Poisoning Attacks on Data-Driven Utility Learning in Games

Ruoxi Jia*, Ioannis C. Konstantakopoulos*, Bo Li, Costas Spanos

Abstract— **Game theory has been employed for modeling agents' decisions in various science and engineering fields. Game theoretic analysis often assumes that the utility function of each agent is known *a priori*, and yet this assumption does not hold for many real-world applications. The combination of Internet of Things (IoT) and advanced data analysis techniques has stimulated fruitful research on learning agents' utility functions from data. Just as many other data-driven methods, utility learning also suffers from potential security risks. Due to the great economic value of accurate forecasting of agents' behaviors, there are huge incentives for adversaries to attack utility learning methods by poisoning training datasets and mislead predictions to achieve malicious goals. In this paper, we introduce and analyze optimal poisoning attack strategies in order to understand adversarial actions and further encourage potential defenses. Moreover, we study how an adversary might disguise the attacks by mimicking normal actions. The proposed attack strategies are evaluated on both synthetic and real-world social energy game data, and the results show that the root mean squared error in predicting agents' actions increases by up to 67% by adding only 5% well-crafted poisoning training instances.**

I. Introduction

One of the central questions in game theory deals with predicting the behavior of an agent. This question has led to the development of a variety of theories and concepts, such as Nash equilibrium, for predicting agents' actions according to their utilities. These predictions may, in turn, be used to inform economic and engineering decisions. Often in game-theoretic analysis, it is assumed that there exists at the best a prior on the model of an individual agent's utility. However, the model primitives are not directly observable and can be difficult to estimate. Small errors in the estimates may substantially affect the predictions of agents' behaviors. Thus, it is crucial to develop accurate estimates of utility models.

Due to pervasive utilization of IoT, massive amounts of data regarding agents' behaviors are being collected. This fosters a wide range of data-driven algorithms for deriving agents' utility models from the observed decisions, including inverse optimization [1], inverse optimal control [2], [3], among others. A recent real-world game in smart buildings [4] demonstrates that data-driven utility learning approaches provide an agile framework to predict agents' behaviors and further facilitate the design of proper incentives that can lead to improvements in energy efficiency and game participants' well-being.

The increasing reliance upon data comes with the security risks. Recent research in machine learning community has shown the vulnerability of data-driven algorithms in various applications [5], [6], [7], [8]. Adding a few well-crafted data points into the training dataset (poisoning attack) or manipulate the testing data (evasion attack) can induce spurious model outputs. These datasets are often crowdsourced or collected from wireless sensor networks, and therefore easy to falsify. High-profile attacks on spam filtering [9], recommendation system [10], face recognition [11] among others have diminished the credibility of data-driven technologies. In parallel to the study of attack and defensive strategies in the machine learning community, this paper presents the first effort to investigate the effectiveness of data-driven models used for game-theoretical analysis under adversarial conditions.

Forecasting agents' behaviors is of paramount importance in various markets, including commodity, energy, and ride-sharing [12] systems among others. For example, predicting volatility is the basis for pricing and better forecast results in better returns. Due to the great economic impact of behavioral forecasts, there are huge incentives for adversaries to generate attacks to mislead prediction systems. In a smart building energy game, an example we will return to throughout the paper, occupants consume shared resources such as lighting, heating, ventilation, and air conditioning (HVAC) through a game where occupants are incentivized to use energy efficiently through monetary rewards. The energy consumption of the building facilities depends on each occupant's subjective tradeoff between comfort and desire to win the rewards. Compromising the learning of occupants' preferences would result in inaccurate energy prediction, which would, in turn, put the building in a disadvantageous position in the energy markets. In this paper, we investigate the vulnerabilities of utility learning models by proposing several poisoning attack strategies. We hope to encourage potential defenses via careful adversarial risk analysis.

The contributions of our paper are listed as follows:

- We present a general framework to estimate utility functions of individual agents in a non-cooperative game from their historical actions.
- We analyze the optimal poisoning attack strategy and develop an algorithm based on Projected Gradient Ascent to solve for the optimal attack.
- We develop an algorithm to synthesize malicious attack points that imitate normal behaviors and are thereby

*Both authors contributed equally.

R. Jia, I. C. Konstantakopoulos, B. Li, C. Spanos are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. {ruoxijia, ioanniskon, crystalboli, spanos}@berkeley.edu

Ioannis C. Konstantakopoulos is a Scholar of the Alexander S. Onassis Public Benefit Foundation.

hard for a defender to detect.
- We demonstrate through experiments on both synthetic and real-world datasets that the proposed poisoning attack strategy outperforms other baseline attack methods.

II. Game-Theoretic Framework

In this section, we describe the agents' decision-making processes in a game-theoretic framework and present the existing method to infer utility functions from the observed equilibrium points using techniques of inverse optimization.

A. Game Formulation

Consider a non-cooperative game with n agents indexed by the set $\mathcal{I} = \{1, \cdots, n\}$. Let agent i's utility function be denoted by $f_i(x_i, x_{-i})$, where x_i abstracts agent i's decision and $x_{-i} = (x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$ is the collective choices of all agents excluding agent i.

Each agent is modeled as a *utility maximizer* that seeks to select $x_i \in \mathcal{C}_i$ by optimizing

$$\max_{x_i \in \mathcal{C}_i} f_i(x_i, x_{-i}), \tag{1}$$

where the constraint set $\mathcal{C}_i$ is given by $\{x_i | h_{i,j}(x_i) \geq 0, j = 1, \cdots, l_i\}$, where $h_{i,j}$ is assumed to be a concave function of x_i.

The game $(f_1, \cdots, f_n)$ is a non-cooperative continuous game on a convex strategy space $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_n$. We use the *Nash equilibrium* concept to model the outcomes of the strategic interactions of agents [13].

Definition 1. A point $x \in \mathcal{C}$ is a Nash equilibrium for the game $(f_1, \cdots, f_n)$ on $\mathcal{C}$ if, for each $i \in \mathcal{I}$,

$$f_i(x_i, x_{-i}) \geq f_i(x_i', x_{-i}) \quad \forall x_i' \in \mathcal{C}_i \tag{2}$$

We say $x \in \mathcal{C}$ is an ϵ-approximate Nash Equilibrium for $\epsilon > 0$ if the above inequality is relaxed as:

$$f_i(x_i, x_{-i}) \geq f_i(x_i', x_{-i}) - \epsilon \quad \forall x_i' \in \mathcal{C}_i \tag{3}$$

Assuming that f_i is concave and $\mathcal{C}$ is convex, the resulting non-cooperative game is a n-person concave game and thus a (pure) Nash equilibrium exists according to [14].

The Lagrangian of agent i's optimization problem is given by

$$L_i(x_i, x_{-i}, \mu_i) = f_i(x_i, x_{-i}) + \textstyle\sum_{j \in \mathcal{A}_i(x_i)} \mu_{i,j} h_{i,j}(x_i) \tag{4}$$

where $\mathcal{A}_i(x_i)$ is the active constraint set at x_i and $\mu = (\mu_1, \ldots, \mu_n)$ with $\mu_i = (\mu_{i,j})_{j=1}^{\ell_i}$ representing the Lagrange multipliers. Assuming appropriate smoothness conditions on each f_i and $h_{i,j}$, the differential game form [15] that characterizes the first-order conditions of the game is given by

$$\omega(x, \mu) = [D_1 L_1(x, \mu_1)^\top \ \cdots \ D_n L_n(x, \mu_n)^\top]^\top \tag{5}$$

where $D_i L_i$ denotes the derivative of L_i with respect to x_i.

B. Utility Learning Framework via Inverse Optimization

The main assumption for the utility learning framework is that the observed decisions of agents are approximate differential Nash equilibria. This framework has been proven to be useful for understanding agents' preferences and behavior patterns in various IoT applications, such as shared resource games [4] and demand response programs [16].

Agent i's utility function f_i is parametrized by $\theta_i = (\theta_{i,1}, \cdots, \theta_{i,m_i}) \in \mathbb{R}^{m_i}$ and a finite set of basis functions $\{\phi_{i,j}\}_{j=1}^{m_i}$ such that

$$f_i(x; \theta_i) = \langle \phi_i(x_i, x_{-i}), \theta_i \rangle + \bar{f}_i(x) \tag{6}$$

where $\phi_i = [\phi_{i,1}, \cdots, \phi_{i,m_i}]^\top$ and $\bar{f}_i(x)$ is a function that captures prior knowledge of agent i's utility function.

Let the k-th observation of the game be $x^{(k)}$. Since it may be the case that only a set of the agents, say $\mathcal{S}^k \subset \mathcal{I}$ at k-th observation, participate in the game. Then notationally, $x^{(k)} = (x_i^{(k)})_{i \in \mathcal{S}^k}$. If agent i participate in p_i instances of the game, then there are p_i observations for that agent. We can consider first-order optimality conditions for each agent's optimization problem and define a residual function capturing the degree of suboptimality of $x_i^{(k)}$. Indeed, for agent i's optimization problem, the residual of the stationary condition can be calculated as

$$r_{s,i}^{(k)}(\theta_i, \mu_i) = D_i f_i(x_i^{(k)}, x_{-i}^{(k)}) + \sum_{j=1}^{l_i} \mu_{i,j} D_i h_{i,j}(x_i^{(k)}) \tag{7}$$

where D_i denotes the derivative of a function with respect to x_i. The residual of the complementary conditions is given by

$$r_{c,i,j}^{(k)}(\mu_i) = \mu_{i,j} h_{i,j}(x_i^{(k)}), j \in \{1, \cdots, l_i\} \tag{8}$$

Define $r_{c,i}^{(k)}(\mu_i) = [r_{c,i,1}^{(k)}(\mu_i), \cdots, r_{c,i,l_i}^{(k)}(\mu_i)]$. Using data from the agents' decisions, the utility learning framework consists of solving the optimization problem for all agents $i = 1, \cdots, n$

$$\min_{\mu_i, \theta_i} \sum_{k=1}^{p_i} \chi_i(r_{s,i}^{(k)}(\theta_i, \mu_i), r_{c,i}^{(k)}(\mu_i)) \tag{9}$$

$$\text{s.t. } \theta_i \in \Theta_i, \mu_i \geq 0 \quad \forall i \in \{1, \cdots, n\} \tag{10}$$

where Θ_i is a constraint set on the parameters θ_i that captures prior information about the objective. Herein, we consider Θ_i an orthant, i.e., $\Theta_i = \{\theta_i | \theta_i \geq d_i\}$, and the values of d_i ensure that the estimated utility functions are concave. χ_i is a non-negative, convex penalty function satisfying $\chi_i(z_1, z_2) = 0$ if and only if $z_1 = 0$ and $z_2 = 0$.

In this paper, we consider the squared loss for utility learning. Define

$$X_i^{(k)} = \begin{bmatrix} D_i h_i(x_i^{(k)}) & D_i \phi_i(x^{(k)}) \\ \hat{h}_i(x_i^{(k)}) & \mathbf{0}_{l_i \times m_i} \end{bmatrix} \tag{11}$$

where

$$\hat{h}_i(x_i^{(k)}) = \mathrm{diag}(h_{i,1}(x_i^{(k)}), \cdots, h_{i,l_i}(x_i^{(k)})) \tag{12}$$

$$D_i h_i(x_i) = [D_i h_{i,1}(x_i), \cdots, D_i h_{i,l_i}(x_i)] \tag{13}$$

Further, let $\beta_i = [\mu_{i,1}, \cdots, \mu_{i,l_i}, \theta_i^\top]^\top$ and define $X_i = [(X_i^{(1)})^\top, \cdots, (X_i^{(p_i)})^\top]^\top$ and

$$Y_i = [-D_i \bar{f}_i(x^{(1)}), \mathbf{0}_{l_i}, \cdots, -D_i \bar{f}_i(x^{(p_i)}), \mathbf{0}_{l_i}] \tag{14}$$

We can convert the utility learning problem into a simple constrained least squares form

$$\min_{\beta_i \in \mathcal{B}_i} \|Y_i - X_i \beta_i\|_2^2 \triangleq \mathcal{L}_{trn}(\mathcal{D}) \tag{15}$$

where $\mathcal{B}_i = \{\beta_i | \mu_i \geq 0, \theta_i \geq d_i\}$ and $\mathcal{D} = (x_1^{(k)}, \cdots, x_n^{(k)})_{k=1}^K$ denotes the training dataset. K is the total number of observations in the training set, and $\mathcal{L}_{trn}(\mathcal{D})$ denotes the training loss. The parameters of agent i's utility function, i.e., θ_i, can thus be estimated by the last m_i coordinates of (15)'s solution. We can then predict the agents' decisions by computing the Nash equilibrium of the estimated utility functions.

III. Poisoning Utility Learning

In this section, we describe the data poisoning attack model considered in this paper and a practical algorithm to solve for optimal attack strategies.

A. The Attack Model

The goal of the attacker is to maximally compromise the utility learning algorithm such that the utility functions learned from the observed decisions of agents are barely useful for predicting agents' future decisions in a game.

We consider a strong threat model based on the Kerckhoffs's principle [17]. An attacker is a player in the game and attempts to achieve the malicious goal by taking well-crafted actions. We assume that the attacker has the following capabilities:

1) The attacker has access to the training dataset which contains the historical decisions of all agents in the game.
2) The attacker can observe other players' action when taking attack actions
3) The attacker knows the utility learning algorithm

Let a denote the index of the attacker player and $\tilde{x}_a$ denote the attack action. Correspondingly, $\tilde{x}_{-a}$ represents the actions of all non-attacker players during the attack time. The optimal attack strategy can thus be formulated as

$$\max_{\tilde{x}_a} \sum_{k=1}^{K} \sum_{i \in \mathcal{S}^k} (x_i^{(k)} - \hat{x}_i^{(k)})^2 \triangleq \mathcal{W} \tag{16}$$

where $x_i^{(k)}$ and $\hat{x}_i^{(k)}$ are the ground truth and the prediction of agent i's k-th decision, respectively. It is worth noting that $\hat{x}_i^{(k)}$ depends on the utility models, which further depend on the attack action $\tilde{x}_a$ because the utility functions are learned by minimizing $\mathcal{L}_{trn}(\mathcal{D} \cup \{\tilde{x}_a, \tilde{x}_{-a}\})$.

B. Optimal Attack Strategies

We use the *projected gradient ascent* (PGA) to solve the optimization problem in (16). In iteration t, we update $\tilde{x}_a^t$ as follows:

$$\tilde{x}_a^{t+1} = \mathrm{Proj}_{\mathcal{C}_a}(\tilde{x}_a^t + s_t \nabla_{\tilde{x}_a} \mathcal{W}) \tag{17}$$

where $\mathrm{Proj}_{\mathcal{C}_a}$ is the projection operator onto the feasible region $\mathcal{C}_a$ and s_t is the step size in iteration t. $\nabla_{\tilde{x}_a} \mathcal{W}$ can be expressed as

$$\nabla_{\tilde{x}_a} \mathcal{W} = -2 \sum_{k=1}^{K} \sum_{i \in \mathcal{S}^k} (x_i^{(k)} - \hat{x}_i^{(k)}) \frac{\partial \hat{x}_i^{(k)}}{\partial \tilde{x}_a} \tag{18}$$

Next, we show how to approximately compute $\frac{\partial \hat{x}_i^{(k)}}{\partial \tilde{x}_a}$. This is challenging because two implicit optimization problems are involved to establish the mapping from $\tilde{x}_a$ to $\hat{x}_i^{(k)}$. We apply the chain rule and get

$$\frac{\partial \hat{x}_i^{(k)}}{\partial \tilde{x}_a} = \sum_{j \in \mathcal{S}^k} \frac{\partial \hat{x}_i^{(k)}}{\partial \theta_j} \frac{\partial \theta_j}{\partial \tilde{x}_a} \tag{19}$$

We refer to $\frac{\partial \hat{x}_i^{(k)}}{\partial \theta_j}$ as the *testing sensitivity*, which characterizes how the estimated Nash equilibria vary according to the parameters of an agent's utility function. This gradient is non-trivial to evaluate since given θ_j, $\hat{x}_i^{(k)}$ is determined by concurrently maximizing the utility function of each agent who participates in the game. $\frac{\partial \theta_j}{\partial \tilde{x}_a}$ is referred to as the *training sensitivity*, which captures the change of the learned utility models with respect to the attack value $\tilde{x}_a$. The least squares optimization in the training phase dictates how the change of $\tilde{x}_a$ propagates to that of θ_j. Inspired by [10], [18], [19], we leverage the KKT conditions to approximately evaluate the training and testing sensitivity.

1) Computing the training sensitivity: The KKT conditions of (15) states that the optimal solution β_j satisfies

$$2X_j^\top (X_j \beta_j - Y_j) + 2\tilde{X}_j^\top (\tilde{X}_j \beta_j - \tilde{Y}_j) - \nu_j = 0 \tag{20}$$

$$\nu_{j,r} \beta_{j,r} = 0, r = 1, \cdots, l_j \tag{21}$$

$$\nu_{j,r}(d_{j,r} - \beta_{j,r}) = 0, r = l_j + 1, \cdots, l_j + m_j \tag{22}$$

where $\tilde{X}_j$ and $\tilde{Y}_j$ are composed in a similar way to (11) and (14) using the observed actions of agent j during the attack time, i.e.,

$$\tilde{X}_j = \begin{bmatrix} D_j h_j(\tilde{x}_j) & D_j \phi_j(\tilde{x}_j, \tilde{x}_{-j}) \\ \hat{h}_j(\tilde{x}_j) & \mathbf{0}_{l_j \times m_j} \end{bmatrix} \tag{23}$$

and

$$\tilde{Y}_j = \begin{bmatrix} -D_j \bar{f}_j(\tilde{x}_j, \tilde{x}_{-j}) \\ \mathbf{0}_{l_j} \end{bmatrix} \tag{24}$$

Note that if agent j is the attacker, then $\tilde{x}_j = \tilde{x}_a$; otherwise, $\tilde{x}_a$ is an element in $\tilde{x}_{-j}$. Among the terms in the KKT conditions, $\tilde{X}_j$, β_j and ν_j are functions of $\tilde{x}_a$, while X_j consists of observed equilibria before the attack time and thereby does not depend on $\tilde{x}_a$.

Assuming the KKT conditions under perturbation of the attack value $\tilde{x}_a$ remain satisfied, we obtain

$$2(X_j^\top X_j + \tilde{X}_j^\top \tilde{X}_j)\frac{\partial \beta_j}{\partial \tilde{x}_a} - \frac{\partial \nu_j}{\partial \tilde{x}_a} = 2\frac{\partial \tilde{X}_j^\top \tilde{Y}_j}{\partial \tilde{x}_a} - 2\frac{\partial \tilde{X}_j^\top \tilde{X}_j}{\partial \tilde{x}_a}\beta_j \tag{25}$$

$$\frac{\partial \nu_{j,r}}{\partial \tilde{x}_a}\beta_{j,r} + \nu_{j,r}\frac{\partial \beta_{j,r}}{\partial \tilde{x}_a} = 0, r = 1, \cdots, l_j \tag{26}$$

$$\frac{\partial \nu_{j,r}}{\partial \tilde{x}_a}(d_{j,r} - \beta_{j,r}) - \nu_{j,r}\frac{\partial \beta_{j,r}}{\partial \tilde{x}_a} = 0, r = l_j + 1, \cdots, l_j + m_j \tag{27}$$

Since $\beta_{j,r}$ and $\nu_{j,r}$ can be computed at each iteration step, we can compute $\frac{\partial \beta_{j,r}}{\partial \tilde{x}_a}$ and $\frac{\partial \nu_{j,r}}{\partial \tilde{x}_a}$ by solving the system of linear equations (25)-(27), and $\frac{\partial \theta_j}{\partial \tilde{x}_a}$ can be obtained from the last m_j coordinates of $\frac{\partial \beta_j}{\partial \tilde{x}_a}$.

2) Computing the testing sensitivity: To compute $\frac{\partial \hat{x}_i^{(k)}}{\partial \theta_j}$, we consider the optimality conditions of each agent's optimization problem at the Nash equilibrium. For agent i, we have

$$\langle D_i\phi_i(\hat{x}_i^{(k)}, \hat{x}_{-i}^{(k)}), \theta_i\rangle + D_i\bar{f}_i(\hat{x}^{(k)}) + \sum_{j\in\mathcal{A}_i(\hat{x}_i^{(k)})} \mu_{i,j} D_i h_{i,j}(\hat{x}_i^{(k)}) = 0 \tag{28}$$

$$h_{i,j}(\hat{x}_i^{(k)}) = 0, \forall j \in \mathcal{A}_i(\hat{x}_i^{(k)}) \tag{29}$$

The equations can be differentiated with respect to $\tilde{x}_a$ using the chain rule and the result of the differentiation is

$$\langle \sum_{h=1}^{n} D_{i,h}\phi_i(\hat{x}_i^{(k)}, \hat{x}_{-i}^{(k)})\frac{\partial \hat{x}_h^{(k)}}{\partial \theta_j}, \theta_i\rangle + \mathbb{1}(j = i)D_i\phi_i(\hat{x}_i^{(k)}, \hat{x}_{-i}^{(k)}) + \sum_{h=1}^{n} D_{i,h}\bar{f}_i(\hat{x}^{(k)})\frac{\partial \hat{x}_h^{(k)}}{\partial \theta_j} + \sum_{j\in\mathcal{A}_i(\hat{x}_i^{(k)})} \frac{\partial \mu_{i,j}}{\partial \theta_j} D_i h_{i,j}(\hat{x}_i^{(k)}) + \sum_{j\in\mathcal{A}_i(\hat{x}_i^{(k)})} \mu_{i,j} D_{i,i} h_{i,j}(\hat{x}_i^{(k)})\frac{\partial \hat{x}_i^{(k)}}{\partial \theta_j} = 0 \tag{30}$$

$$D_i h_{i,j}(\hat{x}_i^{(k)})\frac{\partial \hat{x}_i^{(k)}}{\partial \theta_j} = 0, \forall j \in \mathcal{A}_i(\hat{x}_i^{(k)}) \tag{31}$$

where $\mathbb{1}(\cdot)$ stands for the indicator function and $D_{i,h}$ denotes the second partial derivative with respect to agent i's and h's decision. To solve for $\frac{\partial \hat{x}_i^{(k)}}{\partial \theta_j}$, we can calculate the optimality conditions for all agents, treat them as a system of linear equations in $\frac{\partial \hat{x}_i^{(k)}}{\partial \theta_j}$ $(i = 1, \cdots, n)$, and compute the solution.

Algorithm 1 summarizes the proposed optimal attack strategy. The algorithm can be easily extended to optimizing multiple attack actions by sequentially adjusting one attack action at a time.

C. Mimicking Normal Agent Behaviors

Normally, agents would make decisions that maximize their utility. As a result, malicious actions aimed at compromising utility learning will choose some extreme values that can be easily identified by running a simple statistical test against the normal actions. To mitigate this issue, we propose an alternative approach to computing data poisoning attacks such that the resulting malicious actions mimic the normal ones, but at the same time render the utility learning procedure less effective.

Algorithm 1: Optimizing $\tilde{x}_a$ via PGA

```
Input : Original training dataset $\mathcal{D}_{trn}$, step size
        $\{s_t\}_{t=1}^{\infty}$, non-attacker players' actions during
        the attack time $\tilde{x}_{-a}$
Output: Attack action $\tilde{x}_a$
1 Initialization: Initialize $\tilde{x}_a^0$ with values uniformly
    randomly sampled from $\mathcal{C}_a$; $t = 0$;
2 while $\tilde{x}_a^t$ does not converge do
3     $\{\theta_i\}_{i=1}^n \leftarrow$ learn utility functions on
        $\mathcal{D}_{trn} \cup \{\tilde{x}_a^t, \tilde{x}_{-a}\}$;
4     $\{\hat{x}_1^{(k)}, \cdots, \hat{x}_n^{(k)}\}_{k=1}^K \leftarrow$ predict agents' decisions
        using the learned utility functions;
5     Compute $\nabla_{\tilde{x}_a}\mathcal{W}$ using (18);
6     Update: $\tilde{x}_a^{t+1} = \mathrm{Proj}_{\mathcal{C}_a}(\tilde{x}_a^t + s_t\nabla_{\tilde{x}_a}\mathcal{W})$;
7     $t \leftarrow t + 1$;
8 end
```

We adopt a Bayesian formulation to take into account both data poisoning and detection avoidance objectives. The prior distribution captures the normal decisions and is defined as a Gaussian distribution

$$p_0(\tilde{x}_a) = \mathcal{N}(\tilde{x}_a^*, \sigma_a^2) \tag{32}$$

where $\tilde{x}_a^* = arg\max_{\tilde{x}_a} f_a(\tilde{x}_a, \tilde{x}_{-a})$ represents the optimal action that an agent would take in the game in normal cases. σ_a is the standard deviation for the normal actions. The likelihood $p(\mathcal{D}_{trn}|\tilde{x}_a)$ is defined as

$$p(\mathcal{D}_{trn}|\tilde{x}_a) = \frac{1}{Z}\exp(\gamma\mathcal{W}) \tag{33}$$

where $\mathcal{W}$ is the training loss defined in (16), Z is a normalization constant and $\gamma > 0$ is a parameter that adjusts the tradeoff between attack performance and detection avoidance. Decreasing γ will shift the posterior of $\tilde{x}_a$ towards its prior, which makes the resulting attack strategy less effective but more difficult to detect, and vice versa.

Given the prior and likelihood function, an attack strategy that is mindful of possible detection can be obtained by sampling from the posterior distribution

$$p(\tilde{x}_a|\mathcal{D}_{trn}) = \frac{p(\mathcal{D}_{trn}|\tilde{x}_a)p_0(\tilde{x}_a)}{p(\mathcal{D}_{trn})} \tag{34}$$

$$\propto \exp(-\frac{(\tilde{x}_a - \tilde{x}_a^*)^2}{2\sigma_a^2} + \gamma\mathcal{W}) \tag{35}$$

Sampling from the above posterior distribution is intractable due to the complex dependence of $\mathcal{W}$ on $\tilde{x}_a$. To circumvent this problem, we apply Stochastic Gradient Langevin Dynamics (SGLD) [20] to approximately sample the posterior. More specifically, SGLD iteratively computes a sequence of posterior samples $\{\tilde{x}_a^t\}_{t\geq 0}$, and in iteration t the new sample is calculated by

$$\tilde{x}_a^{t+1} = \tilde{x}_a^t + \frac{s_t}{2}\left(\nabla_{\tilde{x}_a} \log p(\tilde{x}_a|\mathcal{D}_{trn})\right) + \epsilon_t \tag{36}$$

where $\{s_t\}_{t\geq 0}$ are step sizes and $\epsilon_t \sim \mathcal{N}(\mathbf{0}, s_t\mathbf{I})$ is independent Gaussian noise. The gradient $\nabla_{\tilde{x}_a} \log p(\tilde{x}_a|\mathcal{D}_{trn})$ is given by

$$\nabla_{\tilde{x}_a} \log p(\tilde{x}_a|\mathcal{D}_{trn}) = -\frac{(\tilde{x}_a - \tilde{x}_a^*)}{\sigma_a^2} + \gamma \nabla_{\tilde{x}_a}\mathcal{W} \tag{37}$$

where $\nabla_{\tilde{x}_a}\mathcal{W}$ can be computed using (18) and the procedure described in Section III-B.1 and III-B.2. Finally, the sampled malicious action $\tilde{x}_a^T$ is projected onto the feasible set $\mathcal{C}_a$. The pseudo-code of the proposed method is provided in Algorithm 2.

Algorithm 2: Optimizing $\tilde{x}_a$ via SGLD

```
   Input : Original training dataset $\mathcal{D}_{trn}$, step size
           $\{s_t\}_{t=1}^{\infty}$, non-attacker players' actions during
           the attack time $\tilde{x}_{-a}$, tuning parameter $\gamma$, the
           number of SGLD iterations $T$, the parameter
           of the attacker's utility function $\theta_a$
   Output: Attack action $\tilde{x}_a$
9  Initialization: Compute the mean and standard
     deviation of the normal action
     $\tilde{x}_a^* = arg\max_{\tilde{x}_a} f_a(\tilde{x}_a, \tilde{x}_{-a}; \theta_a)$, $\sigma_a$, and sample
     $\tilde{x}_a^0 \sim \mathcal{N}(\tilde{x}_a^*, \sigma_a^2)$;
10 for $t = 0$ to $T$ do
11     $\{\theta_i\}_{i=1}^n \leftarrow$ learn utility functions on
         $\mathcal{D}_{trn} \cup \{\tilde{x}_a^t, \tilde{x}_{-a}\}$;
12     $\{\hat{x}_1^{(k)}, \cdots, \hat{x}_n^{(k)}\}_{k=1}^K \leftarrow$ predict agents' decisions
         using the learned utility functions;
13     Compute $\nabla_{\tilde{x}_a}\mathcal{W}$ using (18);
14     Update $\tilde{x}_a^{t+1}$ according to (36);
15 end
16 Project $\tilde{x}_a^T$ onto $\mathcal{C}_a$
```

IV. Experiments

We evaluate the efficacy of the proposed attack strategies on both synthetic and real-world experimental data collected from a social energy game in a smart building.

A. Social Game in Smart Buildings

We briefly describe our social game experimental setup, and refer the reader to our previous work [21] for a detailed account.

We instrumented the Center for Research in Energy Systems Transformation (CREST) on the University of California, Berkeley campus with a Lutron automated lighting control system. In the game, each occupant can vote for the dim level of the shared lighting resource, and the average vote of all occupants who share the same lighting system will be implemented in the physical space. Occupants win points based on how energy efficient their vote is compared to others. The points are used to determine an occupant's likelihood of winning in a lottery. Only occupants who are present at the office are allowed to vote, and their presence is inferred from the power measurements collected from the meters at their desks [22]. They can actively vote according to their preferences or choose to opt out, in which case their votes are set to the default values. An online platform is designed so that occupants can log their votes, monitor their points, and view all other occupants' consumption patterns and points in real time. This platform also stores past data that is used to estimate occupants' behaviors.

B. Occupant Decision Making

In the social game each agent's utility function is modeled using two basis functions that capture the trade-off between the desire to win and lighting satisfaction:

$$f_i(x_i, x_{-i}) = \theta_i \underbrace{(-\rho c x_i^2)}_{\text{desire to win}} \underbrace{-(\bar{x} - x_i)^2}_{\text{lighting satisfaction}} \tag{38}$$

where $x_i \in [0, 100]$ is agent i's lighting vote. $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ represents the average of all agents' votes and is also the lighting level implemented in the physical space. ρ is the total number of points distributed by the building manager and c is a scaling factor which we set to 10^{-4}. Since the design of the reward system greatly impacts agents' participation, we can inflate the total distributed points to engage agents. This is so-called a "framing" process [23]. The scaling factor c in the "desire to win" term removes the framing effect from the estimation procedure, and its value is selected to ensure that the scales of the two basis functions are similar.

C. Evaluation on Synthetic Data

We first evaluate the attack strategy on a synthetic dataset. Using the parametric utility functions in (38), we devise profiles for eight users with different parameters, and include three default users in order to simulate the real-world game scenario. Default users follow the pre-defined lighting vote without actively adjusting it. Moreover, we randomly select the number of active users in each simulation to simulate the fact that the number of occupants who are present at the office and vote for lighting varies across time in a real game.

We simulate the synthetic data by computing the Nash equilibrium of the agents who are present at the office. The synthetic data is further split into training and testing sets. We consider one of the agents in the game the attacker who aims at hampering the process of utility learning by taking attack actions. Since training data is arranged in chronological order, we assume that the attacks happen at the end of the training period when the attacker can observe the entire history of all agents' actions. To evaluate the impact of attack actions on the utility learning, we examine the predictive power of the utility functions learned from the poisoned training dataset. The training and testing error in terms of root mean square error (RMSE) between the ground truth and predicted actions are used as a measure of the predictive power. Because our goal is to evaluate how well the utility function can predict agents' normal behaviors, the training error is calculated only on the portion of training set without poisoning instances as illustrate in Fig. 1.

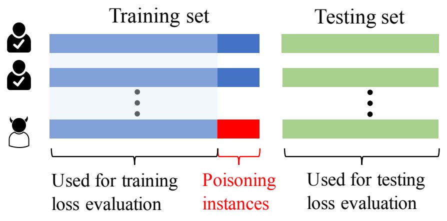
Training set
Testing set
Used for training loss evaluation
Poisoning instances
Used for testing loss evaluation

Fig. 1: Setup of attack effectiveness evaluation.

We compare the PGA strategy with the uniform attack where the poisoning actions are sampled uniformly at random from the allowable lighting vote range $[0, 100]$. We plot the training and testing error against the percentage of poisoning instances for different attack strategies in Fig. 2. The prediction errors without attacks are also shown as a baseline. Fig. 2 indicates that the training and testing error increase as more poisoning instances are added into the training set. It is clear that PGA is more effective than the uniform attack in reducing the predictive power of the learned utility functions.

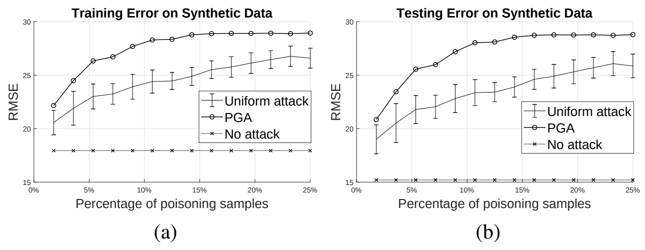

Fig. 2: RMSE for predicting agents' actions using the utility functions learned from the training set with different percentage of poisoning instances.

D. Evaluation on Real-World Data

Next, we compare different attack strategies on the dataset collected in a real-world social game lasting 18 days. The dataset contains 19 users' lighting votes recorded every 6 hours. Since the amount of the data is limited, we utilize the entire dataset for learning utility functions. The efficacy of different attack strategies is evaluated based on the prediction error on the pristine part without adversarial instances. Fig. 3 shows that PGA produces the largest RMSE compared with the other methods. However, PGA only considers the objective of compromising the predictive power of the learned utility functions, and therefore might be detected by an informed defender. As shown in Table I, the paired t-test on the poisoning actions produced by PGA rejects the null hypothesis that the poisoning actions produced by the attack strategies are the same as normal actions. In contrast, SGLD has slightly lower attack efficacy but can generate poisoning actions that are difficult to distinguish from the normal actions. The t-test results for SGLD are also presented in Table I. It can be seen that the null hypothesis cannot be rejected when SGLD is used, and increasing the parameter γ can disguise the poisoning actions better.

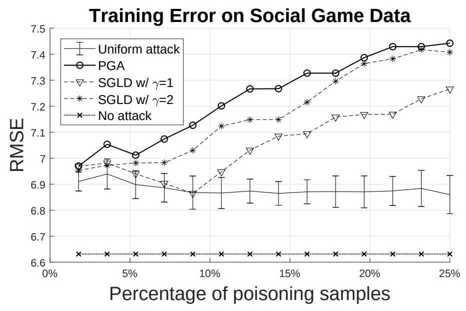

Fig. 3: Comparing the efficacy of different attack strategies on real-world social energy game data.

TABLE I: Paired t-test on the null hypothesis that there is no difference in the mean between the distribution of poisoning attack actions and that of normal actions.

Attack strategy	Test result with 5% significance level	P-value
PGA	Reject the null	0.045
SGLD $\gamma = 1$	Not reject the null	0.423
SGLD $\gamma = 2$	Not reject the null	0.403

V. Conclusions

In this paper, we study the vulnerabilities of data-driven utility learning algorithms in adversarial environments. We first present a general game-theoretic framework to learn the utility function from observations of agents' past actions. We demonstrate how a powerful attacker can generate malicious poisoning attack instances that give rise to large errors in predicting agents' behaviors but at the same time remain indistinguishable from normal actions. Finally, we conduct extensive experiments on both synthetic and real-world game datasets, and show that the proposed PGA attack strategy is more effective in compromising utility learning compared with other baseline methods.

Our ultimate goal for the poisoning attack analysis is to develop possible defensive strategies. From the experimental results, there exists a tradeoff between attack efficacy and detectability. Therefore, dynamically tracking deviations in action patterns to detect malicious actions and applying robust learning methods for utility estimation can be potential defenses.

Acknowledgment

This work is supported in part by the Republic of Singapores National Research Foundation through a grant to the Berkeley Education Alliance for Research in Singapore (BEARS) for the Singapore-Berkeley Building Efficiency and Sustainability in the Tropics (SinBerBEST) Program. This work is also supported in part by Berkeley DeepDrive.

We would like to acknowledge Dawn Song for helpful discussions.

REFERENCES

[1] D. Bertsimas, V. Gupta, and I. C. Paschalidis, "Data-driven estimation in equilibrium using inverse optimization," *Mathematical Programming*, vol. 153, no. 2, pp. 595–633, 2015.

[2] D. Tsai, T. L. Molloy, and T. Perez, "Inverse two-player zero-sum dynamic games," in *Control Conference (AuCC), 2016 Australian*. IEEE, 2016, pp. 192–196.

[3] V. Kuleshov and O. Schrijvers, "Inverse game theory," *Web and Internet Economics*, 2015.

[4] I. C. Konstantakopoulos, L. J. Ratliff, M. Jin, S. S. Sastry, and C. J. Spanos, "A robust utility learning framework via inverse optimization," *IEEE Transactions on Control Systems Technology*, 2017.

[5] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv:1412.6572, 2014.

[7] B. Li and Y. Vorobeychik, "Feature cross-substitution in adversarial classification," in *Advances in Neural Information Processing Systems*, ser. NIPS, 2014, pp. 2087–2095.

[8] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *arXiv preprint arXiv:1707.08945*, 2017.

[9] B. Li and Y. Vorobeychik, "Scalable optimization of randomized operational decisions in adversarial classification settings," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015, pp. 599–607.

[10] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in neural information processing systems*, 2016, pp. 1885–1893.

[11] B. Biggio, L. Didaci, G. Fumera, and F. Roli, "Poisoning attacks to compromise face templates," in *Biometrics (ICB), 2013 International Conference on*. IEEE, 2013, pp. 1–7.

[12] L. J. Ratliff and E. Mazumdar, "Risk-sensitive inverse reinforcement learning via gradient methods," *arXiv preprint arXiv:1703.09842*, 2017.

[13] S. W. Salant, S. Switzer, and R. J. Reynolds, "Losses from horizontal merger: the effects of an exogenous change in industry structure on cournot-nash equilibrium," *The Quarterly Journal of Economics*, vol. 98, no. 2, pp. 185–199, 1983.

[14] J. B. Rosen, "Existence and uniqueness of equilibrium points for concave n-person games," *Econometrica*, vol. 33, no. 3, p. 520, 1965.

[15] L. J. Ratliff, S. A. Burden, and S. S. Sastry, "On the Characterization of Local Nash Equilibria in Continuous Games," *IEEE Trans. on Autom. Control*, vol. 61, no. 8, pp. 2301–2307, 2016.

[16] L. J. Ratliff, R. Dong, H. Ohlsson, and S. S. Sastry, "Incentive design and utility learning via energy disaggregation," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3158–3163, 2014.

[17] C. E. Shannon, "Communication theory of secrecy systems," *Bell Labs Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[18] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *International Conference on Machine Learning*, 2015, pp. 1689–1698.

[19] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *ICML*, 2012.

[20] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 681–688.

[21] L. J. Ratliff, M. Jin, I. C. Konstantakopoulos, C. Spanos, and S. S. Sastry, "Social game for building energy efficiency: Incentive design," in *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*. IEEE, 2014, pp. 1011–1018.

[22] M. Jin, R. Jia, Z. Kang, I. C. Konstantakopoulos, and C. J. Spanos, "Presencesense: Zero-training algorithm for individual presence detection based on power monitoring," in *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*. ACM, 2014, pp. 1–10.

[23] A. Tversky and D. Kahneman, "The framing of decisions and the psychology of choice," in *Environmental Impact assessment, technology assessment, and risk analysis*. Springer, 1985, pp. 107–129.