Fantasy Football Roster Prioritization Using Q-Learning

Daniel Morton, Walter Manuel, and Zahra Ahmed
Stanford University, Stanford, California, 94305

This work applies reinforcement learning to the game of Fantasy Football. Fantasy football is a game where human agents receive points each week based on how well American Football players on their fictitious roster perform in the National Football League (NFL) each week. The uncertainty present in both actual and fantasy football leads to the problem of how an agent playing fantasy football can prioritize different football positions and sequentially make lineup changes to a fantasy roster each week in order to maximize performance over the course of an NFL season. We formatted this problem as a Markov Decision Process (MDP) and employed a model-free approach using Q-learning to learn the optimal roster decision policy. To accomplish this, the problem space was constrained to three player positions, and a limited set of actions that involved swapping players with others at their position or keeping the same roster for the week. Also, a reward function was developed based on both the relative performance of the new roster compared to the previous roster and transaction costs for player swaps. Once completed, the model was trained on multiple seasons of fantasy football data. The policy generated from the trained model was able to outperform a baseline random policy the majority of the time when tested on different NFL seasons.

I. Introduction

Fantasy football is a popular game that many fans and followers of American Football play throughout the football season. In fantasy football, participants in a fictitious "league" each create a "roster" of players made up of players in the National Football League (NFL). Every time a player on your fictitious roster plays in real life, you win or lose points based on their actual performance. The objective of the game is to choose the best players for your team to maximize your points. In season-long leagues, there is typically a fantasy draft at the start of the season, where the participants in the fantasy league are assigned a draft order and pick players to fill out their roster. Then, throughout the season, each week the participants have to decide who to start on their roster in order to win weekly match-ups against other human agents in the league.

Uncertainty is inherent to the game of fantasy football, as a player's performance can often be unpredictable from week to week. This performance can be affected by numerous variables, such as the skill of the team the player is competing against that week, the weather, the player's health or proclivity for sustaining injuries, and more, in addition to simply pure luck. There are also multiple constraints in fantasy football, so that a human agent cannot add a better player without giving something up in return. Ultimately, this makes the game of fantasy football difficult to play and requires strategical thinking.

This project modified the traditional fantasy football objective and instead focused primarily on the week-to-week roster adjustment decision over the course of the season. At each week, given our current lineup of players and their ranking with respect to the league performance in the previous week, the agent made a decision: should we trade a player, or not? If so, which player should we trade, and what type of trade should be made? Thus, the primary goal of this project was to solve the problem of how to prioritize positions and make lineup changes to a fantasy roster each week in order to maximize performance over the course of an NFL season.

II. Literature Review

Prior research into using machine learning for fantasy football has primarily focused on optimizing a single lineup by modeling it as a variation of the knapsack problem using integer or mixed-integer programming [1][2]. For the sequential decision-making process that this project focuses on, the application to fantasy football has been limited. However, similar sequential allocation problems have been addressed for various other applications, including in other fantasy sports [3] as well as stock trading [4].

In one source, researchers attempted to optimally form a fantasy soccer team lineup from players competing in the English Premier League [3]. They formulated the problem as a belief-state Markov Decision Process, where the beliefs are the player's characteristics, an action is the selection of a certain fantasy lineup, and the observations are the players available, matches to be played, and the outcomes of the prior week's matches. They then created a Bayesian Q-Learning algorithm and evaluated its efficacy. Similarly, Neuneier employed Q-learning to determine optimal financial asset location [4]. They modeled the problem as a traditional MDP with the state composed of elements representing both the financial market as well as an investor's current portfolio. In order to manage the large state space, Neuneier used neural networks to approximate the action value function instead of solving it discretely. These prior works make it clear that Q-learning is an appropriate and proven method for approaching allocation problems.

III. Approach

For this project, we framed the problem as an MDP with a relatively small state space and employed a model-free approach to learning the optimal policy for making roster decisions. Q-learning is particularly well suited for this problem due to the stochastic nature of the reward and transition dynamics. All the coding involved in creating our model, processing data, conducting rollouts, and running Q-learning was done using Julia.

To limit the scope of the problem, we focused on a modeling a single agent that makes decisions about a three-player roster, independent of the other participants in the league. This agent must decide how to prioritize upgrading their roster each week based on the results from the previous week's game. In order to further increase the tractability of the problem, we focused on a limited number of available players and actions.

A. Model Definition

1. State

Each roster consisted of selecting a single player for each of three positions: Quarterback (QB), Running Back (RB), and Wide Receiver (WR). Our league is comprised of 8 potential players for each of the three positions. The state space is then equal to the number of permutations of those 8 players for each position, giving us a total of 512 possible states.

The state was defined by the rank of each player on the team instead of by player name in order to generalize the model so it can be applied to different groups of players. The ranks were determined by number of fantasy points scored each week. The tuple of player ranks was then encoded into a single integer from 1 to 512 using binary. Figure 1 shows an example of how the state is encoded.

Fig. 1 The state encodes the ranks of the chosen lineup

2. Action

Our seven possible actions are listed in Table 1. Trading up or down takes the ranking for the current week and then swaps the player for the player immediately above or below them in the ranking. Doing nothing means that the previous week's lineup will be carried over into the next iteration. While this is not the way that trades are made in actual Fantasy Football leagues, this was intended to represent the overall trading process while simplifying our action space.

Table 1 Actions

1	Swap QB Up
2	Swap QB Down
3	Swap RB Up
4	Swap RB Down
5	Swap WR Up
6	Swap WR Down
7	Do Nothing

3. Reward Model

Our reward model incorporated two main components: a "transaction cost" every time you swap a player, and the roster's performance as a function of the number of fantasy points it scored.

For the transaction cost, we penalized the agent for making a trade upwards and rewarded the agent for deciding to trade down. If the agent does not make a trade (action 7), the transaction cost is 0. Each position receives a different scaling based on the expected number of fantasy points the position traditionally scores - in our case, there is a higher scalar associated with the quarterback since players at this position tend to score more points. The entire transaction cost was multiplied by a scalar so that it was the same order of magnitude as the fantasy points. This transaction cost reflects actual fantasy football, which has costs associated with acquiring new players, such as trading away a player of equal value, or spending a portion of a fictitious budget. For the fantasy points, we used the difference between how well the current roster performed for the week and how well the roster from the prior week (before the agent took an action) would have performed in the current week. The fantasy points were obtained from the rollout.

$$P = \begin{cases} 2 & Quarterback \\ 1.5 & Running\ Back \\ 1 & Wide\ Receiver \end{cases}$$

$$Transaction\ Cost = \begin{cases} G * P * \frac{-1}{N} & Action\ \epsilon\ [1,3,5] \\ G * P * \frac{1}{N} & Action\ \epsilon\ [2,4,6] \end{cases}$$

$$Reward = (Transaction\ Cost) + (Fantasy\ Points\ for\ New\ Lineup) - (Fantasy\ Points\ for\ Old\ Lineup)$$

where,

P = Position Scalar

G = General Scalar to increase importance of transaction cost

N = New Rank of the position that was swapped

B. Rollouts

In order to simulate an NFL season playing out each week, we conducted rollouts using data from the 1999-2019 NFL seasons. The data on weekly fantasy football player performance used to train and test our algorithm was downloaded from `https://www.fantasyfootballdatapros.com/csv_files`. We parsed the data into CSV files for each week of each season. The CSV files contained the player names, positions, and total fantasy points scored. To conduct our rollouts, we sequentially stepped through each week to re-calculate the ranks of the players in our state space and determine our reward for that iteration.

C. Q-Learning

After defining our model and the necessary inputs for Q-Learning (state, next state, action and reward), we implemented Q-learning on the 2017 and 2018 seasons for 1000 episodes. For each episode, we initialized with a randomly generated lineup and then stepped through each of the 17 weeks of the football season. For our Q-learning parameters, we chose a learning rate, α, of 0.5. In order to adequately explore the state space when training the model, we implemented an ϵ-greedy exploration strategy with $\epsilon = 0.9$, which we decayed every iteration at a rate of $\alpha = 0.9$.

IV. Results

A. Q-Learning Policy

After training our model, we extracted a policy from our action-value matrix. The frequency of each action as seen in the extracted policy is shown in Figure 2. Our agent learned that it was more advantageous to swap a player up rather than down, despite the negative transaction cost associated with swapping up. More specifically, the agent preferred to swap the quarterback up over the other two positions, which is in line with what we would expect as quarterbacks tend to score more points than running backs and wide receivers. Additionally, our agent preferred the "Do Nothing" action over every action except for swapping a quarterback up. Since player performance can vary substantially from week to week, retaining the same lineup even after a poor performance can often be an optimal decision. Therefore, the agent's preference for the "Do Nothing" action is representative of a common strategy employed by human decision-makers when playing fantasy sports. In particular, the agent preferred the "Do Nothing" action for states associated with higher ranks for all three positions, essentially assuming that a higher ranked player is more likely to continue performing well.

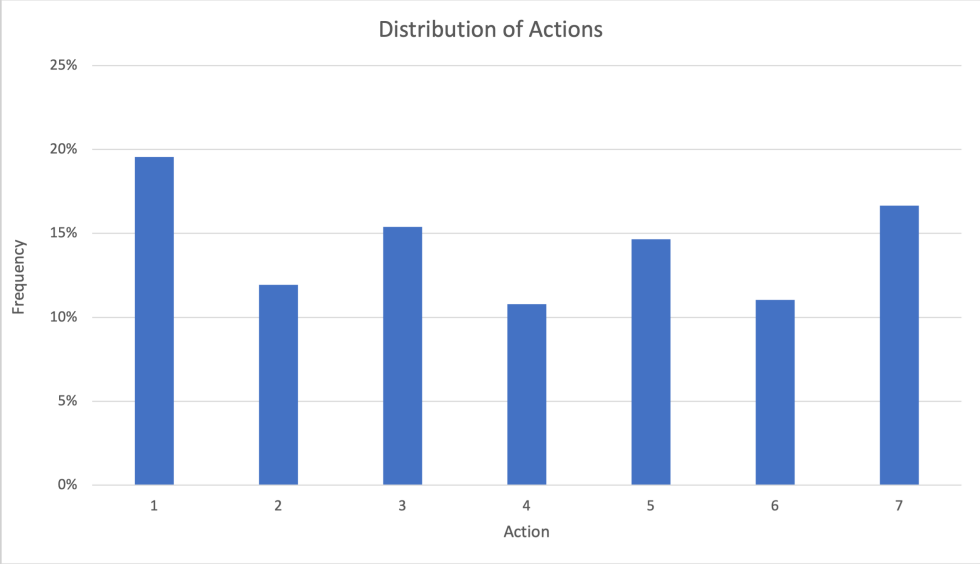

Fig. 2 Frequency of each Action in Learned Policy

B. Comparison Against Baseline

After running Q-learning on the 2017 and 2018 seasons to extract a policy, we tested this policy on every season from 2002 through 2012. We established the baseline for comparison to be a random policy. For each season, we obtained the average total reward gained for the season over 20 iterations - each time with a random initial lineup. We compared our learned policy to the random policy to determine the effectiveness of our agent. Table 2 tabulates the average reward per season for each policy while Figure 3 shows the average reward for each season using both the learned policy and the random policy.

We tested our learned policy on various seasons under the assumption that we could expect similar trends in performance in every season. This assumption was largely justified in the results shown in Table 2 which show that the learned policy significantly outperformed a random policy for most years. However, from Figure 3, we can see that there are a few

Table 2 Comparison of Learned to Random Policy

Average Reward Per Season, Learned Policy	2.4102
Average Reward Per Season, Random Policy	-0.8497
Average Reward Above Baseline Per Season	3.2599
Average % Improvement in Reward Per Season	383.66%

seasons (2004 and 2008) where the random policy did better. There are several reasons why this the case. First, our model does not account for the variance in a player's past performance - it exclusively ranks players based on their performance for the current week. Thus, a player who is ranked highly one week could be toward the bottom of the list the next week. For this reason, a random policy has the potential to outperform the learned policy because it can inadvertently take more risks with it's lineup choices. Additionally, our reward function accounts for the difference between the current and previous lineup's performance in terms of fantasy points. Therefore, a random, favorable trade during a single week in the season can provide enough positive reward to offset potentially negative rewards for the rest of the season. There are several ways to improve the robustness of the model in order to account for these biases which were outside the scope of this project but will be addressed in future work.

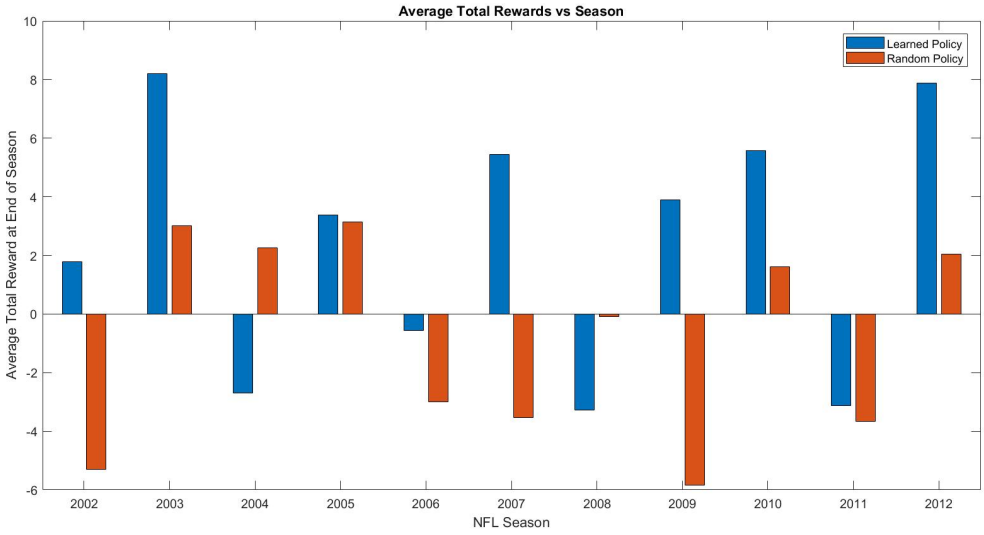

Fig. 3 Comparison of Learned to Random Policy by Total Reward Per Season

V. Conclusion

In this report, we considered a reinforcement learning-based approach to fantasy football lineup management and roster prioritization. We implemented a Q-learning model using condensed fantasy lineups as states, swapping players at a position up or down according to their rank as actions, and a reward function based on the transaction cost to swap a player and the improvement in fantasy points scored by a lineup. This model was trained on actual historical fantasy football data using an epsilon-greedy exploration strategy, and then the extracted policy was tested on a different set of historical data. The results indicated an average improvement of 3.2599 reward points, or 383.66% per season, for the learned policy over the baseline random policy. Furthermore, the model was able to learn to prioritize some actions over others in a logical way, such as swapping the QB up (QBs score more fantasy points) or doing nothing (stability is often a winning strategy to deal with the week-to-week volatility of fantasy football). These results demonstrate that our approach was valid and could be useful if extended further in scope and complexity.

VI. Future Work

Future work on this project would include many improvements to our model to capture the uncertainty and variability in the domain of fantasy football. Our current model only accounts for a player's current rank, and does not take past player performance and trends into consideration. We began work on a model that encodes player mean and variance into the state, and while it was ultimately not used for this project, further developing this advanced model could result in generating a policy that can outperform a random policy more often. By incorporating historical data, the model will be more equipped to make predictions about a player's future performance. There are also many variables beyond a player's individual traits and history that can affect outcomes in fantasy football, such as weather conditions and the quality of the opponent a player is facing. Finding a way to incorporate these factors into our model so that it can learn how they affect player performance could lead to significant improvements to the outputted policy.

Another area for future improvements would be to expand the state and action space to make it more representative of the formats used in other fantasy leagues. A standard fantasy football roster has a total roster size of 15 players (9 starting spots and 6 bench spots), with 6 positions (QB, RB, WR, TE, K, DEF). Our extremely simplified model only considered a 3-player team that started one player at the QB, RB, and WR positions. We also limited the action space to swapping one position's player each week, whereas in real fantasy football, multiple trades can be made in a week and players can be exchanged with players at different positions.

VII. Contributions

All three team members contributed equally to the project. Walter was responsible for obtaining, processing, and integrating all of the data used for the rollouts and testing, as well as for post-processing the results of Q-learning and random policies. Daniel was responsible for implementing several versions of the model that accounted for various parameters in the state space as well as helping to integrate all of the code. Zahra was responsible for integrating the rollouts and reward function with the model and implementing the Q-learning. All three members contributed to scoping the project, debugging code, and writing the report.

References

[1] Becker, A., and Sun, X. A., "An analytical approach for fantasy football draft and lineup management," *Journal of Quantitative Analysis in Sports*, Vol. 12, No. 1, 2016, pp. 17–30. https://doi.org/10.1515/jqas-2013-0009.

[2] Hunter, D. S., Vielma, J. P., and Zaman, T., "Picking Winners Using Integer Programming," 2016. https://doi.org/arXiv:1604.01455v2.

[3] Matthews, T., Ramchurn, S. D., and Chalkiadakis, G., "Competing with Humans at Fantasy Football: Team Formation in Large Partially-Observable Domains," *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, Association for the Advancement of Artificial Intelligence, 2012.

[4] Holleis, P., Wagner, M., and Koolwaaij, J., "Enhancing Q-Learning for Optimal Asset Allocation," *Advances in Neural Information Processing Systems 10*, 1997.