What game are we playing? Differentiably learning games from incomplete observations

Chun Kai Ling
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
chunkail@cs.cmu.edu

J. Zico Kolter
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
zkolter@cs.cmu.edu

Fei Fang
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA 15213
feifang@cmu.edu

Abstract

This paper proposes a learning method for uncertain two-player games. We focus on the under-explored yet important problem of learning game payoffs by observing actions. We present a fully differentiable module capable of learning small zero-sum games purely from observing the actions of individual players. We demonstrate the effectiveness of the learning method on several security game tasks. The proposed method also leads to potential applications in the domains of reinforcement and deep learning.

1 Introduction

This paper proposes a learning method for uncertain two-player games. Although there has been a great deal of work at the intersection of game theory and reinforcement learning [1, 2], most game-theoretic analysis either assumes that the payoffs underlying the game are known (this is the standard game theory setting), or forgoes trying to learn an explicit and complete representation of the game and instead looks for merely learning agent strategies that will perform well [3, 4, 5]. However, in many cases when the true underlying payoffs of the agents are *not* known, our primary goal is precisely to recover or understand the payoffs. For example, in security games, we may want to understand the underlying payoffs of an adversary, rather than just their observed strategy, to better understand how aspects of the game can be manipulated or changed in order to get a desirable outcome.

The few exceptions that focus on learning the payoffs often rely on special structures of the game (e.g., symmetry in multiplayer setting [4]), or querying the best response of the agent with unknown payoffs by asking other agents to play carefully designed strategies [6, 3]. However, the general problem of learning game payoffs by observing actions is still under-explored. In this paper, we propose one method for this challenge and demonstrate its effectiveness on several security game tasks. The crux of our approach is to consider the *quantal response equilibrium* (QRE), a generalization of Nash equilibrium that includes some possibility of agents acting suboptimally. We show that the solution of the QRE is a *differentiable* function of the game payoff matrix, computable via implicit differentiation. We develop a solver that jointly solves the QRE for two-player zero-sum games using a primal-dual Newton Method, and allows us to compute the derivatives of agent actions with respect to the underlying payoff matrix. This enables us to develop end-to-end learning approaches that can

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

infer the payoff matrix underlying a game merely from samples of the agents acting according to their QREs. Naturally, there are some questions here about when games are identifiable or not; in general, the answer is no, because many payoff matrices can lead to identical strategies or policies for the agent, so we do not expect to be able to recover a true underlying payoff matrix. However, we show that for various classes of parametrized games, our approach *is* able to recover the true underlying payoffs of different agents.

More generally, the method allows for (relatively small-scale, normal form) game-solving to be integrated as a module in deep learning systems, a strategy that can find use in multiple application areas. We demonstrate such integration by showing how to learn and predict the agents' behavior when the environment is dynamically changing. The agents may not be playing the same game repeatedly, but instead, the payoffs of the game may be dependent on side information or non-stationary *contextual features*. For example, in wildlife conservation domains, we expect the payoffs to depend on recent weather and animal distribution, which may change over time. In airport patrols, one would reasonably expect that payoffs to an attacker would greatly increase with the density of traffic, which in turn follows a seasonal pattern. Similarly, in a cybersecurity setting, the payoffs encode the probability of an exploit being detected and the benefits of a successful attack, both of which are highly dependent on contextual features such as server and software configurations. When the observed actions are sampled from such dynamic environments, end-to-end learning remains feasible, allowing us to predict both payoff matrix and equilibrium strategies from contextual features.

2 Problem Formulation

2.1 Learning Zero-Sum Normal Form Games

In two-player zero-sum game with payoff matrix P, a classic min-max formulation to compute the Nash equilibrium is as follows

$$\min_{u \in \mathbb{R}^n} \max_{v \in \mathbb{R}^m} \quad u^T P v \tag{1}$$

$$\text{subject to} \quad 1^T u = 1, u \geq 0 \tag{2}$$

$$1^T v = 1, v \geq 0, \tag{3}$$

u and v denote the (mixed) strategies employed by the row (min) and column (max) player respectively. The solution (u^*, v_0) to this optimization problem and the solution (u_0, v^*) of the corresponding problem with inversed player order (i.e., $\min_v \max_u u^T P v$) forms the Nash equilibrium (NE) (u^*, v^*). With a dataset comprising action samples $a^{(i)}$ for one or both players, presumably sampled from (u^*, v^*), we aim to learn the payoff matrix P.

While the normal form representation enjoys a fair amount of expressive power, their applicability in real-world scenarios is limited by the assumption of a fixed P. Two players may not play the same game repeatedly while drawing from the same (u^*, v^*). This serves as motivation to consider a generalized setup similar to supervised learning, where a dynamically changing environment comes into play. A dataset $\{x^{(i)}, a^{(i)}\}$ is observed, where $x^{(i)}$ denotes context and $a^{(i)}$ denotes one or both of the actions taken. It is assumed that the players with observed actions have computed their Nash equilibrium for games decided by the contextual features, and act in accordance with draws from them.

2.2 Quantal Response Equilibria

While extremely powerful both theoretically and as a modeling tool, the NE is poorly-suited for our purposes for the following reasons:

1. NEs are overly strict. In practice, many payoff matrices result in actions being played with zero probability. This tends to be overly restrictive and does not adequately describe real-world scenarios where players are boundedly rational.
2. NEs are not unique. This leads to difficulties when resolving which NE to select.
3. NEs are discontinuous with respect to P – a small change in P can lead to jumps in u^*, v^*. This is troublesome when performing backpropagation.

To address these issues, we model the player's action not with the NE, but the quantal response equilibria [7]. In general, QRE models situations where payoff matrices are injected with some noise. Specifically, we consider the *logit* equilibrium, where payoffs are perturbed by samples from a Gumbel distribution. It is well-known that for zero-sum games, the logit equilibrium obeys the following fixed point [1]

$$u_i^* = \frac{\exp(-Pv)_i}{\sum_{q\in[n]}\exp(-Pv)_q}, \qquad v_j^* = \frac{\exp(P^T u)_j}{\sum_{q\in[m]}\exp(P^T u)_q}. \tag{4}$$

It is further known that for a fixed opponent strategy, the logit equilibrium corresponds to a strategy regularized by the Gibbs entropy[8, 9]. Furthermore, the Gibbs entropy is strictly convex, hence the regularized best response to a fixed opponent strategy is unique.

3 End-to-End Learning

3.1 QRE Solver

We observe that finding the fixed point in (4) is equivalent to solving the regularized min-max game

$$\begin{aligned} \min_{u\in\mathbb{R}^n}\max_{v\in\mathbb{R}^m} \quad & u^T P v - H(v) + H(u) \\ \text{subject to} \quad & 1^T u = 1, \quad 1^T v = 1, \end{aligned} \tag{5}$$

where $H(y)$ is the Gibbs entropy $\sum_i y_i \log y_i$. Notice that the non-negative constraints are implicit from the entropy term, and that the entropy regularization renders the equilibrium continuous with respect to P. Intuitively, entropy regularization encourage players to play more randomly, and no action has probability 0. Furthermore, since the objective is strictly a convex-concave problem, it has a unique saddle point which corresponds to (u^*, v^*).

This formulation leads to a solver that solves the QRE for two-player zero-sum games using a primal-dual Newton Method. To begin, the KKT conditions for the above problem are given by

$$\begin{aligned} Pv + \log(u) + 1 + \mu 1 &= 0 \\ P^T u - \log(v) - 1 + \nu 1 &= 0 \\ 1^T u &= 1 \\ 1^T v &= 1, \end{aligned} \tag{6}$$

where μ, ν are Lagrange multipliers for the equality constraints on u, v respectively. Taking derivatives again yields the following updates for Newton's method, which provides a convergent method for computing the logit response for any 2 player zero-sum game.

$$\begin{bmatrix} \mathrm{diag}(1/u) & P & 1 & 0 \\ P^T & -\mathrm{diag}(1/v) & 0 & 1 \\ 1^T & 0 & 0 & 0 \\ 0 & 1^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta\mu \\ \Delta\nu \end{bmatrix} = - \begin{bmatrix} Pv + \log u + 1 + \mu 1 \\ P^T u - \log v - 1 + \nu 1 \\ 1^T u - 1 \\ 1^v - 1 \end{bmatrix} \tag{7}$$

3.2 Differentiably learning payoff matrices

The QRE solver also allows us to learn P via gradient descent. To do so, we need to obtain $\nabla_P L$ – the gradient of an arbitrary loss function L with respect to P. As with typical neural network arrangements, we only require that the gradients with respect to the computed equilibrium u, v, denoted by $\nabla_u L, \nabla_v L$ be furnished. The derivation of $\nabla_P L$ follows in a manner similar to recent work in [10]. By taking differentials of the KKT conditions and rearranging, we obtain the following equations

$$\begin{bmatrix} \mathrm{diag}(1/u) & P & 1 & 0 \\ P^T & -\mathrm{diag}(1/v) & 0 & 1 \\ 1^T & 0 & 0 & 0 \\ 0 & 1^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathsf{d}u \\ \mathsf{d}v \\ \mathsf{d}\mu \\ \mathsf{d}\nu \end{bmatrix} = \begin{bmatrix} -(\mathsf{d}P)v \\ -(\mathsf{d}P^T)u \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

[1] In the logit response, there is an additional temperature parameter λ. In this work, we fix $\lambda = 1$ throughout.

For small changes denoted by $\mathsf{d}u, \mathsf{d}v$, we have

$$\begin{aligned}
\mathsf{d}L &= \begin{bmatrix}\nabla_u^T L & \nabla_v^T L & 0 & 0\end{bmatrix} \begin{bmatrix}\mathsf{d}u \\ \mathsf{d}v \\ \mathsf{d}\mu \\ \mathsf{d}\nu\end{bmatrix} \\
&= \begin{bmatrix}\nabla_u^T L & \nabla_v^T L & 0 & 0\end{bmatrix} \begin{bmatrix}\text{diag}(1/u) & P & 1 & 0 \\ P^T & -\text{diag}(1/v) & 0 & 1 \\ 1^T & 0 & 0 & 0 \\ 0 & 1^T & 0 & 0\end{bmatrix}^{-1} \begin{bmatrix}-(\mathsf{d}P)v \\ -(\mathsf{d}P^T)u \\ 0 \\ 0\end{bmatrix} \\
&= \begin{bmatrix}v^T \mathsf{d}P^T & u^T \mathsf{d}P & 0 & 0\end{bmatrix} \begin{bmatrix}\text{diag}(1/u) & P & 1 & 0 \\ P^T & -\text{diag}(1/v) & 0 & 1 \\ 1^T & 0 & 0 & 0 \\ 0 & 1^T & 0 & 0\end{bmatrix}^{-1} \begin{bmatrix}-\nabla_u L \\ -\nabla_v L \\ 0 \\ 0\end{bmatrix}.
\end{aligned}$$

The above expression governs how how small changes in $\mathsf{d}P$ affect L. For example, we may obtain the change in L after the perturbation of a single entry in P. Applying this procedure to all entries in P, simplifying and taking limits as $\mathsf{d}P$ is small yields

$$\nabla_P L = y_u v^T + u y_v^T, \tag{9}$$

where

$$\begin{bmatrix}y_u \\ y_v \\ y_\mu \\ y_\nu\end{bmatrix} = \begin{bmatrix}\text{diag}(1/u) & P & 1 & 0 \\ P^T & -\text{diag}(1/v) & 0 & 1 \\ 1^T & 0 & 0 & 0 \\ 0 & 1^T & 0 & 0\end{bmatrix}^{-1} \begin{bmatrix}-\nabla_u L \\ -\nabla_v L \\ 0 \\ 0\end{bmatrix}.$$

Hence, the forward and backward passes with our module are respectively given by: 1) Using the expression in (7), solve for the logit response given P, and 2) Using $\nabla_u L$ and $\nabla_v L$, obtain $\nabla_P L$ using (9). It is stressed that the module is sufficiently general to be included in any existing architecture where the designer believes learning a zero-sum game is appropriate.

3.3 Learning the mapping through backpropagation

We further concatenate the game-solving module that connects the payoff matrix to observed actions with a neural network-based module that maps the contextual features to the payoff matrix. The resulting learning approach can, therefore, learn the mapping, predict the payoff matrix and compute equilibrium strategies under a new set of contextual features. An example architecture utilizing our work is presented in Figure 1. In many settings, P will be parameterized by a domain-dependent low-dimensional vector ϕ, which is dependent on a differentiable function $M_1(x)$. Similarly, the loss function may be taken after applying any differentiable $M_2(u^*, v^*)$.

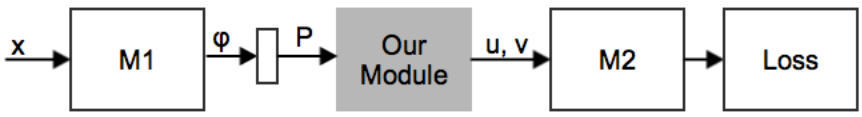


Figure 1: An example architecture utilizing our proposed module (shaded).

3.4 A note on identifiability

As mentioned in Section 1, it is natural to ask if the games are identifiable – that is, is there a unique P which under the logit QRE, generates u^*, v^*? The answer is no, at least in general. Assuming u^*, v^* are fixed, we can rewrite the KKT conditions in (6) as a system of linear equations in P. This system has $\mathcal{O}(nm)$ unknowns but only $\mathcal{O}(n+m)$ constraints. This implies that without a sufficiently compact parametrization, there will be infinitely many payoff matrices leading to identical policies. For example, one can add a constant to all entries in P without changing the QRE. When

	R	P	S
R	0	$-b_1$	b_2
P	b_1	0	$-b_3$
S	$-b_2$	b_3	0

Figure 2: Payoff matrix of modified Rock-Paper-Scissors. Values shown are for the row player.

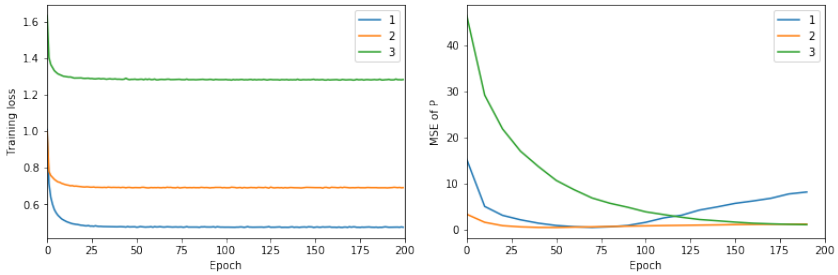


Figure 3: Modified RPS. Left: Training loss. Right: Mean-squared error in predicted payoff matrices.

under-constrained, one cannot expect to recover P. In fact, we present experiments where even with sufficient constraints (i.e., the games are identifiable), payoff matrices which differ greatly (say, in the Frobenius norm) may lead to almost identical policies. This phenomenon of being 'near-unidentifiable' may hinder efforts to recover the true P.

4 Experiments

We present our experiments on a simple variant of the rock-paper-scissors game and more realistic examples of security games. These games have been parameterized such that they are identifiable. Experiments were performed using the *Pytorch* library. Unless otherwise stated, the log-loss is minimized using the RMSProp optimizer [11] with a learning rate of 0.05.

4.1 A Variant of Rock Paper Scissors

Rock Paper Scissors (RPS) is among the most well-studied 2-player zero-sum game. In the standard setting, Rock beats Scissors, Scissors beats Paper, and Paper beats Rock. The winner for each round obtains a payoff of 1 (and the loser -1), while ties reward neither player. It is well known that playing uniformly is an NE and logit QRE for RPS. In this experiment, we consider the following variant (Figure 2), which breaks symmetry between the 3 actions; payoff (penalties) to the winner (loser) depend on their action. Notice that this reduces to the traditional RPS when b_1, b_2, b_3 are all 1.

We assume that each of the b's is a linear function of some features $x \in \mathbb{R}^2$, i.e., $b_y = x^T w_y, y \in \{1, 2, 3\}$, where w_y are to be learned. Features and weights are drawn uniformly from $[0, 1]$, and $[-10, 10]$ respectively. [2] Three such datasets, each with a unique set of weights were generated. For each dataset, 1400 training and 600 validation samples were randomly sampled. The actions of both players, a_1, a_2 is observed for every sample. The model was trained for 200 epochs in mini-batches of size 128. The reconstructed P for each sample is compared to the ground truth, and the MSE monitored every 10 epochs. The results are summarized in Figure 3. By the end of training, each model achieves an MSE of u^* and v^* of less than $10 * -4$.

Except for the first case, good estimates of P were obtained. The failure in case 1 is an example of near-unidentifiability (Section 3.4) – despite predicting u^* and v^* well, the estimates for P are remain poor.

4.2 Resource Allocation Security Games

In this section, we demonstrate the ability to learn from *incomplete observations*. Consider the following zero-sum security game. The attacker (row player) chooses a single attack out of n possible

[2] Note that we do not constrain the sign of the game in any way, i.e., the values of b_i may be negative.

$\{\#D_1, \#D_2\}$	$\{0, 3\}$	$\{1, 2\}$	$\{2, 3\}$	$\{3, 0\}$
T_1	$-R_1$	$-\frac{1}{2}R_1$	$-\frac{1}{4}R_1$	$-\frac{1}{8}R_1$
T_2	$-\frac{1}{8}R_2$	$-\frac{1}{4}R_2$	$-\frac{1}{2}R_2$	$-R_2$

Figure 4: Resource allocation game, $n = 2, k = 3$. The entire budget k is assumed to be utilized.

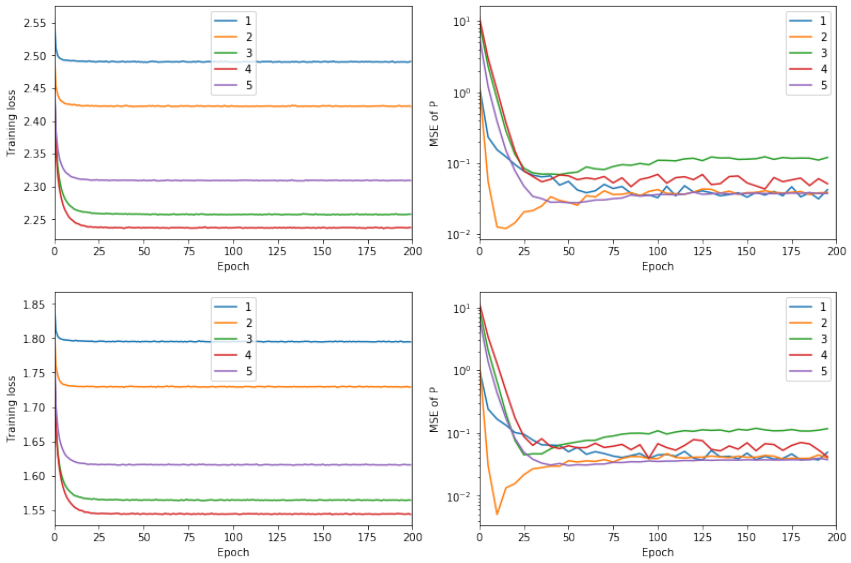


Figure 5: Top left: Log-loss given full observations. Top right: MSE in predicted payoff matrices given full observations (a_1, a_2). Bottom left: Log-loss given partial observations. Bottom right: MSE in payoff matrices given partial observations a_2 only. Note the log-scale in both MSE losses.

choices, $\{T_1, ..., T_n\}$. The defender (column player) may employ defenses; however, each defense $D_i \in \{D_1, ..., D_n\}$ is effective only against a single type of attack T_i. While the defender may purchase multiple defensive resources (potentially of the same type), he is constrained by a budget of k. In the event an attack on T_i succeeds, the attacker obtains a reward of R_i (and the defender $-R_i$). Otherwise, the payoff to both parties is 0. Each defensive resource successfully prevents an intrusion half the time. For example, if there are two defenders, the chance of a successful attack is $\frac{1}{2^2}$. The matrix of expected payoffs when $n = 2, k = 3$ is shown in Figure 4. Due to diminishing returns on each defensive resource, playing this game effectively requires carefully balancing between spreading defensive resources and preventing the most serious attacks. Thus, the equilibrium depends greatly on the parameters of the game, R_1, R_2. As before, R_1, R_2 depend on side features x; for simplicity, we assume they are linear functions of x, with weights drawn from $[0, 10]$. We investigate the case where $n = 2, k = 6, x \in \mathbb{R}^3, x \geq 0$. Each experiment was conducted with 3500 training and 1500 validation samples, and with a mini-batch size of 128.

What makes this experiment interesting is that we assume the viewpoint of a potential attacker, and *only observe defensive strategies* played by the column (maximizing) player. Actions of the attacker are *not* observed; training loss is taken only with respect to the actions of the defender (a_2).

Our results are presented in Figure 5. All cases in both experiments ended with an MSE in u^*, v^* of less than $2 * 10^{-4}$, suggesting that the optimal strategies $u^*(x), v^*(x)$ have been learned well. Furthermore, P is learned up to around 1 decimal place which is reasonable given the scale of the game. Even without observable a_1, we can get good estimates of $u^*(x), v^*(x), P(x)$. This is partly because of the nature of resource allocation games; in most cases, it is a QRE for the attacker to play almost at uniform, suggesting that the loss of information from partial information is limited.

4.3 Compact Security Games

To demonstrate the applicability of our method to real-world settings, we adopt a simplified version of *Compact Security Games* [12], which have been used extensively in modeling security games

	Safe	Attack
Open	0	θ_i
Guard	0	$\phi_i = 0$

	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$
T_1	0	0	θ_1
T_2	0	θ_2	0
T_3	θ_3	0	0

	$\{1\}$	$\{2\}$	$\{3\}$
T_1	0	θ_1	θ_1
T_2	θ_2	0	θ_2
T_3	θ_3	θ_3	0

Figure 6: Security Games. Left: Payoffs for target T_i alone. By assumption (iii), $\phi_i = 0$. Center and right: Payoff in normal form when $n = 3, k = 2$ and $n = 3, k = 1$ respectively.

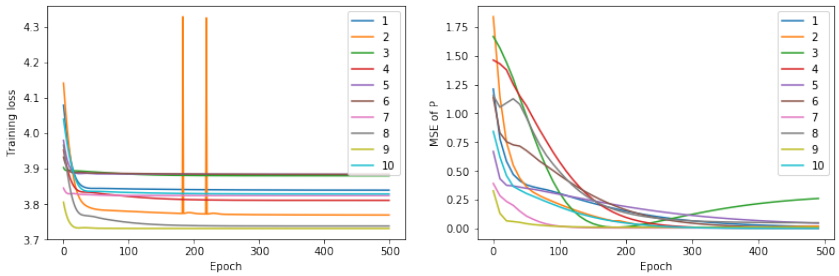


Figure 7: Left: Training loss for DL. Right: Mean-squared error in predicted payoff matrices.

with multiple indistinguishable defensive resources. In this setting, there are n potential targets $\{T_1, ..., T_n\}$ and k defensive resources to be placed. The goal of the attacker is to cause as much harm as possible while avoiding the defender's resources. Only a single target may be attacked. Thus, the game in normal form has n rows and $m = \binom{n}{k}$ columns. To simplify matters, the following assumptions are made: (i) Defensive resources are identical; (ii) Rewards for a successful defense is constant regardless of the target; (iii) Reward for successful defenses is normalized to 0. Under these assumptions, only the payoff for a successful attack needs to be learned. This dramatically reduces the number of parameters to be estimated, while retaining significant expressive power. For instance, in wildlife conservation domains [13], patrollers have identical capabilities, and the penalty for caught poachers is independent of where they are caught. An example of such a game is shown in Figure 6. Compared to the previous experiments, the payoff matrices here are larger and more complex. Hence, our experiments are divided into three portions in increasing complexity.

4.3.1 Direct Learning (DL)

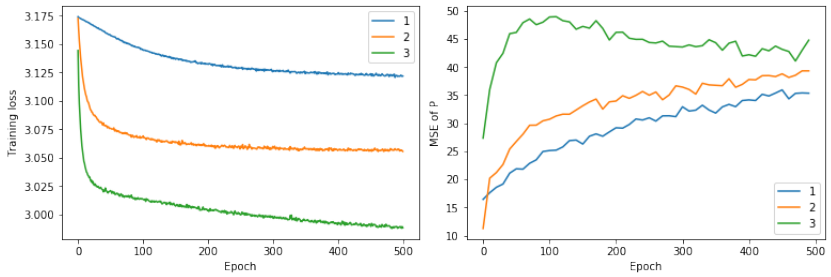
Here, we have directly estimated θ_i by minimizing the log-loss. Setting $n = 5$, $m = 2$, 10 games were randomly generated with rewards uniformly chosen between 0 and 3. For each game, the QRE was computed and 1000 pairs of actions sampled. After training, we compare the mean-squared-error (MSE) of the learned payoffs with the true game. The optimizer was run for 500 epochs in mini-batches of size 32. Results are presented in Figure 7.

4.3.2 Linear Features (LF)

In this scenario, it is assumed that each game j has a *context* described by some features x_j, and that $\theta_{i,j} = x_{i,j}^T w$ for some weights w to be learned. Note that w is shared between targets and x varies between targets. For each of the scenarios, features are uniformly chosen between $[0, 1]$. The learning algorithm is oblivious to this generating process; we use a neural network with 3 hidden layers to approximate g. 3 separate experiments were conducted, each with randomly selected g, x, where $n = 4$ and $k = 2$. Each experiment has 3500 training and 1500 validation samples. The networks were trained for 500 epochs in mini batches of size 32. The results are summarized in Figure 8.

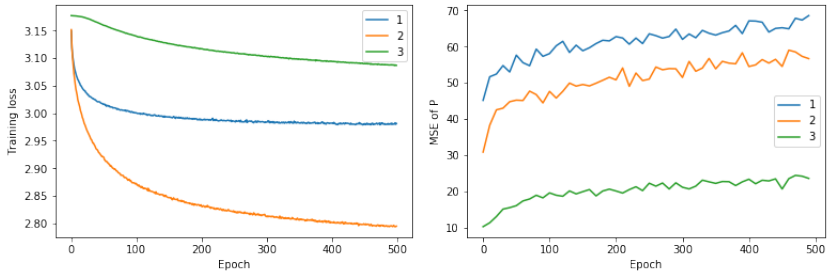
4.3.3 Non-linear Features (NLF)

For this experiment, g is a sparse quadratic in θ, with exactly 1 nonzero quadratic term and 3 nonzero linear terms. Both quadratic and linear terms are generated uniformly between $[-5, 0]$. The rest of the experimental setup is identical to LF. The results are summarized in Figure 9.



#	Val.	Optimal
1	3.117	2.985
2	3.054	2.955
3	2.990	2.895

Figure 8: Left: Training loss for LF. Center: Mean-squared error in predicted payoff matrices. Right: validation loss compared to best-possible loss, using the true u^*, v^*.



#	Val.	Optimal
1	2.993	2.841
2	2.776	2.524
3	3.083	2.957

Figure 9: Left: Training loss for NLF. Center: Mean-squared error in predicted payoff matrices. Right: validation loss compared to best-possible loss, using the true u^*, v^*.

4.3.4 Discussion and Analysis

DL appears to have converged well by the end of training. Furthermore, Figure 7 shows that the true payoff matrix is recovered for all but one the third experiment, where the predicted P moves *away* from the true P after achieving an MSE of 0.0115. It was discovered that the learned payoff matrix yields a *better* log-loss than the ground truth, possibly because sampling error.

For LF and NLF, a reasonable log-loss is achieved by the end of training. Regrettably, the true payoff matrix was not obtained – in fact, the predicted matrix deviates *further* from the true P even as training loss decreases. Again, we postulate that this phenomenon is caused by the model being 'near unidentifiable,' even slight sampling errors lead to large deviations in P.

Nonetheless, these experiments highlight two important points. First, our method works on games which are of modest size. Secondly, we may use non-linear, general purpose function approximators to map features x to parameters θ. This allows practitioners to leverage upon the existing reservoir of work in neural networks and automatic feature extraction.

5 Conclusion

We present a fully differentiable module capable of learning small zero-sum games purely from observing the actions of individual players. Experimental results are promising and point towards potential developments in the future. These include learning of extensive-form and non-zero sum games, as well as rich applications in the domain of reinforcement learning.

Acknowledgments

References

[1] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*, 2008.

[2] Michael Bowling and Manuela Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2000.

[3] Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to.

[4] Yevgeniy Vorobeychik, Michael P Wellman, and Satinder Singh. Learning payoff functions in infinite games.

[5] John Fearnley, Martin Gairing, Paul W Goldberg, and Rahul Savani. Learning equilibria of games via payoff queries. *Journal of Machine Learning Research*, 16:1305–1344, 2015.

[6] Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Learning optimal commitment to overcome insecurity. In *Advances in Neural Information Processing Systems*, pages 1826–1834, 2014.

[7] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

[8] Panayotis Mertikopoulos and William H Sandholm. Learning in games via reinforcement and regularization. *Mathematics of Operations Research*, 41(4):1297–1324, 2016.

[9] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[10] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. *arXiv preprint arXiv:1703.00443*, 2017.

[11] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[12] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 689–696. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[13] Fei Fang, Thanh Hong Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, and Andrew Lemieux. Deploying paws: Field optimization of the protection assistant for wildlife security. 2016.