Large Scale Learning of Agent Rationality in Two-Player Zero-Sum Games

Chun Kai Ling[1], **Fei Fang**[1], **J. Zico Kolter**[1,2]
[1] School of Computer Science, Carnegie Mellon University
[2] Bosch Center for Artificial Intelligence
{chunkail, feif, zkolter}@cs.cmu.edu

Abstract

With the recent advances in solving large, zero-sum extensive form games, there is a growing interest in the inverse problem of inferring underlying game parameters given only access to agent actions. Although a recent work provides a powerful differentiable end-to-end learning frameworks which embed a game solver within a deep-learning framework, allowing unknown game parameters to be learned via backpropagation, this framework faces significant limitations when applied to boundedly rational human agents and large scale problems, leading to poor practicality. In this paper, we address these limitations and propose a framework that is applicable for more practical settings. First, seeking to learn the rationality of human agents in complex two-player zero-sum games, we draw upon well-known ideas in decision theory to obtain a concise and interpretable agent behavior model, and derive solvers and gradients for end-to-end learning. Second, to scale up to large, real-world scenarios, we propose an efficient first-order primal-dual method which exploits the structure of extensive-form games, yielding significantly faster computation for both game solving and gradient computation. When tested on randomly generated games, we report speedups of orders of magnitude over previous approaches. We also demonstrate the effectiveness of our model on both real-world one-player settings and synthetic data.

1 Introduction

Game theory has traditionally been centered around finding players' strategies in equilibrium. In recent years, there has been growing interest in the inverse setting of learning game parameters from observed player actions (Vorobeychik, Wellman, and Singh 2007; Blum, Haghtalab, and Procaccia 2014; Waugh, Ziebart, and Bagnell 2011). Recent work by Ling, Fang, and Kolter (2018) tackle this problem in the zero-sum setting by providing an end-to-end learning framework to learn game parameters such as payoff matrices and chance node probability distributions assuming the actions are sampled from the Quantal Response equilibrium. At the core of the framework is a differentiable game solving module.

However, their proposed method suffers from two major flaws. Firstly, the assumption that players behave in accordance to the QRE severely limits the space of player strategies, and is known to exhibit pathological behavior even in one-player settings. Second, their solvers are computationally inefficient and are unable to scale.

Our work addresses these deficiencies in two ways. First, we propose the Nested Logit Quantal Response equilibrium (NLQRE), which draws upon ideas from from behavioral science and allows for varying levels of player rationality at each stage of the game. We show that the NLQRE is strictly more general than the models considered by Ling, Fang, and Kolter, and may not be replicated by a straightforward scaling of payoff matrices. We derive the required gradients and show that player rationality can be learned via gradient descent can be learned using the same end-to-end learning framework. Second, we substantially reduce training time by reformulating the backward pass as a min-max convex optimization problem and uses state-of-the-art first order primal-dual methods for both the forward pass and backward pass. Unlike previous work, which relied on second-order methods, our first-order solver does not require explicit formation of Hessians and only requires access to a fast best-response oracle. In our evaluation with random payoff matrices and one-card poker, we report orders of magnitude of speedups. Lastly, we evaluate the NLQRE on real-world data in an one-player information gathering game and provide qualitative insights. In total, we believe that our work is a significant step towards the practical learning of human behavior in zero-sum settings.

2 Background and related work

Although much less well studied than traditional equilibrium finding, there are several approaches aimed at the task of learning games in the setting where underlying game payoffs are unknown. These include methods which rely on specific game structure such as symmetry (Vorobeychik, Wellman, and Singh 2007), operate in an active setting (Blum, Haghtalab, and Procaccia 2014), or focus primarily on normal form games and straightforward linear settings (Waugh, Ziebart, and Bagnell 2011). Ling, Fang, and Kolter (2018) provide an alternative framework which embeds a differentiable game solver within another gradient based learner (e.g., a deep network), as illustrated in Figure 1. This enables game parameters to be learned via simple gradient descent. We now describe their framework briefly.

Suppose $P_\Phi(x)$ is the zero-sum payoff matrix given some

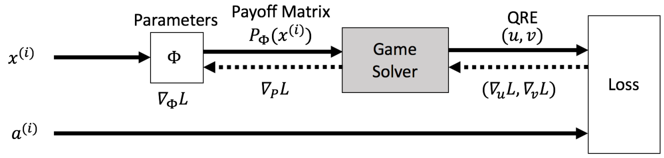

Figure 1: The framework used by Ling, Fang, and Kolter

features x and game parameters Φ which we wish to learn. The game solver takes in $P_\Phi(x)$ and outputs the QRE (u, v), which correspond to mixed strategies of the min and max player. During training, the log loss $L(a, u, v)$ of the solver's predicted strategies is computed against observed actions a. The game parameters Φ are then optimized by minimizing L. This is performed by propagating gradients backwards through the game solver and performing gradient descent, where the required gradients for the backward pass are readily derived by using the implicit function theorem. The training phase is summarized in Algorithm 1. We will now touch on two key ideas from decision and game theory, which will eventually culminate in the proposed NLQRE.

Algorithm 1: Learning game parameters Φ using SGD

```
Input: training data $\{(x^{(i)}, a^{(i)})\}$, learning rate $\eta$, $\Phi_{\text{init}}$
for ep in $\{0, \dots, ep_{max}\}$ do
    Sample $(x^{(i)}, a^{(i)})$ from training data;
    Forward pass: Compute $P_\Phi(x^{(i)})$, QRE $(u, v)$ and
      loss $L(a^{(i)}, u, v)$;
    Backward pass: Compute gradients
      $\nabla_u L, \nabla_v L, \nabla_P L, \nabla_\Phi L$;
    Update parameters: $\Phi \leftarrow \Phi - \eta \nabla_\Phi L$;
end
```

2.1 Nested-logit choice models

One of the fundamental research problems in behavioral science is to mathematically model seemingly irrational (or non-utility maximizing) human behavior. Among the most important models is the class of the random utility models (RUM) (Thurstone 1927). The Logit model[1] is the most notable RUM, where given a set of alternatives $\mathcal{A}$ each with (known) utility U_a, the probability that alternative a is picked is $u^*_a = \frac{\exp(U_a)}{\sum_{a' \in \mathcal{A}} \exp(U_{a'})}$. It is equivalent to the probability that alternative a has highest utility under Gumbel noise, i.e., $u^*_a = \mathbb{P}\left(\arg\max_{a'} (U_{a'} + \epsilon_{a'}) = a\right)$, where $\epsilon_{a'}$ are i.i.d. Gumbel distributed.

However, the logit model suffers from limitations. This includes classic 'red and blue bus' pathologies[2] (Luce 2012) which restrict the class of behaviors permitted. Specifically, logit models obey the property of *independence of irrelevant alternatives*, which does not take into account cases when alternatives are 'qualitatively' similar.

Nested-logit (NL) models (Train 2009) address this limitation by grouping fundamentally similar alternatives together and allows for correlations between ϵ's belonging to the same group. In a two-level NL model, $\mathcal{A}$ is divided into K disjoint clusters, with alternatives a belonging to cluster $k(a)$ chosen with probability

$$u^*_a \propto \exp\left(U_a/\lambda_{k(a)}\right) \left(\sum_{a' \in k(a)} \exp\left(U_{a'}/\lambda_{k(a)}\right) \right)^{\lambda_{k(a)}-1},$$

where λ's are parameters governing noise correlation. These probabilities may be interpreted as a two-stage decision making process: in the first stage, a cluster is chosen, and in the second stage, the specific action is selected based on (scaled) softmax on U_a within the cluster. The probability of choosing each cluster in the first stage is given by the softmax over the (scaled) log-sum-exp of each cluster. When $\lambda = 1$, the standard logit model is recovered, and when $\lambda \to 0^+$, the 'elimination by aspects' is obtained (Tversky 1972). NL models can have multiple layers, leading to a NL tree representing the nested grouping. The reader is directed to the book by Train for background about nested logits and their various interpretations.

2.2 Quantal response equilibria (QRE)

We now turn our attention to 2-player games. Seminal work by McKelvey propose QRE as a noisy alternative to NE. (McKelvey and Palfrey 1995). Similar to logit choice models, the QRE is the equilibrium obtained when payoffs are perturbed by noise obeying a Gumbel distribution. Formally, (u^*, v^*) is a QRE of a normal form game with action sets $\mathcal{A}_{\text{u}}$ and $\mathcal{A}_{\text{v}}$ for the two players and payoff matrix P if

$$u^*_a = \frac{\exp(-Pv^*/\lambda)_a}{\sum\limits_{a' \in \mathcal{A}_{\text{u}}} \exp(-Pv^*/\lambda)_{a'}}, v^*_a = \frac{\exp(P^T u^*/\lambda)_a}{\sum\limits_{a' \in \mathcal{A}_{\text{u}}} \exp(P^T u^*/\lambda)_{a'}},$$

where $1/\lambda$ is a parameter governing the level of agent rationality. Observe that as $\lambda \to \infty$, players behave uniformly at random, while (u^*, v^*) approaches a NE as $\lambda \to 0^+$. For zero-sum games, it is further known (Mertikopoulos and Sandholm 2016) that QRE is the unique solution of the following convex-concave program

$$\min_{u \in \mathbb{R}^n} \max_{v \in \mathbb{R}^m} \quad u^T P v + \lambda \sum_a u_a \log u_a - \lambda \sum_a v_a \log v_a$$
$$\text{subject to} \quad 1^T u = 1, \quad 1^T v = 1.$$

For a two-player extensive form game characterized by a game tree with information sets $\mathcal{I}_{\text{u}}$ and $\mathcal{I}_{\text{v}}$ for the min and max player respectively, Ling, Fang, and Kolter show that when $\lambda = 1$, the QRE in reduced normal form of the game is equivalent to the solution of the following regularized min-max problem, where u and v are the players' strategies in

[1] Logits are more commonly known by the machine learning community as the 'softmax' operator.

[2] Suppose there are 3 alternatives for transport – a red bus, a blue bus, and a car. The player derives the same utility for each alternative, $x_{\text{car}} = x_{\text{red}} = x_{\text{blue}}$. Applying the logit model gives an equal probability of choosing each vehicle. One would however, expect the car to be taken with probability $1/2$ and each bus to be chosen with probability $1/4$, since the color of buses should have no impact on decisions.

sequence form (Von Stengel 1996).

$$\begin{aligned} \min_u \max_v u^T P v + \sum_{h \in \mathcal{I}_u} \sum_{a \in \mathcal{A}_h} u_a \log \frac{u_a}{u_{p_h}} \\ - \sum_{h \in \mathcal{I}_v} \sum_{a \in \mathcal{A}_h} v_a \log \frac{v_a}{v_{p_h}} \\ \text{subject to} \quad Eu = e, \quad Fv = f \end{aligned} \tag{1}$$

In the above, P is the sequence form payoff matrix and E and F are the sequence form constraint matrices. $\mathcal{A}_h$ denotes the possible actions at information set h, while p_h is the action (from the same player) preceding h. In the sequence form, one works with *realization plans* u, v as opposed to probability vectors. These realization plans represent probabilities of choosing a given sequence, while the constraint matrices E, F are matrices containing $\{0, 1, -1\}$ and contain parent-child relationships in the game tree. The sequence form is significantly more compact than the normal form while retaining virtually all of its strategic elements.

3 Nested-logit quantal response equilibria

Our proposed Nested-logit QRE (NLQRE) is a generalization of both the QRE (in zero-sum games) and NL models. That is, it generalizes NL models to two player zero-sum games, or equivalently, extends the QRE by permitting a more general nested logit structure. This allows us to model a far wider range of player behaviors, and in particular, cases where player rationality varies between stages of the game. We assume that the grouping of actions within each information set is known a-priori. The NLQRE is given by the unique solution to the following optimization problem

$$\begin{aligned} \min_u \max_v \qquad & u^T P v + \sum_{h \in \mathcal{I}_{\mathrm{u}}} \lambda_h \sum_{a \in \mathcal{A}_h} u_a \log \frac{u_a}{u_{p_h}} \\ & - \sum_{h \in \mathcal{I}_{\mathrm{v}}} \lambda_h \sum_{a \in \mathcal{A}_h} v_a \log \frac{v_a}{v_{p_h}} \\ \text{subject to} \qquad & Eu = e, \quad Fv = f. \end{aligned} \tag{2}$$

The NL model is recovered in a one-player setting (i.e., Pv is a constant vector) and the QRE is recovered when there is no nesting and λ's are equal. Ling, Fang, and Kolter assumes that all the λ's are equal to 1 and focus on learning entries of the payoff matrix P by exploiting the smoothness of QRE solutions. This allows us to employ gradient-based approaches (Amin, Singh, and Wellman 2016) for learning. In this paper, we do not assume λ's are known in our solution concept and instead treat them as parameters to be learned.

The additional representation power brought by introducing λ's to 1 cannot be achieved by a simple scaling of the payoff matrix in the original formulation by Ling, Fang, and Kolter, even in the non-nested, simultaneous move normal form settings. To see why this is so, consider the game of symmetric rock-paper-scissors with non-uniform rewards (i.e., the payoffs for winners depend on their specific action). Suppose the game is played between 'strong' and 'weak' players, and this is reflected by low and high λ parameters respectively. Due to differing λ's for each player, the strategies of the two players in equilibrium are different. However, scaling P, or even changing individual payoffs for winners (while maintaining symmetry) can only result in symmetric equilibrium.

Remark 1. Readers familiar with nested logits may recall that the most common form of nested logits do not admit chance nodes (or in our 2-player setting, parallel information sets). It may be shown that there is a natural way of doing so by considering representing each alternative as a pure strategy in the *reduced normal form*, and by nesting each action based on the information sets which have a non-zero probability of being reached. The details are presented in the appendix.

Remark 2. The expression in (2) is fairly general. Broadly speaking, our framework allows for 2 types of nesting. First we allow for nesting via information sets (i.e., each information set gets its own λ, see Remark 1), and second, by clustering actions within an information set, which is achieved by introducing intermediate information sets (e.g., the 'red and blue bus' example). Our experiments in Section 5 focus on the former. However (2) and our proposed solver is able to handle the latter case, assuming that the nesting structure is known a-priori.

3.1 NLQRE solver

Following the ideas proposed by Ling, Fang, and Kolter, we present a naive solver for the NLQRE based on Newton's method. Denote $\mathcal{C}_a$ and $\mathcal{C}_{a'}$ as sets of possible information sets immediately following a or a'. Define $J_a = \sum_{h' \in C_a} \lambda_{h'}$, and let ρ_a be the information set immediately preceding the action a, i.e. h where $a \in \mathcal{A}_h$. The KKT conditions for (2) are, for all $h' \in \mathcal{I}_v, a' \in \mathcal{A}_{h'}$, and for all $h \in \mathcal{I}_u, a \in \mathcal{A}_h$,

$$\begin{aligned} (Pv)_a + \lambda_{\rho_a}(1 + \log \frac{u_a}{u_{p_h}}) - J_a + \sum_{c \in \mathcal{C}_a} \mu_c - \mu_h = 0 \\ (P^T u)_{a'} - \lambda_{\rho_{a'}}(1 + \log \frac{v_{a'}}{v_{p_{h'}}}) + J_{a'} + \sum_{c \in \mathcal{C}_{a'}} \nu_c - \nu_{h'} = 0 \\ Eu - e = 0 \qquad Fv - f = 0. \end{aligned} \tag{3}$$

These are necessary and sufficient conditions for NLQRE, implying that the NLQRE can be found by applying Newton's method to (3), yielding the following updates

$$\begin{bmatrix} \Xi(u) & P & E & 0 \\ P^T & -\Xi(v) & 0 & F \\ E^T & 0 & 0 & 0 \\ 0 & F^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta \mu \\ \Delta \nu \end{bmatrix} = -g(u, v, \mu, \nu), \tag{4}$$

$$\text{where} \quad \Xi(u)_{ab} = \begin{cases} \frac{\lambda_{\rho_a} + \sum_{h' \in \mathcal{C}_a} \lambda_{h'}}{u_a}, & a = b \\ -\frac{\lambda_{\rho_a}}{u_b}, & p_{\rho_a} = b \\ -\frac{\lambda_{\rho_b}}{u_a}, & p_{\rho_b} = a \end{cases}, \tag{5}$$

$g(u, v, \mu, \nu)$ contains terms in (3) and $\Xi(v)$ is defined analogously in terms of the appropriate v and λ's. Observe that $\Xi(u)$ and $\Xi(v)$ are diagonally dominant and symmetric, implying that they are positive definite. In the backward pass, we require the gradients of the loss L with respect to P and λ. Similar to prior work (Gould et al. 2016; Amos and Kolter 2017; Ling, Fang, and Kolter 2018), this may be done by applying the implicit function theorem or by simply manipulating differentials. This yields the gradients $\nabla_P L = y_u v^T + u y_v^T$; $\nabla_{\lambda_h} L = \kappa_h^T y_u$ for $h \in \mathcal{I}_u$, $\nabla_{\lambda_h} L = -K_h^T y_v$ for $h \in \mathcal{I}_v$, where

$$(\kappa_h)_a = \begin{cases} 1 + \log(u_a / u_{p_{\rho_a}}), & \rho_a = h \\ -1, & h \in C_a \end{cases} \qquad (6)$$

$$(K_h)_a = \begin{cases} 1 + \log(v_a / v_{p_{\rho_a}}), & \rho_a = h \\ -1, & h \in C_a \end{cases} \qquad (7)$$

$$\begin{bmatrix} y_u \\ y_v \\ y_\mu \\ y_\nu \end{bmatrix} = \begin{bmatrix} -\Xi(u) & P & E & 0 \\ P^T & \Xi(v) & 0 & F \\ E^T & 0 & 0 & 0 \\ 0 & F^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\nabla_u L \\ -\nabla_v L \\ 0 \\ 0 \end{bmatrix}. \qquad (8)$$

4 Fast forward and backward pass solvers

In the framework of Ling, Fang, and Kolter, each gradient step in Algorithm 1 involves solving an optimization problem. Thus, having efficient solvers is crucial in scaling up. In the naive solver, the forward pass is solved using Newton's method and we need to solve the system of linear equations (5) in each iteration. When the game tree is large, solving the system of linear equations in (5) multiple times dramatically slows down training. Similarly for the backward pass, one needs to solve a single linear system shown in (8). When the game is large, naively solving the linear system is also prohibitively slow, even when utilizing sparse matrices. This serves as motivation for a first-order iterative method (FOM) which do not require the solution of a linear system as a subroutine. FOMs are also computationally attractive for solving extensive form games because of the underlying tree structures in games which may be exploited. We will focus on optimization problems in the following min-max form.

$$\min_{Ex=x_0} \max_{Fy=y_0} x^T P y + \mathcal{E}(x) - \mathcal{F}(y), \qquad (9)$$

where $\mathcal{E}(x)$ and $\mathcal{F}(y)$ are strictly convex functions. It is obvious from (2) that the forward pass in our problem solves a problem in this form. We will show later that the backward pass problem shown in (8) can also be seen as solving a problem in this format.

Many methods to solve (9) have been proposed. In this paper, we adapt the method proposed by Chambolle and Pock (2016)[3]. This, as well as many other first order methods apply best response subroutines towards smoothed versions of the min or max original problem taken in isolation. The solution is obtained by alternating between best-responses to minimization and maximization. Algorithm 2 gives the high-level overview of the optimization procedure, where BR are smoothed best responses with appropriately chosen Bregman divergences D_x, D_y , their associated convex functions Ψ_x, Ψ_y, and 'step sizes' τ, α.

$$\mathrm{BR}_x(\bar{x}, \tilde{y}) = \operatorname*{arg\,min}_{Ex=x_0} x^T P \tilde{y} + \mathcal{E}(x) + \frac{1}{\tau} D_x(x, \bar{x})$$

$$\mathrm{BR}_y(\bar{y}, \tilde{x}) = \operatorname*{arg\,min}_{Fy=y_0} -\tilde{x}^T P y + \mathcal{F}(y) + \frac{1}{\sigma} D_y(y, \bar{y}).$$

We first set $\tau = \sigma$ for convenience. For Algorithm 2 to be practical, we will require the best response oracles $\mathrm{BR}_x, \mathrm{BR}_y$ to be computed efficiently. By setting $D_x(x, \bar{x})$ and $D_y(y, \bar{y})$ to be of specific form similar to $\mathcal{E}(x)$ and $\mathcal{F}(y)$ respectively will simplify BR_x and BR_y to be (up to a factor) of the forms

$$\operatorname*{arg\,min}_{Ex=x_0} x^T c_x + \mathcal{E}(x) \quad \text{and} \quad \operatorname*{arg\,min}_{Fy=y_0} y^T c_y + \mathcal{F}(y),$$

which are efficiently computed by exploiting the structure of the problem in extensive form games. This avoids the need to solve a linear system with P as part of the design matrix. The remainder of this section outlines the procedures required for both forward and backward passes. For brevity, we discuss this from the view of the minimization – the maximization subroutine is entirely analogous. Lastly, we remark that the computational advances in this section are independent of the NLQRE, i.e., they remains applicable to the original framework of Ling, Fang, and Kolter.

Algorithm 2: FOM method to solve (9)

```
Input: $x^{(0)}, y^{(0)}, P, \tau, \sigma$
for $i$ in $\{0, \dots\}$ do
    $\tilde{y} = y^{(i)}$;
    $x^{(i+1)} = \mathrm{BR}_x(x^{(i)}, \tilde{y}; P, \tau)$;
    $\tilde{x} = 2x^{(i+1)} - x^{(i)}$;
    $y^{(i+1)} = \mathrm{BR}_y(y^{(i)}, \tilde{x}; P, \sigma)$;
end
```

Forward Pass For this section, the $u = x, v = y$ when referring Algorithm 2. Setting $\mathcal{E}(u), \mathcal{F}(v)$ to be the entropy terms in (2) and $u_0 = e, v_0 = f$ gives the expression in the form of (9). The natural divergence to be chosen is the standard entropy divergence adapted to the dilated setting (dropping terms in D_u which do not contain u).

$$\Psi_u(u) = \mathcal{E}(u) = \sum_{h \in \mathcal{I}_u} \lambda_h \sum_{a \in \mathcal{A}_h} u_a \log \frac{u_a}{u_{p_h}}$$

$$D_u(u, \bar{u}) = \Psi_u(u) - u^T \Psi'_u(\bar{u})$$

$$\Psi'_u(\bar{u}) = \lambda_{\rho_a} - \sum_{h' \in C_a} \lambda_{h'} + \lambda_{\rho_a} \log \frac{\bar{u}_a}{\bar{u}_{p_a}}$$

where a similar expression holds for $D_v(v, \bar{v})$. Plugging into the expression for BR_u gives

$$\mathrm{BR}_u(\bar{u}, \tilde{v}) = \operatorname*{arg\,min}_{Eu=e} \frac{\tau}{1+\tau} u^T \left(P\tilde{v} + \Psi'_u(\bar{u}) \right) + \mathcal{E}(u).$$

It is known that, $\mathrm{BR}_u(\bar{u}, \tilde{v})$ may be solved by a single bottom-up traversal of the game tree and a single sparse

[3] Note that the algorithm by Chambolle and Pock is more general and applies beyond game solving.

matrix-vector multiplication (Hoda et al. 2010). At each information set , we solve for the 'behavioral' best response (i.e., assuming that information set was the root). Each of these sub-problems may be expressed in closed form using log-sum-exp and softmax functions. The sequence form is recovered from behavioral strategies with a single downwards traversal of the tree. The precise details are contained in the appendix.

Backward Pass The backward pass also requires solving a linear system to obtain $[y_u \ y_v \ y_\mu \ y_\nu]$. We first begin by making the crucial observation that the (necessary and sufficient) KKT conditions of the following optimization problem is precisely the linear system in (8).

$$\begin{aligned} \min_x \max_y \quad & x^T P y + \frac{1}{2} x^T \Xi(u) x - \frac{1}{2} y^T \Xi(v) y \\ & + \nabla_u L^T x + \nabla_v L^T y \\ \text{subject to} \quad & Ex = 0 \qquad Fy = 0. \end{aligned} \tag{10}$$

Note that u and v are constants in the backwards pass, here we are optimizing over x, y, which are *not* probabilities. Since $\Xi(u)$ and $\Xi(v)$ are positive definite, this is a convex-concave problem of the form required by Algorithm 2. We select the natural distance generating function $\Psi_x = \frac{1}{2} x^T \Xi(u) x$ which yields (ignoring terms containing only $\bar{x}$),

$$D_x(x, \bar{x}) = \frac{1}{2} x^T \Xi(u) x - x^T \Xi(u) \bar{x}$$

Plugging this into the expression for $\text{BR}_x(\bar{x}, \tilde{y})$ and rearranging gives

$$\arg\min_{Ex=0} \frac{\tau}{1+\tau} x^T (\nabla_u L + P\tilde{y} - \frac{1}{\tau} \Xi(u) \bar{x}) + \frac{1}{2} x^T \Xi(u) x \tag{11}$$

Letting $c = \frac{\tau}{1+\tau} \left(\nabla_u L + P\tilde{y} - \frac{1}{\tau} \Xi(u) \bar{x} \right)$ in (16) gives the KKT conditions

$$c + E^T \gamma + \Xi(u) x = 0, \qquad Ex = 0$$

where γ are Lagrange multipliers. Multiplying by $E\Xi^{-1}(u)$ gives a linear system in γ

$$E\Xi^{-1}(u) c + E\Xi^{-1} E^T \gamma = 0. \tag{12}$$

After solving for γ, one may solve for x

$$x = \Xi^{-1}(u) \left(-c - E^T \gamma \right). \tag{13}$$

Proposition 1. *Solving for γ and x in Equations* (12) *and* (13) *require linear time (in the size of the game tree).*

The derivation involves exploiting the tree-structure inherent in extensive form games. Computational details and proofs are deferred to the appendix.

5 Experiments

The proposed first order method was implemented using Cython. We chose to do so since the best-response subroutines require tree-traversals, which are expensive in Python. while the second order method used the Numpy and Scipy libraries for the solution of linear systems. Where possible, we utilized the Scipy sparse matrix library. This was seen to provide a significant speedup for sparse P for both our method and Newton's method. The PyTorch automatic differentiation library (Paszke et al. 2017) was used to automatically obtain gradients for components outside the game solving module.

5.1 Synthetic datasets

Here we use randomly generated extensive form games to illustrate the computational efficiency of our proposed first order method compared to the second order method used by Ling, Fang, and Kolter. We evaluate the solvers for the forward and backward passes in isolation. The experiments are run over several depths d. Normal form games have $d = 1$. When $d > 1$, we adopt the following extensive form game. players play d distinct simultaneous sub-games in succession, where each simultaneous sub-game has $\hat{n}$ actions. Transitions to the next sub-game is governed by the joint action by both players, i.e., the size of P will be exponential in d. The payoff matrices P were generated with each non-zero entry uniformly chosen from $[-10, 10]$, and rationality parameter λ for *each* information set uniformly and independently chosen between $[0.9, 1.1]$. All timings presented are wall-clock timings. Experiments are run on the cloud with identical Amazon EC2 instances. We set $\tau = 0.1$ for all evaluations.

Evaluation of forward passes In the forward pass, we compared the baseline Newton solver to our proposed first-order method. However, the termination criterion for the 2 methods are non-identical; as Newton's method minimizes the residual rather than duality gap. To strike a fair comparison, we evaluated the 2 methods by first running Newton's method till a residual of less than 10^{-3} is achieved. The duality gap of that solution is computed and subsequently used as the termination criterion for the FOM[4]. The timings and speedup are averaged over 50 trials and presented in Figure 2.

Evaluation of backward passes In the backward pass, the comparison for our proposed FOM is against solving the linear system in (8) directly. In the loss function, we will concern ourselves with the setting where the true matrix P and λ parameters are used in computing u, v for the forward pass. This corresponds to the case the model is already fairly well trained. The results over 50 trials are presented in Figure 3.

It is clear from both figures that our method scales much better than Newton's method for randomly generated matrices. Speedups of more than an order of magnitude are fairly common, and the improvement increases with problem size. Furthermore, it was also observed that our method consumed far less memory than sparse solvers. In fact, solving the sparse system when $d = 3, \hat{n} = 17$ (not plotted) required more than 10GB of memory. On the other hand, our FOM was able to solve such instances in less than a minute,

[4] On occasion, the Newton solver gave a gap extremely close to numerical precision. In these cases, we apply to a termination criterion of 10^{-12}.

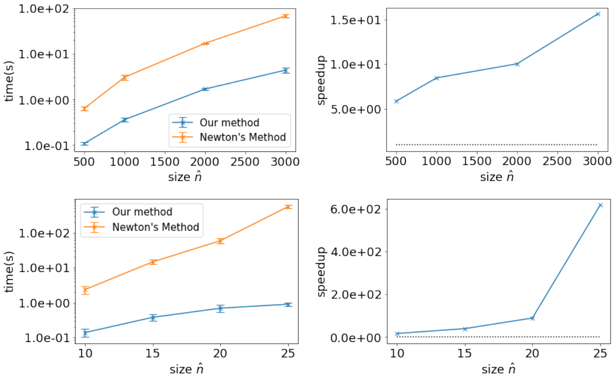

Figure 2: Timings (left) and speedup (right) for forward passes. From top to bottom, $d = 1, 2$. Error bars represent 1 standard deviation. Dotted lines are optimal results given the ground truth.

and with no noticeable increase in memory usage. Note that P contains more than 1.4 million rows and columns in this setting.

5.2 One-Card Poker

Here we evaluate our method on the game of one card poker. This multi-card extension of Kuhn poker contains most interesting strategic elements of game playing (e.g., bluffing) and was used by Ling, Fang, and Kolter to illustrate that it is possible to learn distribution of cards in a deck just by observation player actions. However, the authors worked only on tiny settings with just 4 cards – in sequence form, player strategies may be represented in a 16 dimension vector. Furthermore, the authors assumed that there were no varying input features (i.e., the card distributions were identical for each action observed). These assumptions enabled them to achieve significant speedup by solving the game just once in the forward pass, rather than once for each point in the mini-batch. As we will see, their solver is too slow to be of practical use in larger or featurized settings.

Here, we operate in a slightly different setting. Instead of trying to learn underlying card distributions, we learn player rationality parameters. We assume that player rationality is independent of the cards being drawn, and only depends on the past actions of (both) players. In this setting, there are just 4 parameters to be learned. This is independent of the size of the deck.

We generate our data assuming that player rationality is some linear function of a scalar feature, i.e., there are 4 weights to be learned. The weight vector is drawn uniformly from $[0, 0.01]$. Feature vectors are drawn between $[0, 1]$. Our model is $\lambda_h = w_i \times f + \epsilon_\lambda$. The addition of a small ϵ_λ ensures that the λ's will always remain positive; in our experiments, $\epsilon_\lambda = 0.001$. For each feature, we compute the λ's and find its corresponding equilibrium from which we sample player actions. The training set of size 2000, with an independent test set of size 1000. We minimized the log loss using the Adam optimizer with a batch size of 64 and learning rate of 10^{-4}.

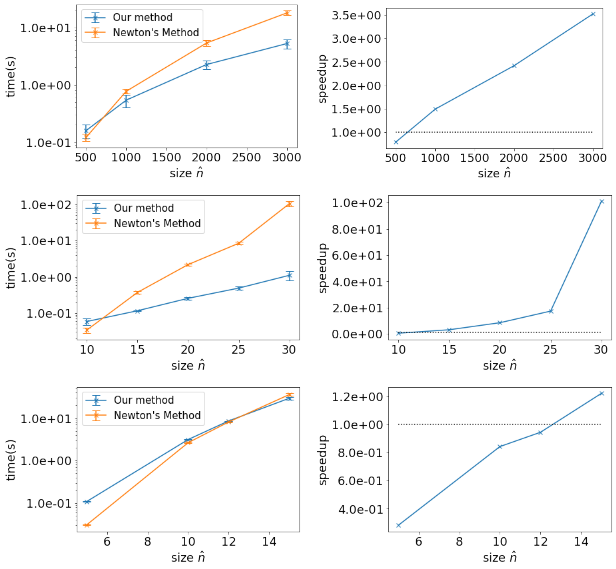

Figure 3: Timings (left) and speedup (right) for backward passes. From top to bottom, $d = 1, 2, 3$. Error bars represent 1 standard deviation. Dotted lines are optimal results given the ground truth.

We compared our solver against Newton's method, which terminates at a residual of 10^{-8}. We fixed $\tau = 1$ for the forward solver and $\tau = 0.1$ for the backward solver. The results are plotted in Figure 4.

In all 3 cases, both the log-loss is close to optimal given around 30 epochs. As expected, our exacts solver exhibits behavior almost identical to that of Newton's method on a per-epoch basis. Our solver is significantly faster than the baseline. It was observed that at almost all stages of training, Newton's method took almost 2 orders of magnitude time in order to learn a model of similar performance. In fact, a single epoch using Newton's method takes as much time as training the entire model using our solver.

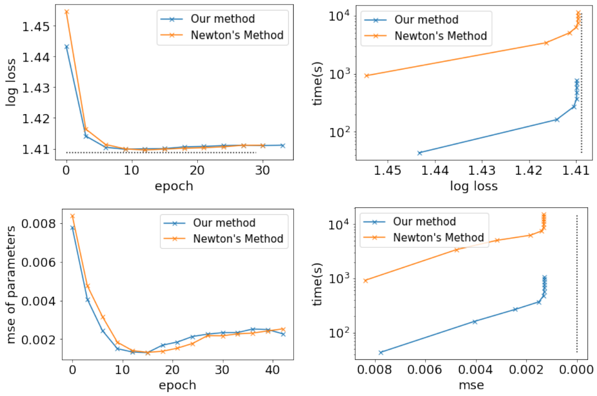

Figure 4: Left: log-loss and mse per epoch, Right: time required to obtain a particular log loss or mse. Dotted lines are optimal results given the ground truth.

	loss	loss(1)	loss(2)	loss(3)	loss(4)
Uniform	1.833	1.099	1.099	1.099	0.693
NoFeat	1.422	0.878	**0.863**	0.826	**0.130**
Feat	**1.419**	**0.874**	0.866	**0.818**	0.145

Figure 5: Log losses for the information gathering dataset. The first column shows losses over the whole game, other columns show losses for individual stages.

5.3 Information gathering dataset

Here we demonstrate the applicability of the nested logit model (i.e., a one player game) using a publically available dataset (Hunt et al. 2016). The game proceeds as follows. Suppose there are 4 faced-down cards ranging from 1-10 placed in a 2×2 matrix (with potential repetitions). The goal of the game is to select the row containing cards with the largest sum. The game proceeds in 4 stages. At each stage of the game, the player may make a guess prematurely, or spend some points in revealing a new card. At the fourth and final stage, the player has to make a guess. The player obtains a reward of 60 and -50 points for correct and incorrect guesses, and may only guess once. The challenge is for the player to judge if it is worth paying to gather more information. Computationally, the optimal policy may be easily obtained using dynamic programming.

However, humans are rarely perfectly rational. We model bounded rationality using the nested logit model. It is assumed that the level of rationality should be a function of a) how many cards are already open, and b) side information such as one's educational qualifications. This leads to a natural description of the game with 4 different λ's, each of which is some function of features, which we describe below.

Two models are trained for this experiment. NoFeat refers to the case when there are no features (i.e., we are simply learning λ) and Feat when we are exploiting demographic information. In this case, features comprise the player's academic qualifications and age, both with one-hot encodings. A player's age is split into 8 age ranges, and education levels follow that of the UK (i.e., GCSE, A levels, Undergraduate, Graduate). Our model employs a neural network with 3 hidden layers of width 100, interspersed by rectified linear activations. To ensure λ's are positive, all inputs were exponentiated before being fed into the solver. Figure 5 shows the log loss over the overall game as well as the loss at each individual stage. For comparison, we also provide the results for a player who picks a random action at every stage of the game. The learned λ parameters for each configuration of features is presented in Figure 6.

From Figure 5, we can see that both trained models significantly outperform Uniform. Log losses at each stage appear to decrease with the stage number. This is unsurprising since players behave more rationally (and hence predictably) as more information is revealed. However, our model appears to perform worse at stage 4, which is in fact a problem with full observability. We suspect this higher loss is a consequence of our model 'overfitting' to be overly confident at the final stage, incurring a huge loss on the rare occasion a player answers incorrectly.

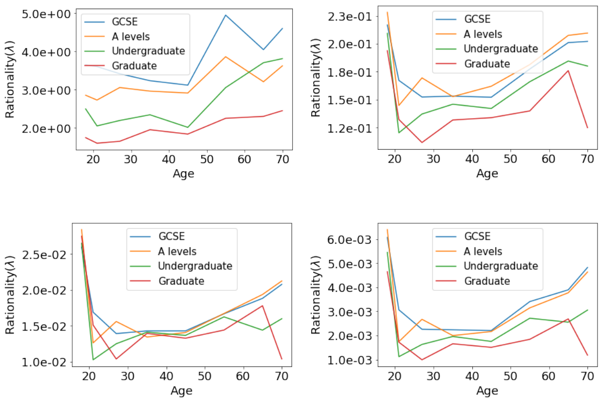

Figure 6: Rationality parameters λ for each stage of the game as a function of age and education. Stages are presented in row-major ordering. Note that ages are binned within some range. All respondents below the age of 18 were grouped together in a single age range.

Several trends are observed from Figure 6. First, notice that λ decrease by approximately a factor of 10 between stages. This is fairly expected, since each information gathering leads to 10 other potential states. Also unsurprisingly, better educated respondents exhibit more rationality (recall a lower λ implies a more rational player). Interestingly, we can see a U-shaped trend in all stages, suggesting that people in the mid twenties and thirties are most rational. Both these observations agree with the findings by (Hunt et al. 2016), where it was shown that higher educated and middle aged respondents obtained the most reward.

6 Conclusion

In this paper, we substantially improve upon existing work in differentiable game learning. We propose the NLQRE which generalizes QRE and NL models. We also derive gradients for backpropagation and learning, and develop solvers which lead to speedups of several orders of magnitude. Future work include the learning of game structure and extensions for general-sum games.

References

Amin, K.; Singh, S.; and Wellman, M. P. 2016. Gradient methods for stackelberg security games. In *Conference on Uncertainty in Artificial Intelligence*, 2–11.

Amos, B., and Kolter, J. Z. 2017. Optnet: Differentiable optimization as a layer in neural networks. *arXiv preprint arXiv:1703.00443*.

Blum, A.; Haghtalab, N.; and Procaccia, A. D. 2014. Learning optimal commitment to overcome insecurity. In *Advances in Neural Information Processing Systems*, 1826–1834.

Chambolle, A., and Pock, T. 2016. On the ergodic convergence rates of a first-order primal–dual algorithm. *Mathematical Programming* 159(1-2):253–287.

Gould, S.; Fernando, B.; Cherian, A.; Anderson, P.; Cruz, R. S.; and Guo, E. 2016. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*.

Hoda, S.; Gilpin, A.; Pena, J.; and Sandholm, T. 2010. Smoothing techniques for computing nash equilibria of sequential games. *Mathematics of Operations Research* 35(2):494–512.

Hunt, L. T.; Rutledge, R. B.; Malalasekera, W. N.; Kennerley, S. W.; and Dolan, R. J. 2016. Approach-induced biases in human information sampling. *PLoS biology* 14(11):e2000638.

Ling, C. K.; Fang, F.; and Kolter, J. Z. 2018. What game are we playing? end-to-end learning in normal and extensive form games. *arXiv preprint arXiv:1805.02777*.

Luce, R. D. 2012. *Individual choice behavior: A theoretical analysis*. Courier Corporation.

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

Mertikopoulos, P., and Sandholm, W. H. 2016. Learning in games via reinforcement and regularization. *Mathematics of Operations Research* 41(4):1297–1324.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Thurstone, L. L. 1927. A law of comparative judgment. *Psychological review* 34(4):273.

Train, K. E. 2009. *Discrete choice methods with simulation*. Cambridge university press.

Tversky, A. 1972. Elimination by aspects: A theory of choice. *Psychological review* 79(4):281.

Von Stengel, B. 1996. Efficient computation of behavior strategies. *Games and Economic Behavior* 14(2):220–246.

Vorobeychik, Y.; Wellman, M. P.; and Singh, S. 2007. Learning payoff functions in infinite games. *Mach Learn* 67:145–168.

Waugh, K.; Ziebart, B. D.; and Bagnell, J. A. 2011. Computational rationalization: the inverse equilibrium problem. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 1169–1176. Omnipress.

Appendix

Best response of the NLQRE as a nested logit

Here we show that the best-response in the NLQRE, which may contain parallel information sets (either due to chance or actions by other players) may be regarded as a nested logit. The idea is to express each strategy in the reduced normal form into a sequence of decisions, each describing what is to be done in each parallel information set.

Consider the following game in Figure 7. Chance (or the other player), labelled as (2), first chooses out of 2 actions, which is made known to the player. For example, this could be the private cards which are dealt to a player in a game of poker. The nodes T_1 and T_2 are the subtrees following these 2 actions by the chance player.

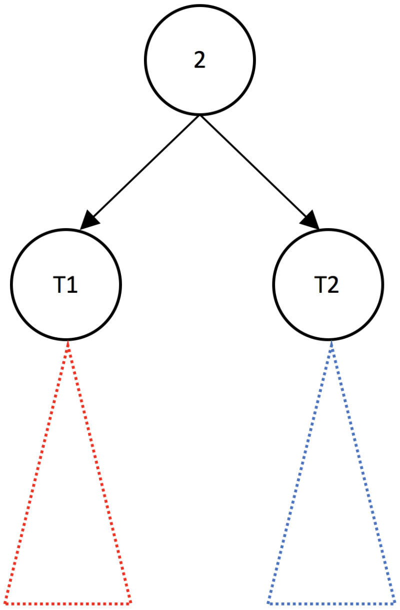

Figure 7: Full 2-player game tree. The first action is performed by chance (or the other player). The player has to provide for contingencies in both the red and blue trees.

Without loss of generality, let T_1' and T_2' be the tree representation of the player's strategies in subgame T_1 and T_2, i.e., T_1' and T_2' are decision trees for the player. Given that chance could have chosen either action to begin with, the pure strategies are the cross product of all strategies between T_1' and T_2', i.e., the player has to account for all possible contingencies. This may be written as a 2-stage decision process in Figure 8, where the first and second stages are choices from T_1' and T_2' respectively, where T_2' is duplicated n_{leaves} times, where n_{leaves} is the number of possible leaves for T_1'.

The rewards are additive in each stage, implying that the best response for each of the duplicated trees are identical. (Note however that the actual payoffs are modulated by the probability of the chance player choosing the left or right action, but this factor is identical for each copy of T_2'). Furthermore, the leaves of T_1' form a probability vector (since T_1' is a decision tree). This implies that setting the rationality parameters for the roots of all copies T_2' to be λ_2 and the rationality parameter at T_1' to be λ_1 yields precisely one-player version of the optimization problem in (2), since the objective in each copy of T_2' is identical and their coefficients sum to 1. Recursively applying this process bottom up to each subtree (i.e., making duplicate copies of subtrees whenever we encounter parallel information sets) gives the desired result.

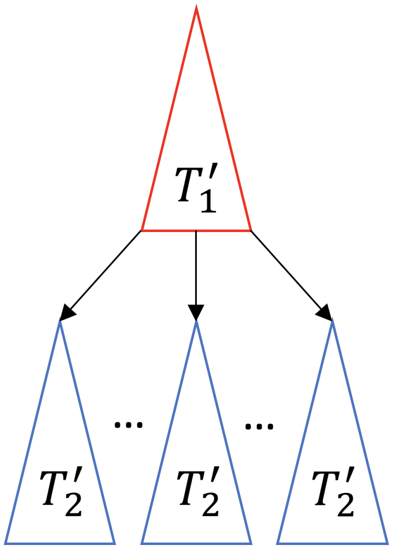

Figure 8: Player's decision tree. Red and blue components are decision trees for the player. Each leaf (or path from the root to a leaf) is a pure strategy in the reduced normal form of the game in Figure 7.

Observe that each pure strategy (path) in Figure 8 is a pure strategy in reduced normal form. However, each path may pass through different information sets (for example, when there is nesting of actions in the bus example), and hence different λ parameters. This is in line with what one would expect with nested logits.

Fast forward and backward pass solvers

In this section, we provide the complete computational details and proofs with regard of how to compute best responses for the forward and backward passes.

Forward Pass For this section, the $u = x, v = y$ when referring to Algorithm 2. Setting $\mathcal{E}(u), \mathcal{F}(v)$ to be the entropy terms in (2) and $u_0 = e, v_0 = f$ gives the expression in the form of (9). The natural divergence to be chosen is the standard entropy divergence adapted to the dilated setting (dropping terms in D_u which do not contain u).

$$\Psi_u(u) = \mathcal{E}(u) = \sum_{h \in \mathcal{I}_u} \lambda_h \sum_{a \in \mathcal{A}_h} u_a \log \frac{u_a}{u_{p_h}}$$

$$D_u(u, \bar{u}) = \Psi_u(u) - u^T \Psi_u'(\bar{u})$$

$$\Psi_u'(\bar{u}) = \lambda_{\rho_a} - \sum_{h' \in C_a} \lambda_{h'} + \lambda_{\rho_a} \log \frac{\bar{u}_a}{\bar{u}_{p_a}}$$

where a similar expression holds for $D_v(v, \bar{v})$. Plugging into the expression for BR_u gives

$$\mathrm{BR}_u(\bar{u}, \tilde{v}) = \arg\min_{Eu=e} \frac{\tau}{1+\tau} u^T \left(P\tilde{v} + \Psi'_u(\bar{u})\right) + \mathcal{E}(u).$$

It is known that, $\mathrm{BR}_u(\bar{u}, \tilde{v})$ may be solved by a single bottom-up traversal of the game tree and a single sparse matrix-vector multiplication (Hoda et al. 2010). At each information set , we solve for the 'behavioral' best response (i.e., assuming that information set was the root). Each of these sub-problems may be conveniently expressed using log-sum-exp and softmax functions. Denoting $c_u = P\tilde{v} + \Psi'_u(\bar{u})$, we compute

$$\min_{u, Eu=e} u^T c_u + \sum_{h \in \mathcal{I}_u} \lambda_h u_{p_h} \sum_{a \in \mathcal{A}_h} \frac{u_a}{u_{p_h}} \log\left(\frac{u_a}{u_{p_h}}\right),$$

where the constraint that $u > 0$ is implicit from the log barrier.

Optimization of the inner summation, along with the relevant part of the inner product may be done in closed form using log-sum-exp. The tree constraints for u allows them to perform traversals bottom up. Throughout the traversal process, denote z_h as the 'value' of each infoset h and r_a as the value of each action.

$$z_h = \lambda_h \log\left(\sum_{\rho_a = i} \exp\left(\frac{r_a}{\lambda_h}\right)\right), \quad r_a = -c_a + \sum_{h \in C(a)} z_h$$

The behavioral strategies $\hat{u}_a$ may be expressed using the softmax function. For an action a belonging to info set h,

$$\hat{u}_a = \frac{\exp(r_a/\lambda_h)}{\sum_{a' \in C_h} \exp(r_{a'}/\lambda_h)} \tag{14}$$

The sequence form may be recovered from behavioral strategies using a single downwards traversal of the tree.

Backward Pass The backward pass also requires solving a linear system to obtain $[y_u \;\; y_v \;\; y_\mu \;\; y_\nu]$. By rewriting the linear system as another min-max problem, we may again apply Algorithm 2. Observe that the solution to the system are precisely the (necessary and sufficient) KKT conditions of the following min-max problem

$$\begin{aligned} \min_x \max_y \quad & x^T P y + \frac{1}{2} x^T \Xi(u) x - \frac{1}{2} y^T \Xi(v) y \\ & + \nabla_u L^T x + \nabla_v L^T y \\ \text{subject to} \quad & Ex = 0 \qquad Fy = 0. \end{aligned} \tag{15}$$

Note that u and v are constants in the backwards pass, here we are optimizing over x, y, which are *not* probabilities. Since $\Xi(u)$ and $\Xi(v)$ are positive definite, this is of the form required by Algorithm 2. We select the natural distance generating function $\Psi_x = \frac{1}{2} x^T \Xi(u) x$ which yields (ignoring terms containing only $\bar{x}$),

$$D_x(x, \bar{x}) = \frac{1}{2} x^T \Xi(u) x - x^T \Xi(u) \bar{x}$$

Plugging this into the expression for $\mathrm{BR}_x(\bar{x}, \tilde{y})$ and rearranging gives

$$\arg\min_{Ex=0} \frac{\tau}{1+\tau} x^T (\nabla_u L + P\tilde{y} - \frac{1}{\tau} \Xi(u) \bar{x}) + \frac{1}{2} x^T \Xi(u) x \tag{16}$$

Proposition 1 *Solving for γ and x in Equations* (12) *and* (13) *require linear time (in the size of the game tree).*

Proof. To obtain efficient solutions for the best responses, we require the following handy results, which may all be verified by algebraic manipulation.

Lemma 2. $\Xi^{-1}(u) E^T = \left[\alpha_1, \alpha_2, \ldots, \alpha_{|\mathcal{I}_u|}\right]$*, where α_i's are column vectors of size equal to number of actions, and contain u_a/λ_h's where a is some descendent of information set i, and 0 otherwise.*

Taking transposes gives the following, $E\Xi^{-1}(u)$ is equal to $\left[\beta_1, \ldots, \beta_{|\mathcal{A}_u|}\right]$, where the β_a's are column vectors of length equal to $|I_u|$, and have entries equal to $\frac{u_a}{\lambda_h}$ if the index i is an ancestor of a.

Lemma 3. $E\Xi^{-1}(u)E^T = diag(\{u_{p_h}\})$*, i.e. equal to a square matrix of size $|I_u|$, with diagonal entries equal to the $\frac{u_{p_h}}{\lambda_h}$ corresponding to a given information state's parent action.*

Lemma 4. *For any vector c, $E\Xi^{-1}(u)c$ may be computed in linear time by traversing the tree bottom-up.*

We are now ready to prove the main theorem. Letting $c = \frac{\tau}{1+\tau}\left(\nabla_u L + P\tilde{y} - \frac{1}{\tau}\Xi(u)\bar{x}\right)$ in (16) gives

$$\mathrm{BR}_x(\bar{x}, \tilde{y}) = \min_{x, Ex=0} x^T c + \frac{1}{2} x^T \Xi(u) x, \tag{17}$$

which has KKT conditions

$$c + E^T \gamma + \Xi(u) x = 0 \tag{18}$$

$$Ex = 0 \tag{19}$$

Multiplying by $E\Xi^{-1}(u)$ gives

$$E\Xi^{-1}(u)c + E\Xi^{-1}E^T\gamma = 0 \tag{20}$$

Note that we should not have introduced new roots in doing so, since these are linear systems and there is a unique solution to γ both before and after the multiplication. Applying Lemma 2 and Lemma 3 gives an expression for γ. Lemma 4, together with the fact that $E\Xi^{-1}E^T$ is diagonal implies that γ may be solved for in linear time. The computation of c requires Py, which may be done in time linear in the size of the *extensive form* game tree. In the extreme case, the game could be a single-stage simultaneous move game, resulting in P being dense. However, for typical EFGs, P should be fairly sparse.

With γ, we may solve for x

$$x = \Xi^{-1}(u)\left(-c - E^T\gamma\right) \tag{21}$$

Since $\Xi(u)$ is tree-structured, the inversion may be done in linear time using Gaussian Elimination. Similarly, $E^T\gamma$ may be computed in linear time because of sparsity in E. That is, the number of non-zero elements in E is equal to the sum of the number of actions over all information sets (recall that each row in E has non-zero entries for the actions for a given information set and its parent). □