FINAL REPORT FOR CS394R - ROBOHEARTS:
SIMPLE STATES: THE IMPORTANCE OF FEATURES IN A PARTIALLY OBSERVABLE MARKOV DECISION PROCESS

Lucas Kabela
lak2258
lucaskabela@utexas.edu

Akash Kwatra
awk457
akashkw@utexas.edu

ABSTRACT

We examine the results of using various combinations of raw state features in value approximation for the game of hearts. Features for the cards in a player's hand, the cards in play, the cards previously played, the cards won by each player, and players' scores were evaluated in both simple linear approximation and nonlinear approximation with a neural network. Our results indicate that both simple linear and nonlinear function approximation can learn to perform between 27.23-35.84% better than random with just the feature for cards in the player's hand, with nonlinear methods outperforming linear methods at a 5.12% difference with this limited feature representation. Building on the neural network approximation, we try approximation with restricted sets of raw state features, and find a combination of cards in hand and cards in play can reach a 34.35% improvement over random agents, while the entire feature set only reaches 30.94% improvement. Using this restricted feature set, we produce an agent using the policy gradient algorithm REINFORCE with baseline which wins 56.60% of games, an improvement of 127.40% over random, and a 33.98% improvement over REINFORCE with the full set of state features in the same number of training epochs.

1 INTRODUCTION

Reinforcement learning[1] has enjoyed recent success in many game settings, thanks in part to the emergence of deep reinforcement learning in recent years. One domain which reinforcement learning has excelled in is card games, with recent results such as those of Brown & Sandholm (2019) outperforming professional poker players. However, networks fueling these improvements have large quantities of raw state information, such as the work of Mnih et al. (2013) on Atari, in which a deep network processed a 84 x 84 gray-scale cropped region of the game state, which totals 7056 values per frame, with frames then stacked 4 at a time, resulting in roughly 28224 parameters. As problems in the learning community continue to scale with the computational cost and complexity of deep networks, it is worth investigating if there is a simple paradigm in which the important regions of raw state space can be identified, and thus model sizes and training time can be reduced with minimal loss in performance.

For our initial inquiry, we seek to obtain empirical evidence in a domain which relies heavily on state representation and has a rich history of feature analysis. Card games are one such domain, in which features can provide not only direct information from processing, but also lend themselves to inference by clever agents. One such game is Hearts, which is naturally phrased as a POMDP in which a player seeks to minimize the points they win. Due to the partially observable nature of this game, literature for feature abstractions and state representation is dense. Furthermore this domain has seen impressive gains due to raw feature input in neural networks in recent work by Wagenaar (2017). Finally, the entire raw feature space for the domain is rather small and hence lends itself to study under different combinations of features. Thus, we find it natural to select this domain to address our question of if restricted subsets of raw state features can serve as a proxy for or even improve performance over the full set of raw state features. To answer this question, we attempt to

[1]The data collected, trained models, and source code is available at https://github.com/akashkw/robohearts. A video explaining this study can be found at https://youtu.be/n8zhiRtXqHM

empirically study the effect of different combinations of restricted state feature sets in both linear and nonlinear function approximation.

2 THE GAME OF HEARTS

2.1 DESCRIPTION OF THE RULES OF HEARTS

We believe Hearts to be a unique setting for a study of raw state features and state simplification as a result of the many complexities of its rules, which we describe here. The game of hearts is a trick-winning game, a "trick" being defined as a collection of four cards won after each player places down a card. In this game, four players are dealt 13 cards to play one round, composed of 13 tricks. The winner of each trick is the player that has placed down the highest card of the leading suit. In the event a player does not have a card of the leading suit, any card may be played, with the exception of the first trick - in which point bearing cards cannot be played. For each heart in a trick, the winner of the trick gets one point. The winner of the queen of spades receives 13 points. After 13 tricks, points are counted, new cards are dealt, and another round begins. These rounds repeat until one player accumulates 100 points, at which point the player with the fewest points is declared the winner.

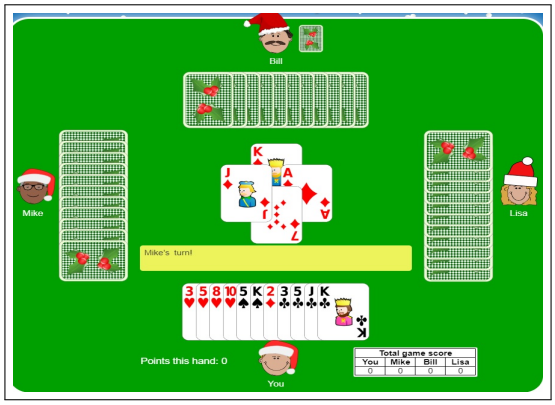

Figure 1: A trick of Hearts, game available at `https://cardgames.io/hearts/`

If a player manages to win every point available in one round, they have "Shot the Moon". Shooting the moon awards 0 points to the winner, and 26 points to all other players. In one round, the player with the 2 of spades leads the first trick, with the winner of the previous trick leading all subsequent tricks. Hearts cannot be used to lead a trick until someone has "broken hearts", meaning someone has placed down a heart in a previous trick. A final rule is that before each round, players can select three cards from their hand to trade with another player.

2.2 HEARTS AS AN MDP

Before beginning our investigation, we must define an environment compatible with the rules of Hearts for our reinforcement learning agent. We shall use the rules of the game to shape the formulation of our MDP. We begin by defining the environment as temporally discrete, as the game of hearts can easily be broken up into logical units of time. In decreasing order of granularity, we have the trick, which corresponds to a single timestep the agent selects an action in, the round, and the game, which we consider an episode for training purposes.

The state can be described as the current score of all the players, 0 to 100, the point bearing cards that have been won by players, the cards that have been played in the current round, the cards in play in the current trick, and the cards in each agents' hand. This is the full extent of state information needed to define a game; however, it is not possible to fully observe the state of the game, as the cards in other players' hands are hidden - therefore it is natural to formulate Hearts as a POMDP. Similarly, the action space is the choice of card to play from the agent's hand, and up to 3 cards to trade at the beginning of each hand. This actions space is dynamic - the size of the action space is

constantly changing between each trick, and is not even fixed for card passing, as the cards available to pass will be different each hand. The negative of the score received from playing a card serves as the reward function, which is a natural formulation as the agent will seek to minimize score. Lastly, the game of Hearts is clearly multi-agent, with up to 4 players in classic settings, and each player selecting actions based on observations of the environment.

2.2.1 RESTRICTED FORMULATION

To produce a tractable setting for the exploration of our problem, we make several restrictions to the full MDP. Our first major relaxation is to ignore partial observability. While other works such as Ishii et al. (2005) have dealt with partial observability by attempting to model other players' hands, we find a full POMDP framework overly strict for investigating raw state features. Hence, we choose to ignore this aspect of the game, instead opting to add history of cards played as a raw feature of state for our agent. Another important abstraction is our treatment of action space. We have already acknowledged the complex and dynamic dimensionality of our action space, so to address this issue, we treat the action space as being a constant 52 dimensions, and simply mask the output of our approximators to only allow valid moves for actions. The other actions, passing cards, is a drastically different task that may require different features, so we randomized the 3 cards we pass, effectively abstracting card passing as part of the environment. Lastly, we relaxed the multi-agent setting, considering enemy agents as part of the environment. We believe this assumption to be especially strong, as the agents we train against are random, and do not have evolving strategies. With these relaxations, we gain tractability, and turn to previous works for motivation.

3 RELEVANT WORKS

The game of Hearts has been heavily studied in terms of feature engineering and abstractions (Sturtevant & White, 2007) (Sturtevant, 2003) (Wagenaar, 2017) (Ishii et al., 2005). Our work draws inspiration from the features, abstractions, and motivations of previous research in this area heavily, and uses the important concept of state abstractions to motivate our investigation into the importance of the raw features in states, and their abstractions (Ho et al., 2019). However, we found most of the prior work to be outdated, as the bulk of it was published pre-deep learning and relied on heavy feature engineering. Therefore, to fully leverage modern techniques and learning in this domain, we looked into more contemporary efforts to solve games such as the work of Brown & Sandholm (2019) Silver et al. (2016) and Charlesworth (2018). Wagenaar (2017) was the only paper in our review for Hearts that successfully incorporated neural networks, and thus served as a starting point for our model architectures and feature representation.

One strategy we found particularly prominent in more modern works was the idea of using neural networks for function approximation and action selection, and training them through policy gradient techniques, with some of the most popular techniques being Proximal Policy Optimization Schulman et al. (2017) and Neural Fictituous Self Play Heinrich & Silver (2016), both of which have achieved remarkable success. After further investigating these techniques, we empirically believed the entire suite of policy gradient methods to be particularly well suited for our domain, as we believed the value function for Hearts to be incredibly complex and hard to model, relying extensively on combination of cards and history to inform the policy. Therefore, we turned to REINFORCE from the work of Sutton et al. (1999) as a simple method for obtaining an upper bound on the performance of our restricted state representation.

4 STATE ABSTRACTION

From our investigation into previous research, it is apparent that the state space of Hearts is intractable to learn using a tabular method. Hearts is not alone in this issue - many card games explode exponentially in the number of states that can be encountered during play. Fortunately, a great deal of prior work such as Sturtevant & White (2007) and Sturtevant (2003) has been done into abstractions via feature engineering in the game of Hearts. However, little work has been done into manipulating the raw state space, with feature engineering utilized heavily to extract semantics from the raw state. Thus, we first decompose the raw state space into a natural restriction of 5 sub-components.

The first natural decomposition of state space is the agent's hand, as this naturally this defines the action space. To model this portion of state space, we produce a binary feature vector for the presence of cards in the agent's hand, which has 52 entries for each card. This constant length encoding enables us to use one approximation function for every trick in a round, as opposed to previous work (Wagenaar, 2017). Another important portion of state space to decompose is the cards in play during a trick. Without a representation of these cards, it is not possible for the agent to know if it will lose a hand before playing its card, although, due to the partial observability of the problem, this cannot be learned perfectly even with this feature. Therefore, we also modeled the cards in play as a 52 length binary feature vector.

As imperfect agents, we believe these two components are all the representation humans will build during a game of Hearts. However, expert card players can utilize more information during the game in the form of the history of cards for a round, which minimize uncertainty of opponents' hands. Thus, we encode the history of cards played in a round with another 52 length binary vector. Similarly, player scores may also inform player strategy so we encode this feature as a length 4 integer vector, with values ranging from 0 to 100. Lastly, the state space must also contain a set of features to prevent shooting the moon. Therefore, we add in four 14 length binary vectors, one vector per player, and one entry for point bearing card that is won by said player. We believe these five features naturally decompose the state space of hearts, and are sufficient to define the state of any given game. Furthermore, they are "raw" state features in the sense they do not require pre-processing to produce, and are immediate from observations of the environment.

While we believe these features to be complete, other approaches such as Sturtevant & White (2007) attempt to further distill this information in favor of specific heuristic based features, such as number of spades played, or number of hearts in hand. While we find such abstraction important, as addressed by Ho et al. (2019), we opt to pass raw information into a large neural network, in imitation of previous work by Brown & Sandholm (2019) and Charlesworth (2018) which have produced some of the most impressive gains in recent learning. Further, we find feature engineering can over-reduce information the agent may find necessary. Therefore, we choose the decomposition of raw feature space to the five sub-spaces we have defined.

5 Monte Carlo and Classical methods

5.1 Linear vs Non-Linear Function Approximation

Our initial inquiry was to compare the effectiveness of linear and nonlinear value function approximation. Intuitively, nonlinear representations can model more interesting functions than linear representations, such as the xor. Consequently, we expected the nonlinear model to greatly outperform the linear model. Both techniques were trained against random agents for 10000 episodes, then evaluated on 25 sets of 1000 games against random agents, with the average number of wins recorded. The models were learned using Monte Carlo roll-outs, due to efficiency of roll-out samples we could collect, and low bias of this method. For this section, we used the simplest restricted features possible, which was the cards in the agent's hand as they define the action space.

For the linear model, we used a value computed by the dot product of a learned weight vector and the computed binary feature vector for a given state. The weights were initialized uniformly to 0, and learned with a parameter α = 1e-4, which was found from a hyperparameter study of 5 values of α to be the highest[2]. The policy π was ϵ-greedy with respect to one step look-ahead using the value function, with ϵ set to .01, as found by a simple hyperparameter study [3]. As a parameter of the environment, we found choice of ϵ to be inconsequential, since the environment is naturally stochastic and places the agent in random states naturally; thus, we set this value to be fairly low. Lastly, we performed a hyperparameter study on γ [4], and selected the setting $\gamma = 1$, as rewards are not counted until the end of hands, so it does not matter at what point in the round the reward was incurred.

[2]Refer to Table 6 in Appendix A

[3]Refer to Table 7 in Appendix A

[4]Refer to Table 5 in Appendix A

With this setting, the simple linear approximation won an average of 321.80 games, which was a 27.23% improvement over random, with extremely high statistical significance using the t-test method and a two tailed P value of < 0.0001, with statistics in Table 1. We found these results especially surprising as the feature representation was incredibly limited. The agent only had access to the cards in the hand, meaning the values learned simply corresponded to what cards are "good", and which cards are "bad". From the weight vector we found [5], in particular, the 2 of clubs is learned to be the only positive value, as the agent will often not win tricks if it leads with this card. Similarly, face cards of all suits, but especially hearts are learned to be especially low value, which we expected as face cards are more likely to win tricks and thus points. However, one anomaly with this data is that the values do not change linearly, it appears in almost all suits there is a significant jump from the 10 to the Jack.

	MC w/ Linear	Random
Mean	321.80	252.92
SD	13.76	11.06
SEM	2.75	2.21
N	25	25

Table 1: MC Wins statistics with linear approximation over 1000 Games

For the nonlinear model, we used a 2 hidden layer, variable node network, with 2*input feature nodes in the first layer, and 4*input feature nodes in the second layer, with ReLU for the non linearity. This architecture was chosen following the results of Wagenaar (2017). We trained this network with the Adam Optimizer, and used an almost identical setting of parameters as the linear case, as we expected these parameters to be particularly robust to technique. We did perform a hyperparameter search on for α, and found 1e-2 to be the most effective [6]. We trained the neural network on 10000 episodes with $\alpha = 1e-2, \epsilon = .01, \gamma = 1$, and evaluated on 10 sets of 1000 games. We recorded an improvement over random by 35.84%. Once more these results were statistically significant, with a a two tailed P value $< .0001$ with statistics in Table 2.

Group	MC w/ NN	Random
Mean	338.1	248.9
SD	14.04	13.60
SEM	4.44	4.30
N	10	10

Table 2: MC wins statistics with nonlinear approximaton over 1000 Games

Comparatively, we expected nonlinear function approximation to drastically outperform linear approximation due to its ability to encode interactions between features. However, we discovered the percent difference between these methods to be 5.12%, a fairly low value. This is likely because the network has issues learning a valid value function that is robust to hand size. The environment is highly stochastic and partially observable, and thus incredibly hard to model with any value function. From these experiments, we established, at least in the case of Hearts, that nonlinear function approximation is indeed more powerful than linear function approximation - however, training took roughly 3 hours for nonlinear function approximation, while it only took on the order of 20 minutes for linear approximation. Furthermore, we were able to investigate the hyperparameter settings of this environment and believe our findings to be representative of many learning techniques in this domain. One final observation from this method is just how high variance monte carlo roll-outs can be - from the tables, we observe standard deviations of around 13.39 wins on average.

5.2 Importance of Raw Features

With the results of the previous subsection established, we proceeded to test combinations of raw state features with nonlinear function approximation to investigate their impact on agent

[5] The weights for this model are available in Appendix B

[6] Refer to Table 7 in Appendix A

performance. We used the same process as the previous section - 10000 training iterations, with the two different settings of $\alpha = 1e-2$, and $1e-5$ to get samples of the models under different learning rates. The results and combinations are established in Table 3 and Table 4

Feature Combination:	1, 2	1, 3	1, 4	1, 2, 3	1, 2, 5	1, 2, 3, 4, 5	Rand
Average Wins	332.20	274.90	283.80	290.50	326.90	325.90	248.90
Difference from Rand	83.30	26.00	34.40	41.60	78.00	77.00	0.00
Percent Improvement	33.47%	10.45%	13.82%	16.71%	31.34%	30.94%	0.00%

Table 3: Statistics for feature combinations, $\alpha = 0.01$

Feature Combination:	1, 2	1, 3	1, 4	1, 2, 3	1, 2, 5	1, 2, 3, 4, 5	Rand
Average Wins	334.40	276.20	286.81	293.20	324.70	319.30	248.90
Difference from Rand	85.50	27.30	37.91	44.30	75.80	70.40	0.00
Percent Improvement	34.35%	10.97%	15.23%	17.80%	30.45%	28.28%	0.00%

Table 4: Statistics for feature combinations, $\alpha = 0.05$

We first examine all pairs of raw state features, 1,2, 1,3, and 1,4[7]. We did not evaluate 1,5, as we believe the score to be inconsequential in combination with only the hand versus random agents. Here, we find the combination 1,2 - in-hand and in-play to perform best, winning an average of 332.20 games, a 33.47% improvement over random. Perhaps more surprisingly was just how good this combination of features did compared to others. Comparatively, the next best combination was 1,4 in-hand and cards won, which we believe to be the next best due to the inconsequential information provided by cards won (4), which is to prevent shooting the moon, something random agents are highly unlikely to do. We hypothesize 1,3, or in-hand and cards played could perform better given more training time, but is difficult to learn due to the complexity of the value function. Therefore, the important observation of the data from pairs is that not all representations of raw state information are the same, and removing some information may improve the learning of models.

Next, we looked at higher order combinations of features. For our purposes, we did not try triplets including cards won, as the agents were random and highly unlikely to shoot the moon. Thus, we tried three triplets of in-hand, in-play, score, and in-hand, history, score. We found the best triplet was 1, 2, 5, or in-hand, in-play, score, with percent improvement of 31.34% over random. However, it is interesting to observe that this triplet did not perform better than just in-hand and in-play. Lastly, when we tried the entire feature space, which again performed worse than 1, 2, only reaching 30.94% improvement over random. By the t-test method, we find that the two sided P value to be .0355 for this method, which while statistically significant by some metrics, is not nearly as convincing as our previous work. Therefore, we conclude given limited training time, a restriction of the entire raw state feature space can indeed perform better than the full raw state features, and are certainly compatible as proxies for some pairs of state features, although not all.

6 POLICY GRADIENT

Lastly, based on the results found in section 5, we wanted to provide an upper bound on the accuracy of the more limited and entire set of raw state features using a method we believed to be more suitable to this domain. We noticed firstly, due to stochasticity, a stochastic policy would be more compatible. We also observed the difficulty of MLPs applied to this setting in learning consistently. Thus, we implemented a policy gradient method, using a value network as a baseline for the REINFORCE algorithm, which to our knowledge, is the first such agent in Hearts[8]. The output of the policy gradient network was a 52 length vector converted to a soft-max distribution, which models the probability of selecting each card to play. We then applied a mask of the binary representation

[7]We refer to features here sometimes by number - see Appendix B for translation

[8]REINFORCE was selected as we found no previous work with policy gradients in this domain. Thus, this is a policy gradient proof of concept, and the most established of these methods

of valid cards and re-normalized to produce a distribution over possible actions for a given hand. We trained two models with this network under the REINFORCE framework: one model used the features 1,2 while the other network trained with the entire set of raw state features. We observed the model with the limited feature set achieved 56.60% wins over random after 100 epochs of training at a learning rate of 1e-4, which was a 127.40% improvement over random. Contrasting this, the full feature set of features achieved 40.16% wins over random after 150 epochs of training at a learning rate of 1e-5, which was a 61.35% improvement over random. We selected these learning rates as they were the highest we were able to train the models with which would not diverge. For a comparison of training over 100 epochs, refer to Fig 2. Our experiments found both REINFORCE models trained much faster than the nonlinear neural network approximation, and vastly outperformed nonlinear value function models, confirming our hypothesis that this domain is naturally suited for policy gradient methods. This is likely a result of how we modeled the policy (epsilon greedy) which required up to 13 look-aheads for the value function network.

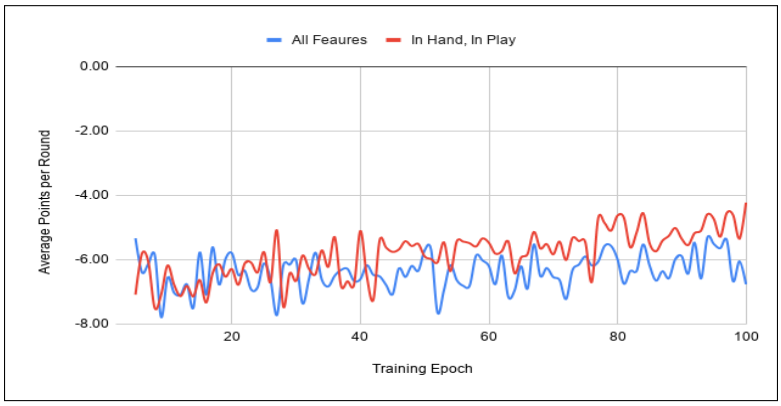

Figure 2: Average Reward per Round vs Training Iteration

7 Results and Discussion

Our experiments have several interesting implications for raw feature representations, and the general goal of artificial supremacy in Hearts. Our first study, the evaluation of linear vs nonlinear approximation, reaffirmed an established idea in reinforcement learning for function approximation - nonlinear models can better represent functions than linear models, but this representation power comes with increased computational complexity, as our nonlinear model achieved a 5.12% difference in wins over the linear model compared to random performance, but required almost 10 times as much training time. Additionally, we affirmed some intuitive hyperparameters of this environment through several studies.

The results of our feature study imply that at least in the face of limited computational power, restricting the raw features of networks can lead to, and in some cases exceed the performance of full raw feature sets. These results manifested in the 35.14% improvement over random of the in-hand and in-play features, which performed better than the full set of features at 30.94% improvement over random. Using the empirically more suited policy gradient methods for this domain, such as REINFORCE, we further show a reduced set of raw state features achieves 33.98% improvement over the full set of features at least in a limited training of 100 epochs with 16 roll-out batch sizes. While best results in high computational environments can likely be achieved with the full feature set, this study indicates a restricted feature set can reach near optimal performance using much less computation with the correct combination of features.

The last implication of this study comes in the domain of Hearts itself, where there has been extensive research to produce artificial agents that can achieve human performance. However, there are many severe limitations to artificial intelligence in Hearts, mostly as a result of stochasticity in the environment. As a result of the partial observability of the environment, optimal score cannot likely be achieved by any A.I. Regardless, while performing this study, we thought it would be beneficial for a more accurate baseline to compare our agent against. Other methods, such as Wagenaar (2017)

have used heuristic based agents as baselines, while ours makes use of random agents. We believe a better comparison can be found in the form of an optimal offline algorithm. If such an agent were produced, results could be standardized across the learning community. Thus, we acknowledge the shortcoming of our results, as these are compared against random agents. On the other hand, we believe random agents have no strategy, and thus instead of learning strategies of other agents, our agents learned direct information about the goodness of the environment, in part encouraged by our relaxation of multiagent conditions. Thus, while not directly applicable due to training with different agents, we compare our REINFORCE network to the results of Wagenaar (2017), and find that we achieve an average of 566 wins in 1000 games and 1 hour of training time, compared to this papers result of 539 average wins with a training time of 5 hours over 13 networks[9]. These results indicate that policy gradient methods are much better suited to Hearts than MLPs with value function approximation, and thus pave the way for future research into artificial supremacy in Hearts.

This work was not without difficulty though - we ran into several problems during work of this project. One particular challenging task was exceeding 30% win rate over random agents. We hypothesized this was possible thanks to the results of Wagenaar (2017), however we greatly struggled to train our MLP to this level. One likely cause of this was that we used one network for every hand, while the work of Wagenaar separated each number of cards in hand into different networks. Furthermore, we had a great deal of difficulty with the computational power required to train networks, and more importantly get them to converge to impressive rates. This may also be an artifact of our selected training method, Monte Carlo, which resulted in high variance between model training. We also regularly faced divergence with reinforce methods, as our results indicate.

8 Future Work

Our work provides promising direction for future endeavours for agents in card games and general domains which lend themselves to state space abstraction and feature approximation. We envision several areas in which this work can be extended. First, this research was mostly empirically based, however it would be intriguing to see if some mathematical framework for the importance of different subsets of raw feature space can be developed, instead of the heuristic based approach of this work. Additionally, we would be interested in investigating a general purpose analysis across domains to see if there is some common features of state space which are especially important, akin to the a meta study on Atari games by Mnih et al. (2013).

In terms of techniques themselves, several limitations in our experiments can be overcome in future work. In particular, future work can include more computational power for training the networks in our work, as many gains in recent game agents have been made because hundreds of hours of GPU time (Brown & Sandholm, 2019). We believe larger and deeper networks may be the key to more impressive results, as well as training agents with more powerful policy gradient methods, which we provide results indicating these may be a promising direction. Next, our hyperparameter analysis was conducted by manual search over set intervals for parameters, but Bayesian parameter optimization could be promising to improve performance and investigate this environment. Lastly, we believe a standardized algorithm for comparing agents would be beneficial, and we hope future work can establish an optimal upper-bound for agents in this environment. Therefore, we believe an important area of future research to be creating an optimal offline algorithm for the environment of Hearts.

9 Conclusion

In conclusion, our results provide empirical quantification for the power of reduced raw state features and function approximation in the game of hearts. In particular, we evaluate the utility of a variety of linear and nonlinear feature approximations with a restricted state features of cards in hand, finding both perform extremely significantly better than random. Next, we evaluate combinations of the decomposed raw feature state, and find the best performing combination in the nonlinear function approximation to be cards in-hand, and the cards in-play. Using the REINFORCE algorithm, we train two networks, one with the full state features, and another with the restricted features in-hand and in-play and find the restricted features to be a 33.98% improvement on the full state space

[9]One network for each number of cards in hand

approximation. We hope this work encourages future researchers to prune feature states in nonlinear approximation, informs the development of a mathematical framework for the importance of subsets of raw state features, and provides a new direction for artificial supremacy in the game of Hearts.

ACKNOWLEDGMENTS

We would like to acknowledge Dr. Peter Stone and Dr. Scott Niekum for their instruction and guidance this semester, and all of the teaching staff for their hard work and assistance. We would also like to acknowledge the OpenAI-Gym-Hearts project located at `https://github.com/zmcx16/OpenAI-Gym-Hearts` which provided a great deal of the starter code for our project.

REFERENCES

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019. ISSN 0036-8075. doi: 10.1126/science.aay2400. URL https://science.sciencemag.org/content/365/6456/885.

Henry Charlesworth. Application of self-play reinforcement learning to a four-player game of imperfect information. *CoRR*, abs/1808.10442, 2018. URL http://arxiv.org/abs/1808.10442.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, abs/1603.01121, 2016. URL http://arxiv.org/abs/1603.01121.

Mark K Ho, David Abel, Tom Griffiths, and Michael L Littman. The value of abstraction, Jun 2019. URL psyarxiv.com/6fm9a.

Shin Ishii, Hajime Fujita, Masaoki Mitsutake, Tatsuya Yamazaki, Jun Matsuda, and Yoichiro Matsuno. A reinforcement learning scheme for a partially-observable multi-agent game. *Machine Learning*, 59(1):31–54, May 2005. ISSN 1573-0565. doi: 10.1007/s10994-005-0461-8. URL https://doi.org/10.1007/s10994-005-0461-8.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

Nathan R. Sturtevant and Adam M. White. Feature construction for reinforcement learning in hearts. In *Computers and Games*, pp. 122–134, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-75538-8.

Nathan Reed Sturtevant. *Multiplayer Games: Algorithms and Approaches*. PhD thesis, University of California, Los Angeles, 2003. AAI3089030.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999. MIT Press. URL http://dl.acm.org/citation.cfm?id=3009657.3009806.

M. Wagenaar. *Learning to play the Game of Hearts using Reinforcement Learning and a Multi-Layer Perceptron*. PhD thesis, University of Gronigen, 2017.

A Hyperparameter Data

A.1 Linear Hyperparameter Studies

Using simple linear function approximation with the in-hand feature set, $\epsilon = .05, \alpha = 1e-4$, and 10000 training iterations. These models were then evaluated over 25 sets of 1000 games

γ:	0.50	0.80	0.90	0.95	1.00	Rand
Average Wins	221.12	307.08	265.92	294.36	302.56	252.92
Difference from Rand	-31.80	54.16	13.00	41.44	49.64	0.00
Percent Improvement	-12.57%	21.14%	5.14%	16.38%	19.63%	0.00%

Table 5: Statistics for γ study

Next, we performed a study on the α parameter, using $\epsilon = .05, \gamma = 1$, and 10000 training iterations. The models were then evaluated over 25 sets of 1000 games, and the average number of wins was reported

α:	1e-2	5e-3	1e-3	5e-4	1e-4	Rand
Average Wins	242.52	262.04	232.96	321.8	276.44	252.92
Difference from Rand	-10.4	9.12	-19.96	68.88	23.52	0.00
Percent Improvement	-4.11%	3.61%	-7.89%	27.23%	9.30%	0.00%

Table 6: Statistics for α linear study

Lastly, we performed a study on the ϵ parameter, using $\alpha = 5e-4, \gamma = 1$, and 10000 training iterations. The models were then evaluated over 25 sets of 1000 games, and the average number of wins was reported

ϵ:	0.100	0.050	0.025	0.010	0.005	0.001	Rand
Average Wins	283.92	279.56	294.00	300.28	290.28	308.2	252.92
Difference from Rand	31.00	26.64	41.08	47.36	37.36	55.28	0.00
Percent Improvement	12.26%	10.53%	16.24%	18.73%	14.77%	21.86%	0.00%

Table 7: Statistics for ϵ study

From these hyperparameter studies, we found the best performance for linear approximation was a 27.23% improvement over random performance, or 321.8 average wins which we found with the setting $\alpha = 5e-4, \gamma = 1, \epsilon = .05$ under 10000 training iterations

A.2 NonLinear Hyperparameter Studies

We concluded from the previous data that $\epsilon = 0.01$, and $\gamma = 1.00$ would be suitable parameters for neural network approximation, as they are more closely related to the environment and policy. However, for α, we performed another hyperparameter study, using the Adam optimizer on the architecture described in Section 5.1. The only feature we used was in-hand, to provide comparison against the linear model, and we trained the model for 10000 iterations. We then evaluated on 10 sets of 1000 games.

α:	5e-2	1e-2	5e-3	1e-3	5e-4	1e-4	Rand
Average Wins	329.90	338.10	34.31	329.90	312.00	333.70	248.90
Difference from Rand	81.00	89.20	85.40	81.00	63.10	84.80	0.00
Percent Improvement	32.54%	35.84%	34.31%	32.54%	25.35%	34.07%	0.00%

Table 8: Statistics for α nonlinear study

A.3 REINFORCE Hyperparameter Studies

Lastly, we performed a hyperparameter search on the learning rate of the REINFORCE model. For this model, we performed rollout batches of size 16 for numeric stability, and 100 training iterations using the $\epsilon = 0.01$, $\gamma = 1.00$. For learning rates larger than $1e - 4$, we found issues with the numerical stability of REINFORCE updates, perhaps indicating a bug in our code. We then evaluated on 10 sets of 1000 games.

α:	1e-4	3e-5	1e-5	3e-6	1e-6	Rand
Average Wins	566.00	380.70	282.40	265.60	254.00	248.90
Difference from Rand	317.10	131.80	33.50	16.70	5.10	0.00
Percent Improvement	127.40%	52.95%	13.46%	6.71%	2.05%	0.00%

Table 9: Statistics for α reinforce features 1 and 2 study

B Models and Training Code

For reference, the feature numbers are: 1:= in hand, 2:= in play, 3:= played cards 4:= won cards 5:= player scores

The weights found for the linear model, yielding the results reported in section 5.1 are weights = [1.9500023, -0.52020892, -0.51449678, -0.51932654, -0.5175196, -0.5175582, -0.51678714, -0.51846171, -0.51621948, -0.51609279, -0.51953635, -0.52063994, -0.5160526, -0.51355163, -0.51074676, -0.51967487, -0.51935238, -0.51299344, -0.51578788, -0.51983645, -0.51997923, -0.5139634, -0.52190721, -0.52158865, -0.52092037, -0.51746118, -0.50992192, -0.51852087, -0.51983415, -0.52316094, -0.52213304, -0.52272803, -0.5210616, -0.52125263, -0.51673395, -0.5232066, -1.71146617, -1.20460467, -1.37976837, -0.52519561, -0.51171468, -0.52028463, -0.5216344, -0.51832642, -0.52239885, -0.52283123, -0.5212919, -0.52551173, -0.73250848, -1.02462125, -1.36087529, -1.84840281]

C Related Links

The data collected, trained models, and source code is available at `https://github.com/akashkw/robohearts`.

A video explaining this study can be found at `https://youtu.be/n8zhiRtXqHM`