice (*SL) (See Figure 2)*

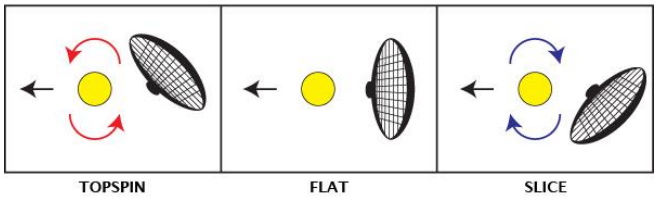

Figure 2: Types of ball hits

The capabilities of the 2 players are parametrized by a vector of the form $[a_1, b_1, a_2, b_2, a_3, b_3]$.
a_i are the parameters corresponding to Player 1, while b_i correspond to Player 2.

- $a_1, b_1 \in \{1,2,3,4,5,6,7,8,9,10\}$ parametrize the players' placement (accuracy) skills
- $a_2, b_2 \in \{1,2,3,4,5,6,7,8,9,10\}$ parametrize the players' velocity
- $a_3, b_3 \in \{1,2,3,4,5,6,7,8,9,10\}$ parametrize the players' serving capability

The simplifying assumptions made in implementing the game's rules are the following:

- There are 4 discrete positions on each side of the court: $\{BR, BL, TR, TL\}$.
- There are 2 types of ball hits: $\{TS, SL\}$.
- There are 4 available actions, corresponding to quarter quadrants: $\{BR, BL, TR, TL\}$; they represent a player's attempt to run from their current position to the respective quadrant.

Using the parametrization described above, the game simulation was implemented stochastically as follows:

- $P_{RETURN}(pos,\ spin,\ ret)$ represents the baseline probability that a player will return a ball successfully from quadrant ret (current position) when the incoming ball bounces in quadrant pos with spin $spin$.

- $P_{OUTCOMES}(pos,\ ret)$ represents the baseline probability that a player aims the ball successfully at opponent's ret when hitting from their current position in quadrant pos

- $P_{ACTIONS}(pos,\ a)$ represents the baseline probability that a player in quadrant pos will successfully move to quadrant a in time to hit the incoming ball. If $pos = a$, then the outcomes is deterministic (100% success);

1. No Ad scoring

otherwise, the outcome is stochastic (e.g. successfully running from BL to BR is more likely than from BL to TR)

- $P_{SERVE}(1\ or\ 2)$ represents the baseline probability that a player successfully serves (1st serve is TS, 2nd serve is SL)

The players' parametrization is then factored in as follows:

- The probability that player 1 returns a ball successfully from quadrant ret (current position) when the incoming ball bounces in quadrant pos with spin $spin$ is
$$P_{RETURN}(pos, spin, ret) \times 1.015^{a_1}$$
- The probability that player 1 in quadrant pos successfully moves to quadrant a in time to hit the incoming ball is
$$P_{ACTIONS}(pos, a) \times 1.04^{a_2}$$
This implies a stochasticity of outcome; with probability $1 - P_{ACTIONS}(pos, a) \times 1.04^{a_2}$ the player ends up in one of the other quadrants (probabilities depend on current position pos and attempted action a.)

- The probability that player 1 serves successfully on his n th serve, $n \in \{1, 2\}$ is $P_{SERVE}(n) \times 1.015^{a_1}$

The scaling factors for the baseline probabilities were determined empirically. Each tennis match simulation runs in ~ 0.01 seconds.

C. Null Agent

In order to evaluate the performance of more advanced agents, a NULL agent state-action function has to be developed and act as a performance *baseline*. Just like other players, the NULL agent is parametrized by 3 values $\{n_1, n_2, n_3\}$. The state-action function then chooses the action a that corresponds to the quadrant in which *the incoming ball is expected to bounce.* Note that this NULL agent is *not* choosing actions randomly; an agent choosing quadrants to move to at random would perform *worse* than out NULL agent, because there is significant correlation between a player's *proximity* to the ball's quadrant and *successfully returning.*

However, oftentimes it's not the case that a player wants to be positioned in the quadrant in which the ball is bouncing. This complicated decision process is, after all, a factor that distinguishes average from top tennis players. As a result, we expect these nuanced action choices to be better represented by the state-action function learned by our next agent, where Q-learning is utilized.

D. Q-Learning Agent

The Q-Learning agent is trained with varying numbers of iterations. In each iteration, a single point is played between the parametrized players, and 2 state-action functions are learned simultaneously, one corresponding to each player. As a result, we expect players to not only learn how to act optimally to exploit their strengths, but to learn state-action functions that also exploit the opponents' weaknesses. Agents receive no rewards for each call hit that results to continuing game play; the player who wins the point receives a reward of $+3$; the player who loses the point receives a reward of -3.

No ad-hoc exploration strategy is used for Q-learning in the context of the described tennis simulation. Each point starts with one of the players serving (uniformly at random). After the serve, the movement of the ball and the success of each player in returning it is described by the stochastic model outlined in the previous sections in conjunction with a random action selector; we postulate that since the action space is very small, a large number of iterations will be enough to explore it substantially.

Since all points are generally considered equally important in tennis, there we use a discount factor of $\gamma = 1$ (no discounting).

E. Eligibility Traces

Since the rewards from the point simulations above are sparse (assuming an average of 10 ball hits per point, only 1/5 of all state-action pairs observed actually result in an immediate reward), eligibility traces can be employed to award a discounted fraction of the reward of the ultimate state-action outcome to the state-action pairs that preceded it. Since winning a point in tennis is a matter of sequential decision making in which good action choices can offer advantages (or disadvantages) in subsequent ball hits, we expect that eligibility traces may improve the performance of our Q-learning agent.

IV. Experiments & Results

A. Null agent vs. Q-learning agent

We begin by evaluating the performance of Player 1 against Player 2 (without loss of generality*) assuming equal skillsets,* as a fraction of wins out of 1,000 matches played in two scenarios labelled as follows:

- NULL: Player 1 and Player 2 choose actions based on the Null model (in Blue)
- RL: Player 1 is a Q-learning agent (utilizing a learned Q state-action function); Player 2 is a Null model agent (in Orange)

We vary the number of iterations of the Q-learning algorithm, expecting that more Reinforcement Learning iterations will result in higher relative performance for the Q-learning agent. The results are presented in *Figure*

3, in which the Q-learning agent is trained using 1,000, 10,000, and 100,000 iterations, respectively. The parameters vector in all 3 graphs parametrizes both players identically; specifically, $[8, 8, 8, 8, 8, 8]$ is used.

As we expected, Null agents pitched against themselves win ~ 50% of the matches (they are equally good) in all three scenarios. However, the margin by which the Q-learning agent dominates the Null agent increases as the iterations in the Q-learning algorithm are increased. Specifically, the percentage of wins for the Q-learning agents seems to converge at ~ 65% with 1,000 iterations (Figure 3a); at ~ 77% with 10,000 iterations (Figure 3b); and at ~ 93% with 100,000 iterations (Figure 3c.)

Meanwhile, utilizing eligibility traces with a discount factor of $\gamma = 0.9$ seems to improve results only under specific conditions. The Q-learning agent *does* perform significantly better when eligibility traces are used at 10,000 iterations (See Figure 4); it didn't seem to do any better than the players trained without eligibility traces at 1,000 and 100,000 iterations. This can be attributed to the following:

- If the number of iterations is significantly low, then Q-learning will not have encountered enough examples to converge to the optimal state-action policy, even if eligibility traces are used to compute more Q(s, a) values.
- If the number of iterations is significantly high, then despite the sparsity of rewards, Q-learning will have encountered enough examples to converge to the optimal state-action policy, irrespective of the use of eligibility traces.
- If the number of iterations is high, but hasn't led to convergence of the optimal Q(s,a) policy, then eligibility traces could help bring the policy closer to convergence.

As a result, we see the Q-leaning agent in *Figure 4* outperform its equally skilled counterpart in *Figure 3b;* but the Q-leaning agents in *Figure 3a* and *Figure 3c,* and the respective agents utilizing eligibility traces (not shown), seem to converge to about the same fraction.

The Null vs. Q-learning agent results are summarized in the *Table 1*.

Q Iterations	**Fraction of wins (P1 in 1,000 matches)**		
	NULL	***RL***	***RL + eligibility traces***
1,000	~ 50%	**65%**	62%
10,000	~ 50%	76%	**83%**
100,000	~ 50%	**93%**	**93%**

Table 1: Fraction of Player 1 wins (1,000 matches)

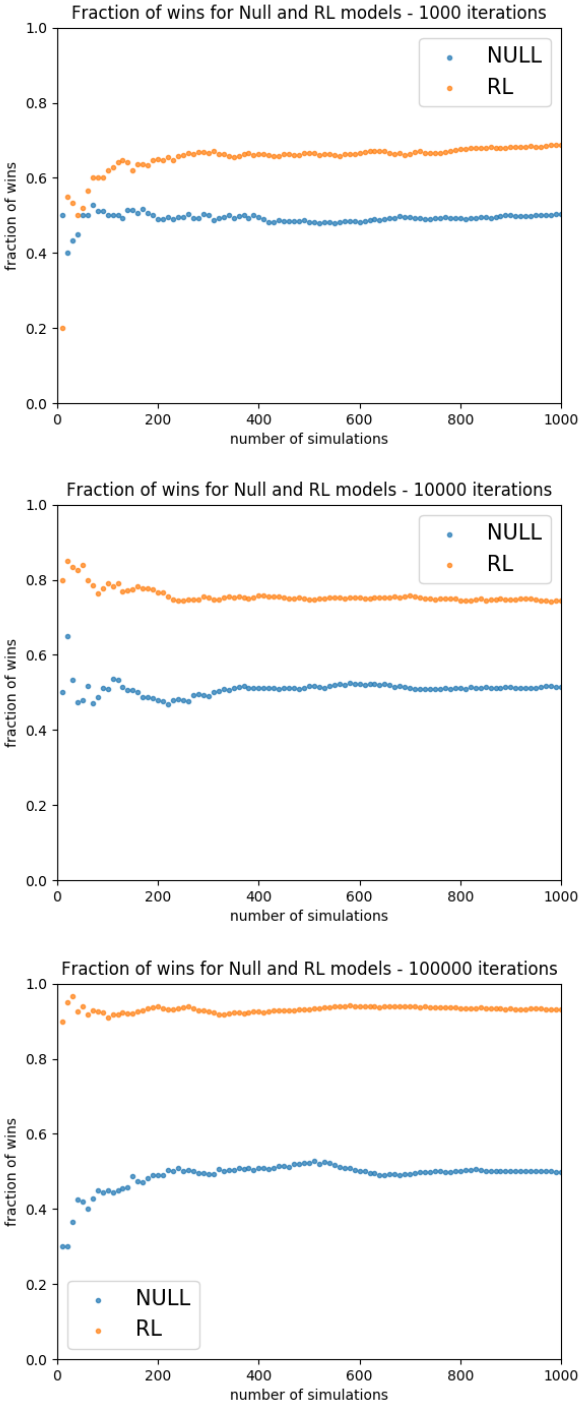

Figure 3: RL vs. Null agent for increasing Q iterations

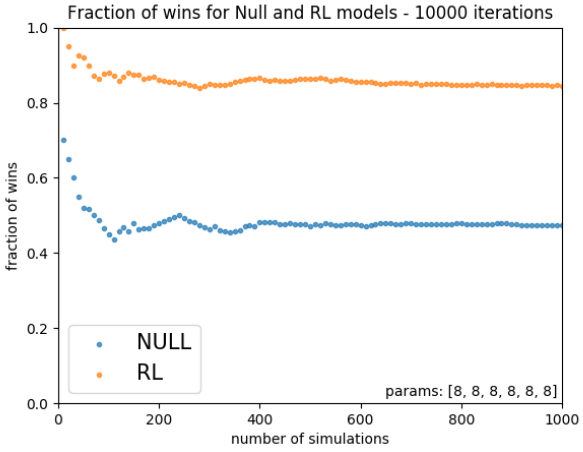

Figure 4: RL vs. Null agent, with eligibility traces

B. Players with Different Skillsets

We now proceed to exploring how Q-learning agents with different skillsets (parametrizations) than their opponents learn to exploit their competitive advantages. First, let's explore how players with a lower *placement* (accuracy) might learn to play differently than their opponents.

Figure 5 shows how Player 1 with skill parameter $a_1 = 6$ performs against Player 2 with $b_1 = 8$, all else equal. As expected, when both agents follow the Null strategy, Player 2 dominates Player 1 (Player 1 wins < 40% of the time). When Player 1's strategy is developed using Reinforcement Learning with 10,000 iterations, however, Player 1 still wins the majority of the matches played, specifically with a win percentage of ~ 63%.

Taking this notion to the limit, *Figure 6* shown a match-up between Player 1 with $a_1 = 4$ and Player 2 with $b_1 = 9$. As shown by the win fraction convergence, 100,000 iterations are enough to lead Player 1 with a Q-learning state-action function to a narrow win margin of ~ 53%.

In order to inspect some of the differences in the state-action decisions between players with the above difference in skill, we pitch *two Q-learning agents with a parametrization of* $[4, 9, 8, 8, 8, 8]$ against each other. *Table 2* shows the actions chosen by each player when sthe ball is heading to quadrant pos with spin $spin$, and player is currently positioned at quadrant cp.

ex	pos, spin, cp	P1	P2
1	TL, SL, TR	**TL**	**TR**
2	BL, SL, BL	**BL**	**TL**
3	BR, SL, TL	**TR**	**BL**

Table 2: Actions chosen by RL (P1) vs. Null (P2) agents

In Example 1 above, Player 1 being less accurate at hitting the ball, decides to run to the left side of the court (TL) to maximize their chances of successfully hitting the ball. Player 2, who has significantly higher skill, stays in TR, because his weighted probability of being successful is higher, and maintaining the TR position is advantageous for the future of the point (he would be exposing his entire right side of the court). In Example 2, Player 1 expects the ball at BL and decides to stay there to maximize the probability of successfully returning; Player 2 plays more aggressively and decided to move up (TL), his high skill making him more likely to succeed in this aggressive play, and if successful he will have the advantage of having a strong net position in the next ball return. In Example 3, Player 1 decided to move to the right side of the court (TR) where the baseline probability of successful return is high; the more skilled player 2 decided to move to BL in order to maximize their success in future points (staying on the net is risky).

ex	pos, spin, cp	P1	P2
1	TL, TS, TR	**BL**	**TL**
2	TR, SL, BL	**TR**	**TL**
3	BL, SL, TR	**BL**	**BR**

Table 3: Actions chosen by RL (P1) vs. Null (P2) agents

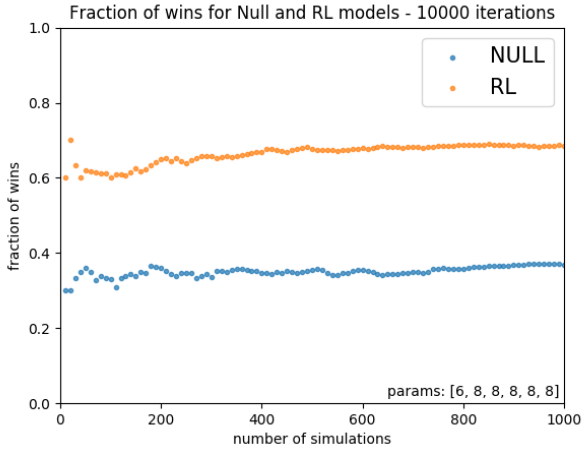

Figure 5: Fraction of wins of less skilled P1

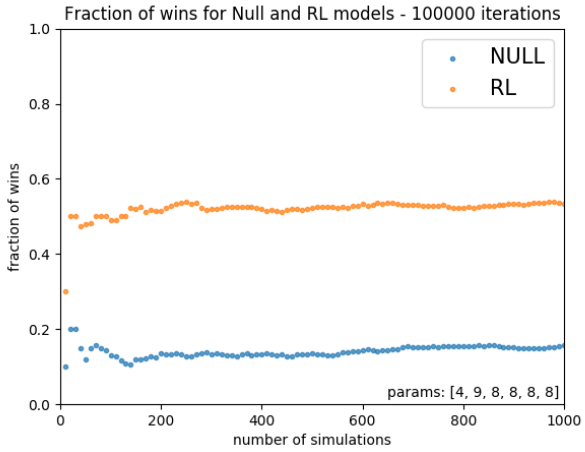

Figure 6: Fraction of wins of less skilled P1

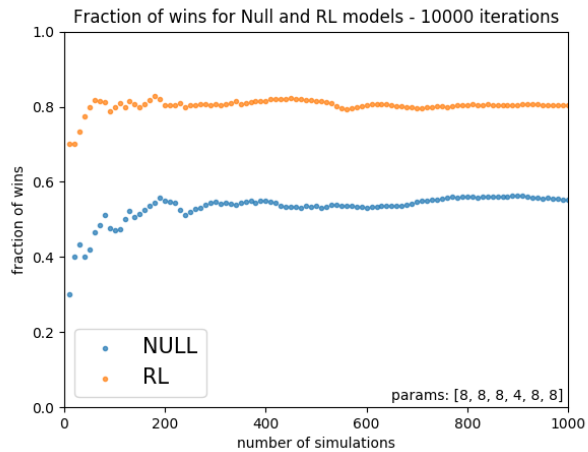

Figure 7: Fraction of wins of quicker P1

Finally, we can explore how players' optimal decision function is shaped when their velocity parameters vary. *Figure 7* shows how Player 1 with skill parameter $a_2 = 8$ performs against Player 2 with $b_2 = 4$, all else equal. As expected, in the Null vs. Null match-up, Player 1 performs better (wins ~ 55% of the time) against Player 2, the former being faster than the latter. Equivalently, in the Q-learning vs. Null match-up with 10,000 iterations, Player 1 wins ~ 81% of matches. *Table 3* shows the actions chosen by each player when the ball is heading to quadrant pos with spin $spin$, and player is currently positioned at quadrant cp.

Player 1 is faster than Player 2, which is reflected in the learned action choices of Player 1. Keeping in mind the non-deterministic, let's see how their choices differ:

- In example 1, Player 1 is fast enough to attempt to run cross court to BL (action with lowest baseline probability of success) in order to control subsequent ball exchanges better (returning a top-spin hit on the net is very likely to lead to a lost point.). Player 2 doesn't run fast enough, so they opt for the best solution available, namely holding the net at TL.
- In example 2, Player 1 is fast enough to attempt to run to cross-court to TR and hit the volley (hitting an incoming slice ball on the net is very desirable.) Player 2 is not fast enough, so they opt for the next best alternative, which is moving to TL, which has a higher baseline probability of success.
- In example 3, equivalent to example 2, Player 1 decides to run back and cross-court to BL and hit the ball as well as possible. Player 2 not being fast enough opts to run backward but stay on the right side (BR)

V. Conclusion & Limitations

Deciding how to move in a tennis court in order to optimize match performance is a complex task that greatly depends on the skills and ability of the competing players. After all, that's why we observe a great variety of game styles by professional tennis players, each of which spends significant amounts of time personalizing their gameplay and optimizing their actions given individual strengths and weaknesses. In this paper, we attempted to model this complicated process as a semi-stochastic game. The general results of Q-learning agents vs. Null agents show a significant improvement from the Null state-action policy to the obtained Q-learning policy. Even at 100,000 iterations of Reinforcement Learning, training ran for ~10 seconds, which suggests more computational resources could be employed to potentially make such policies even better. In addition, the Q-learning agents do seem to learn policies that are tailored to their skillsets, which was the second goal of this project.

There are several potential improvements that could render the simulation more realistic and the results more useful overall. A more expressive parametrization of players' capabilities should be the starting point of such improvements; although reduced to 3 variables in this project, there are dozens of areas of play that competitors can actually be compared at. For more player specific (and more realistic) policy computations, a more expressive scheme could be devised to account for more such areas of play.

Another significant improvement would be collecting real-life professional tennis data to compute maximum likelihood estimates as baselines for specific actions and outcomes. For current lack of such a dataset, tennis experience and expertise were used instead as a baseline for accuracy and stochasticity of outcome. However, quantifying baseline probabilities using observed data over a long period of time would likely result in a much more reliable simulation overall.

GitHub Link

The code used to run simulations and reinforcement learning, as well as all parameters used in the described models, are available at *github.com/steliosrousoglou/238*.

References

[1] Kochenderfer, M. (2015). Decision making under uncertainty : theory and application. Chapter 5. Cambridge, Massachusetts: The MIT Press.

[2] K. Newton, Paul & B. Keller, Joseph. (2005). Probability of Winning at Tennis I. Theory and Data. Studies in Applied Mathematics. 114. 241 - 269. 10.1111/j.0022-2526.2005.01547.x.

[3] George, Stephen. (1973). Optimal Strategy in Tennis: A Simple Probabilistic Model. Appl.Statist.. 22. 97-104. 10.2307/2346309.

[4] Ferrante, Marco & Fonseca, Giovanni & Pontarollo, Silvia. (2017). How long does a tennis game last?. Proceedings of MathSport International 2017 Conference.

[5] Wang, H., Emmerich, M. & Plaat, A. "Monte Carlo Q-learning for General Game Playing," unpublished.