EECS 545 Final Project Report: Learning Finite Empirical Games

Ben Cassell, Elaine Wah, Paolo Bianchi

Fall 2011

1 Motivation

Games are used to model strategic interaction between 2 or more players. When a game is sufficiently complicated, closed-form descriptions of these games may not exist and we must turn to simulation to gather observations of agent interaction. Gathering such observations may be costly, and the number of profiles[1] to observe is exponential in the number of players and strategies. Given a constrained simulation budget, we will be unable to observe all possible profiles. Historically, this issue has been addressed through either examining only small games or using game reduction methods [3, 9] to shrink the effective size of a game. If we restrict ourselves to small games, however, we cannot reasonably model important real-world phenomena. To effectively model the stock market, for example, we would need to simulate a large number of traders to provide the market with realistic levels of liquidity. Creating the full normal-form representation[2] of such a game could require sampling trillions of profiles. The present alternative—game reduction methods—are also not without fault, as they rely on strong assumptions of smoothness in the outcomes, i.e. the utilities that players earn as a function of the profile. For some games, this assumption can be arbitrarily bad. Therefore, to enable the study of very large games, we wish to learn functions that estimate the outcome of play for unobserved profiles based on profiles that we have observed.

2 Related work

In game theory, game representation plays an important role in the definition and solution of a problem. In order to properly define a game, we must specify the strategy sets of all players in the game, and a mapping from each player's choice of strategy to the utility each player receives. A traditional approach has been to analytically specify games, providing a theoretical abstraction of reality that relies on strong, simplifying assumptions. The size

[1] A profile is an assignment of strategies to each player in a game. In rock-paper-scissors, for example, one profile might be represented as (R, P); in this profile the first player plays 'Rock' and the second player plays 'Paper.'

[2] The normal-form representation of a game states the outcome of play for every profile. The outcome of play is a mapping from player to their utility, or "payoff," from playing their part in the profile. For this reason, normal-form representations are often referred to as payoff matrices.

of a game grows exponentially with the number of players and strategies, however, so the problem of specifying the outcome space of even moderately complex games can quickly become intractable. Thus, large games are generally eschewed by theoreticians. The use of empirical game theoretic models [1] has arisen to address the challenge of describing complex interaction, but it faces computational challenges from having to sample an exponential profile space. To date, two approaches have been taken to address the computational burden: reducing large games to small games and estimating large games from a smaller set of observed profiles.

In hierarchical reduction, the profile space is coarsened by restricting the degree of strategic freedom. In Wellman et al. [9]'s representation, agents are grouped into evenly sized clusters, and each are constrained to follow the strategy of the cluster to which it belongs. The potential savings from this approach are considerable, as it involves combinatorially fewer profiles. For example, if we had a symmetric 12 player game, where each player had 8 strategies they could play, and we were contemplating a hierarchical reduction to a 6 player game, the number of profiles would be reduced from $\binom{19}{12} = 50,388$ profiles to $\binom{13}{6} = 1,716$ profiles. Although an agent's payoff depends on the other agents' play, symmetry and the number of players in the game can make payoffs less sensitive to the exact numbers of other agents playing particular strategies, thus making reduction feasible. Another type of reduction has been proposed by Ficici et al. [3], in which agents are grouped based on similarity in their "strategic view" of the game, i.e. agents in the same group have similar payoffs and similar effects on other agents. Ficici et al. showed that this approach may be useful in the case of asymmetric games where one lacks knowledge of the full game.

Rather than reduce the size of a game, some researchers have tried to estimate games using methods from machine learning. These efforts have come in two flavors: agent-focused and analysis-focused. The majority of the work on machine learning in games takes the agent-focused approach, using the reinforcement learning framework as a way for agents to learn a utility-maximizing strategy from repeated play, such as the work of Littman [5]. These approaches can be thought of as specifying adaptive strategies for playing in some uncertain game. Analysis-focused approaches concern themselves with learning underlying game structure, so that traditional game-theoretic analysis can be conducted on the learned game. To our knowledge, there are only two papers that address directly learning the latent game structure: Gao and Pfeffer [4] and Vorobeychik et al. [8].

Gao and Pfeffer [4] present a constraint satisfaction problem approach that learns an approximate equilibrium profile from observations of play. In order to increase the robustness to noise in the data, the idea of Nash equilibrium[3] is replaced by the concept of "quantal response equilibrium," in which players do not choose best responses with probability one, as in Nash equilibrium; instead, the probability that they choose to take a particular action is proportional to the expected payoff of taking that action. Our work differs from [4] in that our observations may be arbitrarily far from equilibrium play. Additionally, we wish to learn the full game form, rather than just a particular equilibrium.

The work Vorobeychik et al. [8] is the closest paper in the literature to our project.

[3]The Nash equilibrium is the central solution concept of game theory. A Nash equilibrium is a strategy profile such that no player can benefit by changing their strategy if the other players keep their strategies the same. In other words, each player's strategy is a best response to the strategies chosen by all other players.

They focus on learning payoff functions for symmetric, infinite games[4] where strategies are parameterized by values from the real numbers. Training data is gathered by sampling profiles from a regular grid in the parameter space. The payoff learning process is approached as a standard regression problem, with various model forms (low-degree polynomials, local regression and support vector machines). This paper introduces the notion of regret[5] of an equilibrium profile from the learned game when played in the full game as a quality metric, which we also adopt. Where our work differs is that we propose applying machine learning in the domain of finite games where strategies are categorically different, rather than just those having different parameters. To address this difference, where Vorobeychik et al. [8] take their observations as grid points in the continuous parameter space of their strategies, our training data are observations of a subset of the finite number of profiles induced by our players' finite strategy sets. Our project is thus an extension of the idea of learning games to discrete outcome spaces.

3 Problem statement

3.1 Game theory preliminaries

We restrict our attention to symmetric games. A game is called symmetric if all players in the game have identical strategy sets, and if the utility of playing a strategy does not depend on which players play each strategy in a profile. The normal-form representation of a symmetric game is given by the tuple $\Gamma = [m, S, u(\cdot)]$, where m is the number of players in the game, S is the strategy set of all players in the game, and $u(s, t)$ gives the utility of playing strategy s in the profile t. Given a symmetric game with m players and n strategies, we represent a pure strategy profile of players' choices by the vector $t = (c_{s_1}, \ldots, c_{s_n})$, with c_{s_i} denoting the number of players that play s_i in the profile for $s_i \in S$, and satisfying the constraint that $\sum_{i=1}^{n} c_{s_i} = m$.

A strategy profile t is a pure strategy, symmetric Nash equilibrium of the game $[m, S, u(\cdot)]$ if for every $s_i, s_j \in S$:

$$u(s_i, t) \geq u(s_j, \left[c_{s_1}, \ldots, c_{s_i} - 1, \ldots, c_{s_j} + 1, \ldots, c_{s_n}\right])$$

In other words, a profile is a pure strategy, symmetric Nash equilibrium if no agent can benefit from deviating to any other pure strategy. We may also define mixed strategies and mixed strategy, symmetric Nash equilibria. A mixed strategy σ is a probability distribution over $s \in S$. Define t_σ to be the profile where all players play the mixed strategy σ, and $t_{\sigma,\sigma'}$ to be the profile where one player plays σ', and all other players play σ. We can restrict ourselves to considering only deviations to pure strategies, since a mixed strategy can only be a beneficial deviation if at least one of its pure strategies is a beneficial deviation. Then

[4] A game is labeled "infinite" if it has an infinite profile space. A profile space can be infinite if it has either an infinite set of players, or if the strategy space is infinite, such as when strategies can take parameters from a continuous space.

[5] Regret of a profile is the maximum amount of additional utility that any player can gain by deviating to a different strategy.

a mixed strategy profile t_σ is a mixed strategy, symmetric Nash equilibrium of the game $[m, S, u(\cdot)]$ if for every $s \in S$:

$$u(\sigma, t_\sigma) \geq u(s, t_{\sigma,s})$$

3.2 Problem definition

For a symmetric game $\Gamma = [m, S, u(\cdot)]$, we are given a set of observations (t, v), where each data point gives a pure strategy profile t, i.e. the point in feature space, and the realized payoff vector $v = (u(s_1, t), \ldots, u(s_n, t))$, the response variable. From these observations we wish to learn a function u' to estimate the payoffs for unobserved profiles. Additional features could be obtained through knowledge of how strategies are constructed for a specific game, but we restrict ourselves to a feature set that can easily be derived for any symmetric game. Once we have learned u', we construct $\Gamma' = [m, S, u'(\cdot)]$, which we shall refer to as the "learned game," by using u' to estimate v for all t in Γ that were not in our training set. Following the lead of Vorobeychik et al. [8], we wish to learn the payoff-estimation function u' that minimizes the regret of playing a Nash equilibrium from the learned game in the original full game. More formally, for each $t_\sigma \in \eta(\Gamma')$, where $\eta(\Gamma)$ is the set of all symmetric, mixed-strategy Nash equilibria of Γ, we calculate regret as:

$$r(t_\sigma, \Gamma) = \max_{s \in S} u(s, t_{\sigma,s}) - u(\sigma, t_\sigma)$$

Note that here we use the utility function from the true game when calculating regret. In practice, we may not be able to evaluate the regret in full game, since that may require observing all the utilities associated with all profiles in the full game[6]. As we describe further in the next section, we use games where we have all of the observations necessary to calculate regret in the full game, as this study is a) a proof of concept and b) a comparison of several available methods.

4 Experimental methodology

4.1 Setup

To test the general applicability of learning game structure, we used three families of random games: congestion games, local effect games, and credit network games. These games were chosen because they met the following criteria: they are symmetric, have payoffs that are correlated with strategy counts, have non-trivial solutions[7], and were easy to obtain. We had to abandon our original plan to use GAMUT [6] to generate games, as most games generated by GAMUT failed the second or third criterion, and thus would not be suitable for learning. We thus turned to Bryce Wiedenbeck of Michigan's Strategic Reasoning Group to provide

[6]If $S_\sigma = \{s \in S : Pr_\sigma(s) > 0\} \subset S$, you can calculate regret using the sub-game $[m, S_\sigma, u(\cdot)]$ plus the set of profiles where a single player deviates from their prescribed strategy. In these instances, regret calculation does not require observing all profiles in $[m, S, u(\cdot)]$.

[7]Many common games have easily summarized equilibria sets. For example, in coordination games, every profile where all the players play the same pure strategy is a Nash equilibria. Since these games are so well characterized, there isn't much need for estimation.

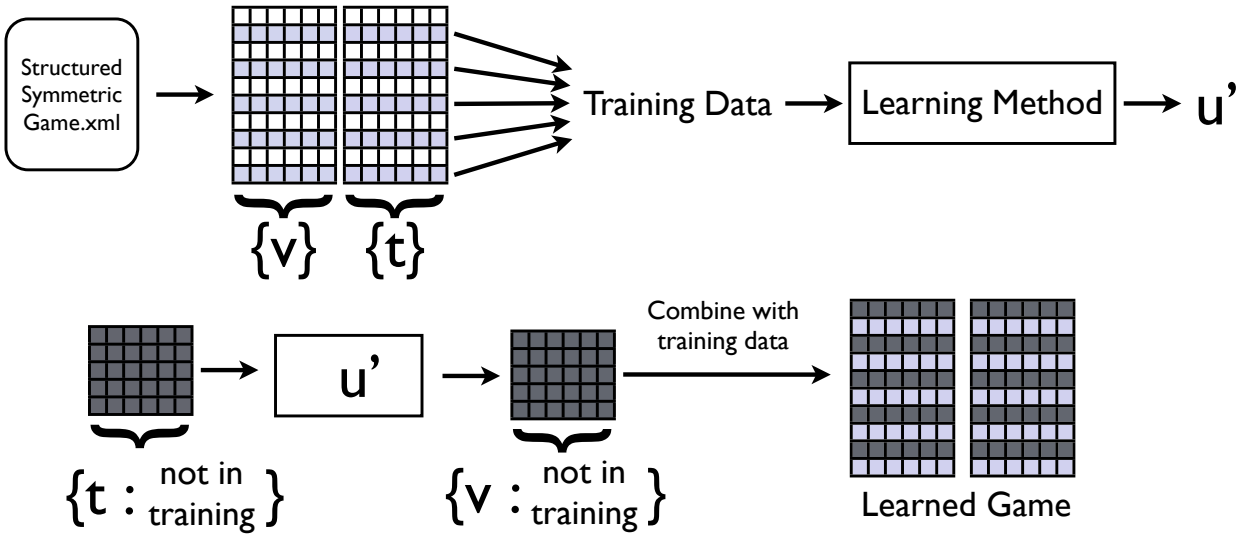


Figure 1: Constructing the "Learned Game".

us with games that met our criteria. Our data falls somewhere between synthetic and real world data, as the games were computer generated but were originally constructed for real use in Bryce's research. For each game family, we were provided with ten 6-player games, each having 462 profiles. For congestion games and local effect games, we were additionally provided with ten 8-player games, also having 6 strategies, each having 1287 profiles.

Figure 1 demonstrates the process we used to construct learned games for each input game and learning method. For each game, we first flatten the game into the input tuples (v, t). From these sets, we randomly select a fixed fraction of these tuples to act as training data. We supply the proposed learning method with the training data to estimate the payoffs, v, for each profile t that was not selected as part of the training set. We then combine the training data with our estimated data to construct the learned game.

Once we have constructed a learned game, we find symmetric, mixed-strategy Nash equilibria of the learned game through the application of replicator dynamics[8] [7]. We then calculate the regret of playing these profiles in the true game, which we use as our measure of quality. More specifically, for the 6-player games, where more than 1 equilibrium may be found, we take the maximum regret among equilibria found for a particular game instance, and average these values over all game instances in the game family. Given this quality metric, we can then compare games learned through different learning methods, as well as examine how approximation quality scales with the portion of the game that is available as training data.

[8]Replicator dynamics is a search method that, if it converges, finds symmetric, mixed strategy Nash equilibria; however, it is not guaranteed to converge, nor are there bounds on the number of steps to convergence, so it can be very costly computationally. As such, for the 8-player games we restrict ourselves to searching for one equilibrium.

4.2 Learning methods

We investigated two approaches to the problem of learning general games: 1) Regression methods to fit a model to the payoffs from each strategy, and 2) Local learning methods that rely on training observations closer to the profile to be predicted in order to estimate the payoffs. Initially we had proposed using a greatly expanded feature space along with dimensionality reduction to discern relevant features, but this proved infeasible, as our proposed feature space expanded the game encoding 2000-fold. Instead, we focused on applying several regression and local learning methods.

4.2.1 Regression

In applying regression methods to the game data, the strategy counts for a given profile t are specified for all strategies $s_i \in S$ by the vector $(c_{s_1}, \ldots, c_{s_n})$, which serves as the set of predictor variables. The response variable for a strategy profile t is $u'(s_i, t)$, i.e. the estimated payoff for playing strategy s_i in the given profile. Therefore, our goal in performing regression on the symmetric strategy space was to find a payoff function f_{s_i} for each strategy s_i where

$$u'(s_i, t) = f_{s_i}(c_{s_i}, \ldots, c_{s_n}).$$

We chose three regression methods to apply to our game space: linear least-squares regression (LIN), ridge regression (RIDGE), and robust regression (ROB). The latter two methods include regularization terms in order to handle instances where the matrix of observations is ill-conditioned or singular or when there are outliers in the data. These are desirable properties in a regression analysis model for our dataset, since profiles are not necessarily associated with exactly n payoffs, where n is the total number of strategies. For instance, a profile t in which zero players select strategy s_k does not have a value for payoff $u(s_k, t)$. This is not equivalent to having a payoff of zero, as it is possible that a given strategy chosen by a nonzero number of players has zero or even negative payoff.

4.2.2 Local learning

The second group of methods we applied were local learning methods. These are relevant for learning general games as oftentimes there are structural properties inherent in the strategic interactions between players, and thus payoffs may be largely influenced by local neighborhoods within the game.

The set of 1-step neighbors of strategy profile $t = (c_{s_1}, \ldots, c_{s_i}, \ldots, c_{s_j}, \ldots, c_{s_n})$ is defined as:

$$\{t' = (c_{s_1}, \ldots, c_{s_i} - 1, \ldots, c_{s_j} + 1, \ldots, c_{s_n}) \,:\, c_{s_i} \geq 1, i \neq j\}$$

This constitutes the set of profiles that can be reached by a single player changing their strategy in a symmetric game. The 1-step neighborhood is the most natural in this domain, since evaluating whether or not a profile is a pure strategy Nash equilibria is equivalent to checking that no single player can improve their payout by changing their strategy. Since some profiles do not have $c_{s_i} \geq 1$ for all $s_i \in S$, profiles can have a variable number of 1-step neighbors (e.g. there can be as many as 30 neighbors and as few as 5 neighbors for

a six-player game with six strategies). Given this definition of a "neighbor" in the strategy space, we applied the k-nearest-neighbor (kNN) algorithm and local regression.

The kNN algorithm predicts a value for the object in question based on the values of its k nearest neighbors. We selected neighbors based solely on deviation distance in the strategy counts rather than using any statistical distance metrics, and we computed the estimate for the missing value as both the mean and the median of the 1-step neighborhood's values. Non-parametric methods such as kNN are particularly appropriate for learning general games as they do not make any assumptions about the underlying distribution of the data.

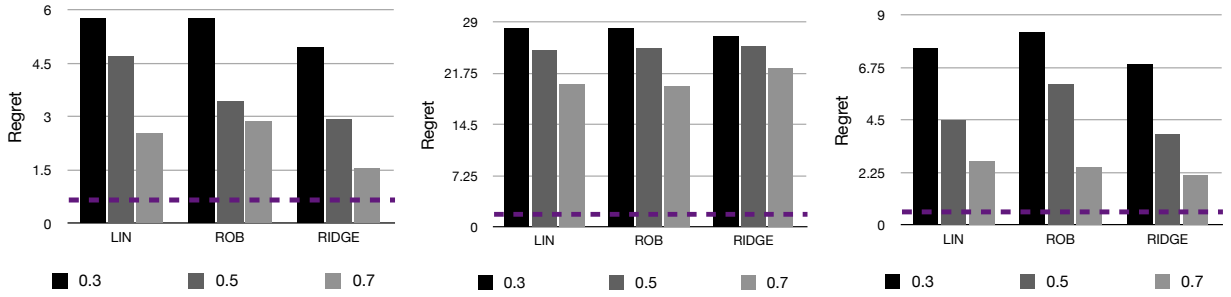
Locally weighted regression (LOESS) [2] fits a model through multivariate smoothing. As LOESS does not require the specification of a global function to fit the payoff data, it allows for greater flexibility in the estimation of unknown profiles. We computed simple regression models on the response variables on localized subsets of the strategy profiles; in our case, each subset is the 1-step neighborhood for a profile, and the neighboring profiles are given equal weight.

5 Results

Figures 2 and 3 summarize our primary findings. We present regret graphs for the regression methods only, as a bug was found in our neighborhood calculations for the local learning methods with too little time remaining to rerun the game-theoretic analysis. The three bars for each method correspond to the fraction of the profiles from the games that we used as training sets, 0.3, 0.5, and 0.7. These may seem to be quite large fractions of the game used for training, but as our profile sampling was random, we needed significantly more profiles to ensure reasonable coverage of the profile space than if we had used an intelligent sampling method. Each graph includes a dotted line signifying the average regret we were able to achieve using hierarchical reduction[9], a prominent alternative in this domain.

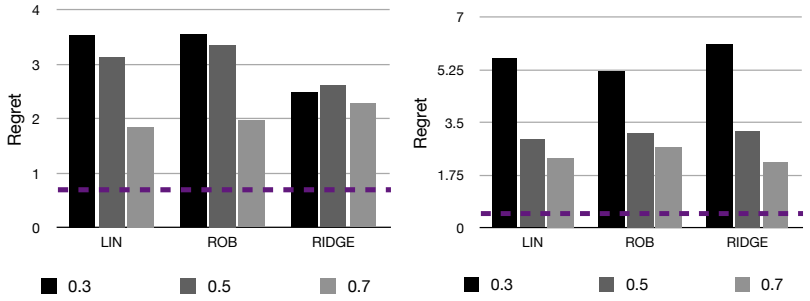
For the 6-player congestion and local-effect games, ridge regression (RIDGE) outperformed robust regression (ROB) and linear regression (LIN), for each level of training data. Although none of our methods achieved the solution quality of the hierarchical reductions, for these two classes of games, ridge regression approached that solution quality, given enough data. The same cannot be said, however, for the credit network games. For these games, none of the proposed regression methods stood out as superior, and all did miserably compared to the hierarchical reduction. For these games, our learning methods tended to construct learned games with symmetric, pure strategy Nash equilibria, in other words, profiles where everyone plays the same strategy from our strategy set. When every player adopts the same strategy in the credit network games, as in other market games, there can be a sharp drop in expected payoff, relative to nearby profiles. Consider, for example, a strategy that says, "I only offer trades to people I have traded with before." If everyone plays this strategy, no trades will occur and no one can profit. If, however, one player is willing to offer trades to everyone, trading in the marketplace can route through this player, and all the agents can profit. These scenarios seem particularly difficult for learning, as payoffs could mono-

[9]We reduced the 6-player games to 3-player games and the 8-player games to 4-player games. These reductions use fewer profiles in their equilibrium estimation than our learning methods, but the profile requirements are on the same order of magnitude.



(a) Congestion games (b) Credit network games (c) Local effect games

Figure 2: Comparing regret of the learning methods, 6-player games.



(a) Congestion games (b) Local effect games

Figure 3: Comparing regret of the learning methods, 8-player games.

tonically increase in the number of players adopting a strategy except when everyone plays that strategy. In such cases, we need methods that can generalize across profiles that have similar structure, even if they have different strategies.

For the 8-player games, no learning method was a clear-cut winner; however, when compared to the 6-player games, all methods improved their average regret. There are at least two effects that may contribute this outcome. First, providing the learning methods with the same constant fractions of data for training in the 6-player and 8-player games actually means the learning methods have more data to learn from in the latter case. The methods may be required to estimate more profiles, but those estimations are based on a function learned from more training data. Additionally, as the game size increases, the number of neighbors the average profile increases. This means that when estimating the payoff of a randomly chosen profile, more observations in our training set are potentially relevant.

As we were unable to perform our proposed game-theoretic analysis for all of the methods we considered, we also examined a more conventional metric, mean squared error. In comparing the mean squared errors of the various methods as given in Tables 1 and 2, the best performing method out of the regression techniques was linear regression, which had the lowest error across all types of games, for both six and eight players. Ridge regression with a fixed ridge parameter $k = 0.005$ performed worse than all the other methods. This is in stark contrast to our domain metric, regret, where ridge regression performed the best on 6-player congestion and local effect games.

Game type	Congestion			Credit networks			Local-effect		
Training set	0.3	0.5	0.7	0.3	0.5	0.7	0.3	0.5	0.7
LIN	2.020	2.022	2.073	104.419	103.597	99.713	2.167	2.087	2.106
ROB	2.352	2.377	2.430	120.814	123.381	121.167	2.660	2.540	2.551
RIDGE	5.069	5.044	5.248	247.375	249.723	244.726	5.189	5.287	5.356
kNN-mean	3.206	2.805	2.489	81.466	78.331	70.927	1.610	1.422	1.341
kNN-median	3.270	2.938	2.583	83.010	81.290	74.212	1.673	1.464	1.395
LOESS	65535	7.240	6.118	65535	739.074	247.765	13.050	12.244	8.478

Table 1: Mean squared errors of learning methods on 6-player games test set

Game type	Congestion			Local-effect		
Training set	0.3	0.5	0.7	0.3	0.5	0.7
LIN	2.643	2.598	2.596	3.105	3.191	3.147
ROB	2.852	2.781	2.757	3.456	3.576	3.525
RIDGE	7.533	7.384	7.468	10.677	10.756	10.528
kNN-mean	5.199	4.815	4.436	2.190	1.989	1.851
kNN-median	5.446	5.022	4.650	2.249	2.072	1.931
LOESS	-	53.275	8.475	-	-	-

Table 2: Mean squared errors of learning methods on 8-player games test set

Of the local learning methods, nearest-neighbors with averaging had the best performance overall. Both kNN-mean and kNN-median had lower errors than some of the regression methods for certain games (e.g. credit networks and local-effect games). Local linear regression performed poorly overall, possibly due to suboptimal sampling. In Table 2, the blank cells for LOESS are indicative of insufficient data for multiple profiles in the test set, i.e. the system of linear equations representing known payoffs in many of the 1-step neighborhoods was underdetermined, causing poor performance when applying the matrix inversion for linear least squares regression. For all methods, the error rate decreased with the addition of more profiles to the training set, as would be expected.

One issue with the local learning methods is that depending on which profiles were known, in some cases the profiles to be estimated did not have any known neighbors in the training set. In such cases one could be propagate estimated values by iteratively applying the method until all unknown payoffs can be fully determined.

6 Conclusions

For this project, we investigated a previously unexamined application of machine learning: estimating payoff functions of finite games. We were rather quickly confronted with a problem that may be a reason why this application has received so little attention. For every savings we would have been able to realize by using machine learning in place of simulation, we were laden with time-consuming steps. As we did not wish to focus on a specific sampling policy, we ended up having to sample the majority of our profiles to prevent large, unexplored regions

of the profile space. Given that our original intent was to address the exponential growth of empirical games, it is less than satisfying that we were unable to outperform hierarchical reduction, even with a greater constant fraction of the profile space as the input set. This poor performance could also have resulted from our unimaginative feature space; however, the alternative that we originally proposed took a problem that grows exponentially and made it 2000 times bigger, making it very difficult to hold the data in memory, let alone perform dimensionality reduction on it.

Beyond all of this, however, is a more central problem: The obvious way to evaluate our learning methods, the metric proposed in the existing literature, was too computationally expensive. Whereas our learning methods could be applied in minutes, evaluating their solution quality could take days. Given the improvement all of the learning methods had when provided with more data, it would be interesting to learn how they scale with this domain. It may be the case that learning methods surpass hierarchical reduction when games are large enough; to evaluate that, though, requires the development of either more efficient game-solving algorithms, or some other reasonable way to approximate solution quality.

Despite these problems, we were able to identify some trends that can inform further development of learning methods for game analysis. We were able to characterize games for which it would be particularly difficult to learn low-regret equilibrium approximations, i.e. games with payoff functions that increase roughly monotonically in the number of players using a particular strategy but have a small number of discontinuities that may be missed during training. By looking at different sizes of games, we observed that providing learning methods with a constant fraction of the strategy space as training data can lead to improved performance as the game size increases. We also proposed two hypotheses to explain this phenomenon. Finally, we evaluated our learning methods under two different quality metrics, and noted that a method's mean squared error rate is not perfectly correlated with the domain-specific metric of regret.

7 Group member contributions

- Ben acquired our data set and handled conversion to and from a Matlab-readable format. He was responsible for composing scripts to find equilibria of our learned games and calculating their regret in the true game. He also conducted the hierarchical reductions that served as benchmarks for solution quality. Ben wrote the majority of the final report, with help from Elaine.

- Elaine worked on developing the regression techniques and the local learning methods for payoff estimation for unobserved strategy profiles, which involved implementation of the methods in MATLAB, sampling the game data and running the experiments on the training sets, and analyzing/comparing the performance of various methods.

- Paolo helped run the regression methods on data for six- and eight-player games and gather results on the performance of the learned functions on estimating payoffs.

References

[1] Olivier Armantier and Jean-François Richard. Empirical game theoretic models: Computational issues. *Computational Economics*, 15:3–24, 2000. ISSN 0927-7099. URL http://dx.doi.org/10.1023/A:1008626508882. 10.1023/A:1008626508882.

[2] William S. Cleveland and Susan J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, pages 596–610, 1988.

[3] Sevan Ficici, David C. Parkes, and Avi J. and Pfeffer. Learning and solving many-player games through a cluster-based representation. *The Proc. 24th Conference in Uncertainty in Artificial Intelligence (UAI'08)*, 2008.

[4] Xi Alice Gao and Avi Pfeffer. Learning game representations from data using rationality constraints. *The Proc. 26th Conference in Uncertainty in Artificial Intelligence (UAI'10)*, 2010.

[5] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann, 1994.

[6] E. Nudelman, J. Wortman, Y. Shoham, and K. Leyton-Brown. Run the gamut: A comprehensive approach to evaluating game-theoretic algorithms. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 880–887. IEEE Computer Society, 2004.

[7] Peter Schuster and Karl Sigmund. Replicator dynamics. *Journal of Theoretical Biology*, 100(3):533 – 538, 1983. ISSN 0022-5193. doi: 10.1016/0022-5193(83)90445-9. URL http://www.sciencedirect.com/science/article/pii/0022519383904459.

[8] Yevgeniy Vorobeychik, Michael Wellman, and Satinder Singh. Learning payoff functions in infinite games. *Machine Learning*, 67:145–168, 2007. ISSN 0885-6125. URL http://dx.doi.org/10.1007/s10994-007-0715-8. 10.1007/s10994-007-0715-8.

[9] Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, Shih-Fen Cheng, and Rahul Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 2*, pages 502–508. AAAI Press, 2005. ISBN 1-57735-236-x. URL http://dl.acm.org/citation.cfm?id=1619410.1619414.