Data Poisoning Attacks on Multi-Task Relationship Learning

Mengchen Zhao, Bo An, Yaodong Yu, Sulin Liu, Sinno Jialin Pan
School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798
zhao0204@e.ntu.edu.sg, {boan,ydyu,liusl,sinnopan}@ntu.edu.sg

Abstract

Multi-task learning (MTL) is a machine learning paradigm that improves the performance of each task by exploiting useful information contained in multiple related tasks. However, the relatedness of tasks can be exploited by attackers to launch data poisoning attacks, which has been demonstrated a big threat to single-task learning. In this paper, we provide the first study on the vulnerability of MTL. Specifically, we focus on multi-task relationship learning (MTRL) models, a popular subclass of MTL models where task relationships are quantized and are learned directly from training data. We formulate the problem of computing optimal poisoning attacks on MTRL as a bilevel program that is adaptive to arbitrary choice of *target* tasks and *attacking* tasks. We propose an efficient algorithm called PATOM for computing optimal attack strategies. PATOM leverages the optimality conditions of the subproblem of MTRL to compute the implicit gradients of the upper level objective function. Experimental results on real-world datasets show that MTRL models are very sensitive to poisoning attacks and the attacker can significantly degrade the performance of target tasks, by either directly poisoning the target tasks or indirectly poisoning the related tasks exploiting the task relatedness. We also found that the tasks being attacked are always strongly correlated, which provides a clue for defending against such attacks.

Introduction

The security of machine learning algorithms has become a great concern in many real-world applications involving adversaries. The threats to machine learning systems can be classified as two kinds: exploratory attacks where attackers modify test examples in order to make machine learning algorithms produce erroneous outputs (Li and Vorobeychik 2014; Biggio et al. 2013), and causative attacks (a.k.a poisoning attacks) where attackers manipulate training examples to subvert the learned model (Barreno et al. 2010; Biggio, Nelson, and Laskov 2012; Xiao et al. 2015). Poisoning attacks usually occur when training data is collected from public sources (e.g., Internet users) and can be very harmful due to its long-lasting effect on the learned model. Studying poisoning attacks provides deep understanding of how well machine learning performs in adversarial training environment, which is critical for improving the robustness of real-world machine learning systems.

In this paper, we formally analyze optimal poisoning attacks on multi-task learning (MTL) models, where multiple tasks are learned jointly to achieve better performance than single-task learning (Zhang and Yang 2017). Specifically, we focus on multi-task relationship learning (MTRL) models, a popular subclass of MTL models where task relationships are quantized and are learned directly from training data (Zhang and Yeung 2010; Liu, Pan, and Ho 2017). Many MTL-based machine systems collect training data from individual users to provide personalized services, including collaborative spam filtering and personalized recommendations, which makes them vulnerable to poisoning attacks launched by cyber criminals. For example, in an MTL-based recommender system, attackers can control a considerable number of user accounts either by hacking existing user accounts or creating fictitious user accounts.

Previous works on poisoning attacks focus on single-task learning (STL) models, including support vector machines (Biggio, Nelson, and Laskov 2012), autoregressive models (Alfeld, Zhu, and Barford 2016) and factorization-based collaborative filterings (Li et al. 2016). However, none of them studies poisoning attacks on MTL models. Computing optimal poisoning attacks on MTL models can be much more challenging than on STL models, because MTL tasks are related with each other and an attack on one task might potentially influence other tasks. This also opens a door for the attacker to attack some accessible tasks and indirectly influence the unaccessible target tasks, which cannot be addressed by existing methods on poisoning STL models.

The major contributions of our work are threefold. First, we formulate the optimal poisoning attack problem on MTRL as a bilevel program that is adaptive to any choice of target tasks and attacking tasks. Second, we develop a stochastic gradient ascent based algorithm called PATOM for solving the optimal attack problem, where the gradients are computed based on the optimality conditions of the convex subproblem of MTRL. Third, we demonstrate experimentally that MTRL is very sensitive to data poisoning attacks. The attacker can significantly degrade the performance of target tasks, by either directly poisoning the target tasks or indirectly poisoning the related tasks. Moreover, we study the change of task relationships under attacks and found that the attacking tasks usually have strong local correlations, which suggests that a group of strongly correlated tasks could be dangerous to the learner.

Related Work

Data poisoning attacks against machine learning algorithms have become an important research topic in adversarial machine learning (Barreno et al. 2006; Huang et al. 2011; Kloft and Laskov 2010; Lowd and Meek 2005). The first work that provides formal study of poisoning attacks investigates the vulnerability of support vector machines, where an attacker progressively injects malicious data points to the training set in order to maximize the classification error (Biggio, Nelson, and Laskov 2012). Xiao et al.(2015) study poisoning attacks on feature selection algorithms and propose an algorithm that repeatedly optimizing the injected data until convergence. Mei and Zhu (2015b) propose an algorithmic framework for computing training set attacks. Recently, poisoning attacks have been analyzed on many important machine learning algorithms, including autoregressive models (Alfeld, Zhu, and Barford 2016), latent Dirichlet allocation (Mei and Zhu 2015a), and matrix factorization-based collaborative filtering (Li et al. 2016). However, existing works focus only on STL models and the vulnerability of MTL models is left to be explored.

Another line of research related to our work is MTL, which has been extensively studied in the literature. In general, MTL can be categorized into four classes: feature learning approaches, low-rank approaches, task clustering approaches, and task relationship approaches. Feature learning approaches aim to learn a common shared feature space among multiple tasks to boost the learning performance of each task (Argyriou, Evgeniou, and Pontil 2007). Low-rank approaches assume that the model parameters of different tasks share a low-rank structure, and discovery of such a low-rank structure could help learning a more precise model for each task (Ando and Zhang 2005). Task clustering approaches assume that different tasks form several task-clusters, each of which consists of similar tasks (Thrun and O'Sullivan 1996). Task relationship learning aims to quantify and learn task relationship automatically from data, such that knowledge can be transferred among related tasks (Zhang and Yeung 2010). However, as we discussed, the vulnerability of MTL has never been studied. In this work, we fill the gap by investigating the vulnerability of task relationship learning approaches, which have proven to be effective in MTL.

Multi-Task Relationship Learning

We denote by $T = \{T_i\}_{i=1}^m$ the set of learning tasks. For each task T_i, we are given a set of training data $D_i = \{(\mathbf{x}_j^i, y_j^i) | \mathbf{x}_j^i \in \mathbb{R}^d, j = 1, ..., n_i\}$. The label $y_j^i \in \mathbb{R}$ if the task is a regression task and $y_j^i \in \{-1, +1\}$ if the task is a binary classification task. Note that a multi-class classification problem can be easily decomposed to a set of binary classification problems using the one-vs-the-rest strategy (Fan et al. 2008). The goal of MTL is to jointly learn a prediction function $f_i(\mathbf{x})$ for each task. In this paper, we consider linear prediction functions where $f_i(\mathbf{x}) = (\mathbf{w}^i)^\top \mathbf{x} + b_i$, but note that it is easy to extend to non-linear cases using kernel methods. For the ease of representation, we denote $(\mathbf{x}, 1)$ by $\mathbf{x}$ and denote $(\mathbf{w}, b)$ by $\mathbf{w}$ so that $f_i(\mathbf{x}) = (\mathbf{w}^i)^\top \mathbf{x}$.

We consider a general multi-task relationship learning (MTRL) formulation (Zhang and Yeung 2010) as follows, which includes many existing popular MTL methods as its special cases (Evgeniou, Micchelli, and Pontil 2005; Evgeniou and Pontil 2004; Jacob, Vert, and Bach 2009; Kato et al. 2008).

$$\begin{aligned} \min_{\mathbf{W},\mathbf{\Omega}} \quad & \sum_{i=1}^m \frac{1}{n_i} \sum_{j=1}^{n_i} l((\mathbf{w}^i)^\top \mathbf{x}_j^i, y_j^i) + \frac{\lambda_1}{2} \mathrm{tr}(\mathbf{W}\mathbf{W}^\top) \\ & + \frac{\lambda_2}{2} \mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top), \qquad (1) \\ \text{s.t.} \quad & \mathbf{\Omega} \succeq 0, \mathrm{tr}(\mathbf{\Omega}) = 1, \end{aligned}$$

where $l(\cdot)$ is an arbitrary convex loss function, $\mathbf{W}$ is a matrix whose i-th column $\mathbf{w}^i$ is the weight vector of task T_i, $\mathbf{\Omega} \in \mathbb{R}^{m \times m}$ is the covariance matrix that describes positive, negative and unrelated task relationships. The first term in the objective function measures the empirical loss of all tasks with the term $1/n_i$ to balance the different sample sizes of tasks. The second term in the objective function is to penalize the complexity of $\mathbf{W}$, and the last term serves as the task-relationship regularization term. The first constraint ensures that the covariance matrix $\mathbf{\Omega}$ is positive semi-definite, and the second constraint controls its complexity.

Data Poisoning Attacks on MTRL

In this section, we introduce the problem settings for the data poisoning attack on MTRL. We define three kinds of attacks based on real-world scenarios and propose a bilevel formulation for computing optimal attacks.

We assume that the attacker aims to degrade the performance of a set of *target* tasks $T_{tar} \subset T$ by injecting data to a set of *attacking* tasks $T_{att} \subset T$. We denote by $\widehat{D}_i = \{(\widehat{\mathbf{x}}_j^i, \widehat{y}_j^i) | \widehat{\mathbf{x}}_j^i \in \mathbb{R}^d, j = 1, ..., \widehat{n}_i\}$ the set of malicious data injected to task i. Specially, $\widehat{D}_i = \varnothing$, i.e., $\widehat{n}_i = 0$, if $T_i \notin T_{att}$. We define and study the following three kinds of attacks based on real-world scenarios.

- **Direct attack**: $T_{tar} = T_{att}$. Attacker can directly inject data to all the target tasks. For example, in product review sentiment analysis, each task is a sentiment classification task that classifies a review as negative or positive. On e-commerce platforms such as Amazon, attackers can directly attack the target tasks by providing crafted reviews to the target products.
- **Indirect attack**: $T_{tar} \cap T_{att} = \varnothing$. Attacker cannot inject data to any of the target tasks. However, he can inject data to other tasks and indirectly influence the target tasks. For example, personalized recommendations treat each user as a task and use users' feedback to train personalized recommendation models. In such scenarios, attackers usually cannot access the training data of target tasks. However,

attackers can launch indirect attacks by faking some malicious user accounts, which will be treated as attacking tasks, and providing crafted feedback to the systems.

- **Hybrid attack**: A mixture of direct attack and indirect attack where the attacker can inject data to both target tasks and attacking tasks.

We denote by $\mathcal{L}(D, \mathbf{w}) = \sum_{k=1}^{|D|} l(\mathbf{w}^\top \mathbf{x}_k, y_k)$ the empirical loss incurred by weight vector $\mathbf{w}$ on data set D, and define the attacker's utility function as the empirical loss on training data of the target tasks:

$$\mathcal{U} = \sum_{\{i|T_i \in T_{tar}\}} \mathcal{L}(D_i, \mathbf{w}^i).$$

Following the Kerckhoffs' principle (Kahn 1998) and existing works on poisoning attacks (Biggio, Nelson, and Laskov 2012; Li et al. 2016), we assume that the attacker has full knowledge of the victim MTRL model. In reality, attackers can either obtain the knowledge of victim models by exploiting insider threats (Greitzer et al. 2008) or probing the machine learning systems by sending queries from the outside (Lowd and Meek 2005). We then formulate the optimal attack problem as the following bilevel optimization problem. Problem (2) is the upper level problem, in which the objective function is the attacker's utility $\mathcal{U}$. The variables of the upper level problem are the injected data points $\widehat{D}_i$, which are usually constrained in real-world scenarios. For example, the injected data should have similar scale with the clean data. The lower level problem (Problem (3)) is an MTRL problem with training set consists of both clean and injected data points. The lower level problem can be regarded as the constraint of the upper level problem. In other words, the variables $\mathbf{W}$ used for computing the objective of Problem (2) should be the optimal solution of the lower level problem.

$$\max_{\{\widehat{D}_i|T_i \in T_{att}\}} \sum_{\{i|T_i \in T_{tar}\}} \mathcal{L}(D_i, \mathbf{w}^i), \tag{2}$$

$$\text{s.t.} \quad \text{Constraints on } \{\widehat{D}_i | T_i \in T_{att}\},$$

$$\min_{\mathbf{W}, \mathbf{\Omega}} \quad \sum_{i'=1}^{m} \frac{1}{n_{i'} + \widehat{n}_{i'}} \mathcal{L}(D_{i'} \cup \widehat{D}_{i'}, \mathbf{w}^{i'}) + \frac{\lambda_1}{2} \text{tr}(\mathbf{W}\mathbf{W}^\top) + \frac{\lambda_2}{2} \text{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top), \tag{3}$$

$$\text{s.t.} \quad \mathbf{\Omega} \succeq 0, \text{tr}(\mathbf{\Omega}) = 1.$$

Computing Optimal Attack Strategies

In this section, we propose an algorithm called PATOM for computing optimal attack strategies. PATOM is a projected stochastic gradient ascent based algorithm that efficiently maximizes the injected data in the direction of increasing the empirical loss of target tasks. Since there is no close-form relation between the empirical loss and the injected data, we compute the gradients exploiting the optimality conditions of the subproblem of MTRL.

General Optimization Framework

Bilevel problems are usually hard to solve due to their non-linearity, non-differentiability and non-convexity. In our bilevel formulation, although the upper level problem (2) is relatively simple, the lower level problem (3) is highly non-linear and non-convex. Inspired by (Li et al. 2016; Mei and Zhu 2015b; Xiao et al. 2015; Zhao et al. 2017), we use a projected gradient ascent method to solve our proposed bilevel problem. The idea is to iteratively update the injected data in the direction of maximizing the attacker's utility function $\mathcal{U}$. In order to reduce the complexity of the optimal attack problem, we fix the labels of injected data $\widehat{y}_j^i$ and optimize over the features of injected data $\widehat{\mathbf{x}}_j^i$. The update rule is written as follows,

$$(\widehat{\mathbf{x}}_j^i)^t \leftarrow \text{Proj}_{\mathbb{X}}((\widehat{\mathbf{x}}_j^i)^{t-1} + \eta \nabla_{(\widehat{\mathbf{x}}_j^i)^{t-1}} \mathcal{U}), \tag{4}$$

where η is the step size, t denotes the t-th iteration, and $\mathbb{X}$ represents the feasible region of the injected data, which is specified by the first constraint in the upper level problem (2). We consider $\mathbb{X}$ as an ℓ_2-norm ball with diameter r. Therefore, $\text{Proj}_{\mathbb{X}}$ can be represented by:

$$\text{Proj}_{\mathbb{X}}(\mathbf{x}) = \begin{cases} \mathbf{x}, & \text{if } ||\mathbf{x}||_2 \leq r, \\ \frac{\mathbf{x}r}{||\mathbf{x}||_2}, & \text{if } ||\mathbf{x}||_2 > r. \end{cases}$$

In order to compute the gradients $\nabla_{(\widehat{\mathbf{x}}_j^i)^{t-1}} \mathcal{U}$, we first apply the chain rule to arrive at

$$\nabla_{(\widehat{\mathbf{x}}_j^i)} \mathcal{U} = \nabla_W \mathcal{U} \cdot \nabla_{(\widehat{\mathbf{x}}_j^i)} \mathbf{W}. \tag{5}$$

However, note that $\mathcal{U}$ is the sum of losses incurred by every point in the target tasks, the first term on the right side could be computationally expensive if the number of data points in target tasks is large. Therefore, we instead propose a projected stochastic gradient ascent based algorithm, called PATOM, to improve the scalability of our approach.

The details of PATOM is shown in Algorithm 1. We first randomly initialize the injected data $\widehat{D}_i$ within the ℓ_2-norm ball with diameter r. Using the injected data, we solve the MTRL problem (the lower level problem (3)), and obtain the initial values of the weight matrix $\mathbf{W}_0$ and the covariance matrix $\mathbf{\Omega}_0$. In each iteration, we perform a projected stochastic gradient ascent procedure steps (7-10) on all the injected data. Specifically, for each data point $(\mathbf{x}_q^p, y_q^p)$ sampled from D_{batch}, we compute the gradients of its associate loss $l((\mathbf{w}_t^p)^\top \mathbf{x}_q^p, y_q^p)$ with respect to each injected data $\widehat{\mathbf{x}}_j^i$. Therefore, by replacing $\mathcal{U}$ in (4) with $l((\mathbf{w}_t^p)^\top \mathbf{x}_q^p, y_q^p)$ we have the stochastic version of the update rule as shown in (6). Then, with the updated injected data $\widehat{D}_i = \widehat{D}_i^t$, we solve the lower level problem (3) again to obtain a new weight matrix $\mathbf{W}_t$ and a new covariance matrix $\mathbf{\Omega}_t$, which will be used in the next iteration.

$$(\widehat{\mathbf{x}}_j^i)^t \leftarrow \text{Proj}_{\mathbb{X}}((\widehat{\mathbf{x}}_j^i)^{t-1} + \eta \nabla_{(\widehat{\mathbf{x}}_j^i)^{t-1}} l((\mathbf{w}_{t-1}^p)^\top \mathbf{x}_q^p, y_q^p)). \tag{6}$$

Gradients Computation

In order to compute the gradients $\nabla_{(\widehat{\mathbf{x}}_j^i)} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p)$ in (6), we still apply the chain rule and obtain:

$$\nabla_{\widehat{\mathbf{x}}_j^i} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p) = \nabla_{\mathbf{w}^p} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p) \cdot \nabla_{\widehat{\mathbf{x}}_j^i} \mathbf{w}^p. \tag{7}$$

Algorithm 1: computing Poisoning ATtacks On Multi-task relationship learning (PATOM)

```
1  Input: $T_{tar}, T_{att}$, step size $\eta$, attacker budget $\widehat{n}_i$.
2  Randomly initialize
     $\widehat{D}_i^0 = \{((\widehat{\mathbf{x}}_j^i)^0, (\widehat{y}_j^i)^0) | j = 1, ..., \widehat{n}_i\}, \forall i \in T_{att}$.
3  $\widehat{D}_i = \widehat{D}_i^0, , \forall i \in T_{att}$.
4  Solve lower level problem (3) to obtain $\mathbf{W}_0$ and $\mathbf{\Omega}_0$.
5  $t \leftarrow 1$.
6  while $t < t^{max}$ do
7      Sample a batch $D_{batch}$ from $\cup_{i \in T_{tar}} D_i$.
8      for $(\mathbf{x}_q^p, y_q^p) \in D_{batch}$ do
9          for $i \in T_{att}, j = 1...\widehat{n}_i$ do
10             Update $(\widehat{\mathbf{x}}_j^i)^t$ according to (6).
11     $\widehat{D}_i = \widehat{D}_i^t, \forall i \in T_{att}$.
12     Solve (3) to obtain $\mathbf{W}_t$ and $\mathbf{\Omega}_t$.
13     $t \leftarrow t + 1$.
```

We can see that the first term on the right side depends only on the loss function $l(\cdot)$ and is relatively easy to compute. However, the second term on the right side depends on the optimality conditions of lower level problem (3). In the rest of this section, we show how to compute the gradients with respect to two commonly used loss functions. For regression tasks, we adopt least-square loss: $l_1(\mathbf{w}^\top \mathbf{x}, y) = (y - \mathbf{w}^\top \mathbf{x})^2$. For classification tasks, we adopt squared hinge loss: $l_2(\mathbf{w}^\top \mathbf{x}, y) = (1 - y\mathbf{w}^\top \mathbf{x})^2$.

We first fix $\mathbf{\Omega}$ to eliminate the constraints of the lower level problem (3), and obtain the following sub-problem:

$$\min_{\mathbf{W}} \quad \sum_{i=1}^{m} \frac{1}{n_i + \widehat{n}_i} \mathcal{L}(D_i \cup \widehat{D}_i, \mathbf{w}^i) + \frac{\lambda_1}{2} \text{tr}(\mathbf{W}\mathbf{W}^\top) + \frac{\lambda_2}{2} \text{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top). \tag{8}$$

As shown in (Zhang and Yeung 2010), MTRL problems can be solved by an alternating approach with $\mathbf{\Omega}$ and $\mathbf{W}$ alternatingly fixed in each iteration. Also note that in bilevel optimization, the optimality of the lower level problem can be considered as a constraint to the upper level problem. Therefore, at convergence, we can treat $\mathbf{\Omega}$ in Problem (8) as a constant-value matrix when computing the gradients. We then substitute the least-square loss function $l_1(\cdot)$ into Problem (8) and reformulate it as the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \sum_{i=1}^{m} \frac{1}{n_i + \widehat{n}_i} \left(\sum_{j=1}^{n_i} (\varepsilon_j^i)^2 + \sum_{j'=1}^{\widehat{n}_i} (\hat{\varepsilon}_j^i)^2 \right) + \frac{\lambda_1}{2} \text{tr}(\mathbf{W}\mathbf{W}^\top) \\ & + \frac{\lambda_2}{2} \text{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top), \\ \text{s.t.} \quad & \varepsilon_j^i = y_j^i - (\mathbf{w}^i)^\top \mathbf{x}_j^i, \ \ \forall i, j, \\ & \hat{\varepsilon}_{j'}^i = \widehat{y}_{j'}^i - (\mathbf{w}^i)^\top \widehat{\mathbf{x}}_{j'}^i, \ \ \forall i, j'. \end{aligned} \tag{9}$$

The Lagrangian of the problem (9) is:

$$\begin{aligned} \mathcal{G} = & \sum_{i=1}^{m} \frac{1}{n_i + \widehat{n}_i} \left(\sum_{j=1}^{n_i} (\varepsilon_j^i)^2 + \sum_{j=1}^{\widehat{n}_i} (\hat{\varepsilon}_j^i)^2 \right) + \frac{\lambda_1}{2} \text{tr}(\mathbf{W}\mathbf{W}^\top) \\ & + \frac{\lambda_2}{2} \text{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top) \\ & + \sum_{i=1}^{m} \left(\sum_{j=1}^{n_i} \alpha_j^i \left(y_j^i - (\mathbf{w}^i)^\top \mathbf{x}_j^i - \varepsilon_j^i \right) \right. \\ & \left. + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i (\widehat{y}_{j'}^i - (\mathbf{w}^i)^\top \widehat{\mathbf{x}}_{j'}^i - \hat{\varepsilon}_{j'}^i) \right). \end{aligned} \tag{10}$$

The gradient of $\mathcal{G}$ with respect to $\mathbf{W}$ is:

$$\begin{aligned} \frac{\partial G}{\partial \mathbf{W}} = & \mathbf{W}(\lambda_1 \mathbf{I}_m + \lambda_2 \mathbf{\Omega}^{-1}) \\ & - \sum_{i=1}^{m} \left(\sum_{j=1}^{n_i} \alpha_j^i \mathbf{x}_j^i \mathbf{e}_i^\top + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i \widehat{\mathbf{x}}_{j'}^i \mathbf{e}_i^\top \right). \end{aligned} \tag{11}$$

where $\mathbf{I}_m$ is $m \times m$ identity matrix, and $\mathbf{e}_i$ is the i-th column of $\mathbf{I}_m$. By setting $\frac{\partial G}{\partial \mathbf{W}} = \mathbf{0}$, we obtain:

$$\mathbf{W} = \sum_{i=1}^{m} \left(\left(\sum_{j=1}^{n_i} \alpha_j^i \mathbf{x}_j^i + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i \widehat{\mathbf{x}}_{j'}^i \right) \mathbf{e}_i^\top \mathbf{\Omega}(\lambda_1 \mathbf{\Omega} + \lambda_2 \mathbf{I}_n)^{-1} \right), \tag{12}$$

which implies that each task's weight vector $\mathbf{w}^i$ can be represented as a linear combination of training data from all tasks. For simplicity in presentation, we denote by $\mathbf{\Phi} = \mathbf{\Omega}(\lambda_1 \mathbf{\Omega} + \lambda_2 \mathbf{I}_n)^{-1}$, and reexpress (12) as the following form:

$$\mathbf{w}^p = \sum_{i=1}^{m} \Phi_{i,p} \left(\sum_{j=1}^{n_i} \alpha_j^i \mathbf{x}_j^i + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i \widehat{\mathbf{x}}_{j'}^i \right), p = 1...m. \tag{13}$$

Similarly, we substitute the squared hinge loss into the loss function $\mathcal{L}(\cdot)$ in Problem (8), and obtain:

$$\mathbf{w}^p = \sum_{i=1}^{m} \Phi_{i,p} \left(\sum_{j=1}^{n_i} \alpha_j^i y_j^i \mathbf{x}_j^i + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i \widehat{y}_{j'}^i \widehat{\mathbf{x}}_{j'}^i \right), p = 1...m. \tag{14}$$

Given (13) and (14), we can compute the gradient in (6). In case of the least-square loss, we have:

$$\begin{aligned} \nabla_{\widehat{\mathbf{x}}_j^i} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p) &= 2((\mathbf{w}^p)^\top \mathbf{x}_q^p - y_q^p) \mathbf{x}_q^p \frac{\partial \mathbf{w}^p}{\partial \widehat{\mathbf{x}}_j^i} \\ &= 2((\mathbf{w}^p)^\top \mathbf{x}_q^p - y_q^p) \mathbf{x}_q^p \widehat{\alpha}_j^i \Phi_{i,p}. \end{aligned} \tag{15}$$

In case of the squared hinge loss, we have:

$$\begin{aligned} & \nabla_{\widehat{\mathbf{x}}_j^i} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p) \\ &= 2(y_q^p (\mathbf{w}^p)^\top \mathbf{x}_q^p - 1) y_q^p \mathbf{x}_q^p \frac{\partial \mathbf{w}^p}{\partial \widehat{\mathbf{x}}_j^i} \\ &= 2(y_q^p (\mathbf{w}^p)^\top \mathbf{x}_q^p - 1) y_q^p \mathbf{x}_q^p \widehat{y}_j^i \widehat{\alpha}_j^i \Phi_{i,p}. \end{aligned} \tag{16}$$

Experimental Results

In this section, we first evaluate PATOM in terms of convergence and solution quality. Experimental results show that PATOM converges to local optima in less than 10 iterations and the attack strategies computed by PATOM significantly outperform baselines. Second, we study the task relationships under the data poisoning attacks and found that task relationships are very sensitive to the attacks. We also found that the tasks under attacking form strong correlations.

Datasets

We use three real-world datasets to validate our proposed methods. The Landmine and the MNIST datasets are used for classification tasks and Sarcos dataset is used for regression tasks. For each dataset, all data points are divided by the maximum ℓ_2 norm among them, so that all data points are within a ℓ_2-norm ball with diameter 1. We consider this ball as the feasible region of the injected data in order to ensure that the injected data and the clean data are at the same scale. We use the area under the ROC curve (AUC) to evaluate the learning performance for classification tasks, and the normalized mean squared error (NMSE) for regression tasks. The higher AUC corresponds to the better performance for classification and the lower NMSE corresponds to the better performance for regression. The detailed description of the datasets are given below.

- **Sarcos**[1] relates to an inverse dynamics problem for a 7 degrees-of-freedom SARCOS anthropomorphic robot arm. The input is a 21-dimensional space that includes 7 joint positions, 7 joint velocities and 7 joint accelerations. Each input instance is associated with 7 joint torques. Following previous work (Zhang and Yeung 2010), each task is to learn a mapping from the 21-dimensional input space to one of the 7 torques. The dataset contains 44,484 training examples and 4,449 test examples.
- **Landmine**[2] consists of 29 tasks collected from various landmine fields. A data point in each task is represented by a 9-dimensional feature vector, and associated with a corresponding binary label ("1" for landmine and "-1" for cluster). The feature vectors are extracted from radar images, concatenating four momentbased features, three correlation-based features, one energy ratio feature and one spatial variance feature. The tasks entail different numbers of data points, varying from 89 to 138 examples.
- **MNIST**[3] is a hand-written digit dataset with 10 classes. We use the one-vs-the-rest strategy to decompose the multi-class classification problem to 10 binary classification problems, and treat each binary classification problem as a task. To form the training set for each task, we randomly draw 300 data points of the designated digits and assign label "+1" and draw an equal number of instances from other classes randomly and assign label "-1". The dataset contains 60,000 training examples and 10,000 test examples. We use principal component analysis (PCA) to reduce the feature space to a 128-dimensional space.

[1]`http://www.gaussianprocess.org/gpml/data/`.

[2]`http://people.ee.duke.edu/~lcarin/LandmineData.zip`.

[3]`http://yann.lecun.com/exdb/mnist/`.

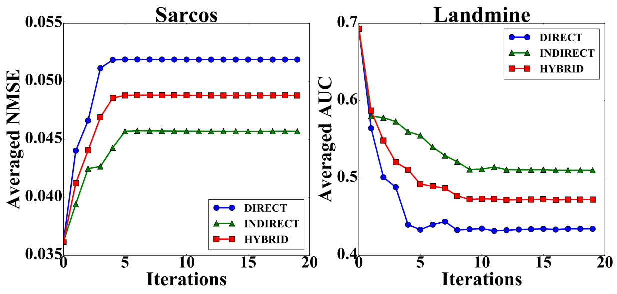


Figure 1: Convergence of PATOM.

Evaluating Convergence of PATOM

Our first set of experiments study the convergence of PATOM on Sarcos dataset and Landmine dataset, with respect to regression tasks and classification tasks. On Sarcos dataset, we randomly draw 300 training examples and 600 test examples from the associated training and test set. For direct attacks, we select 3 tasks on Sarcos dataset and 15 tasks on Landmine dataset as the respective target tasks, and set the attacking tasks the same as the target tasks. For indirect attacks, we use the same target tasks as in the direct attack, and treat the rest of tasks as the attacking tasks. For hybrid attacks, we randomly select the same number of attacking tasks as in the indirect attack experiments from all tasks. We set the step size $\eta = 100$ and the lower level problem parameters $\lambda_1 = \lambda_2 = 0.1$. The batch size is set to be three times of the clean data. The number of injected data points in each task is set to be 20% of the clean data.

Figure 1 shows the results of the convergence experiments on the two datasets, where x-axis represents the number of iterations in PATOM, and y-axis represents the NMSE averaged over target tasks of Sarcos dataset and the AUC averaged over target tasks on Landmine dataset, respectively. We can see that for all the three kinds of attacks on the two datasets, PATOM converges to local optima in less than 10 iterations, where at iteration 0 the injected data points are randomly initialized. Since existing optimization techniques cannot guarantee global optimal solutions for nonconvex programs, all of the solutions we find are approximate. However, we can get an estimation of the global optimal solution by selecting multiple start points and comparing the local optima. In our experiments, we observe very similar local optima values when choosing multiple start points. Based on this observation, we run PATOM with one start point in our remaining experiments.

Evaluating Solution Qualities

Our second set of experiments evaluates the performance of MTRL under direct attacks and indirect attacks with respect to different datasets. On each dataset, we select 4 different pairs of target task set and attacking task set. Each pair

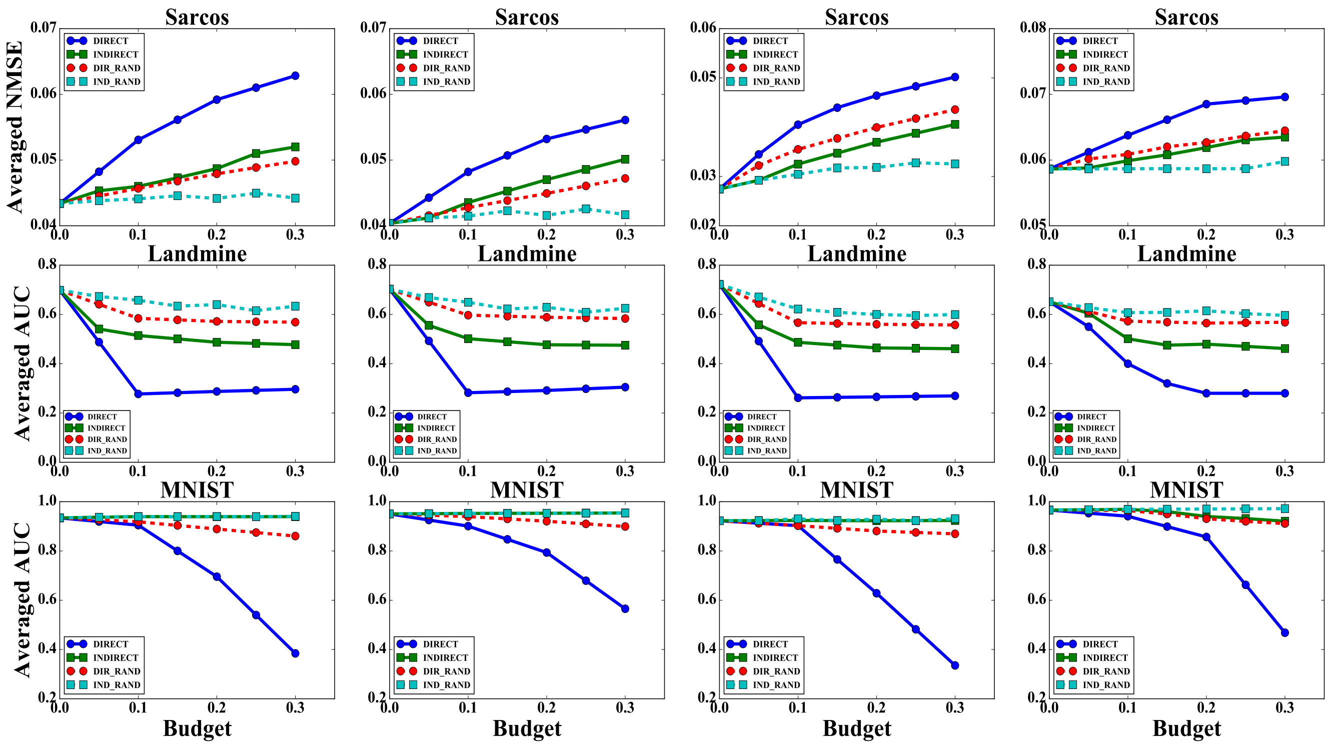
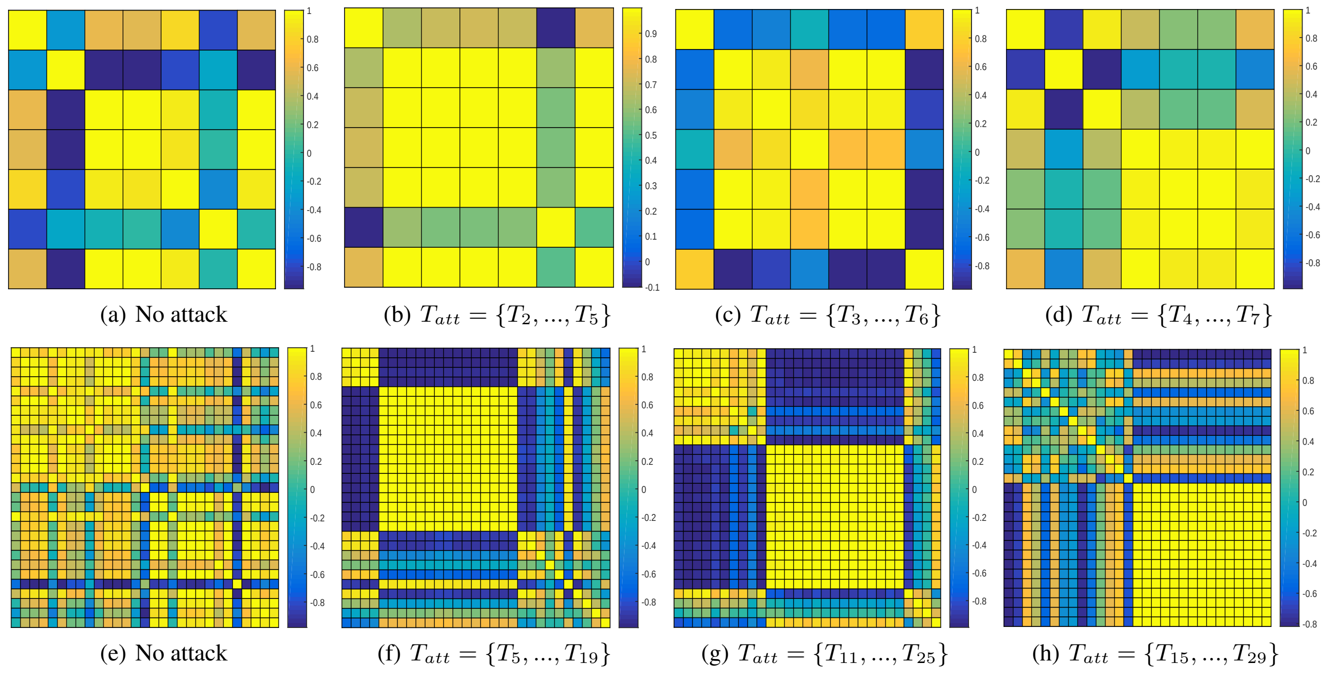


Figure 2: Solution quality comparison. Each figure corresponds to a choice of pair (T_{tar}, T_{att}) of the associated dataset. The bold line (dashed line) with circle marker represents direct attacks (random direct attacks); the bold line (dashed line) with square marker represents indirect attacks (random indirect attacks). The budget represents the ratio of the number of injected data points to the number of clean data points.

(T_{tar}, T_{att}) is chosen by randomly selecting half of tasks to form T_{tar} and the rest of tasks to form T_{att}. We have $|T_{tar}| = 4$ and $|T_{att}| = 3$ on Sarcos dataset, $|T_{tar}| = 15$ and $|T_{att}| = 14$ on Landmine dataset, and $|T_{tar}| = 5$ and $|T_{att}| = 5$ on MNIST dataset. For a pair (T_{tar}, T_{att}) of each dataset, we compare the averaged NMSE or averaged AUC over the target tasks under four kinds of attacks: direct attacks, indirect attacks, random direct attacks and random indirect attacks. The last two kinds of attacks are treated as baselines, where the injected data points are randomly chosen. Figure 2 shows the results of quality comparison among the four kinds of attacks. Some interesting findings include:

- Direct attacks are more effective than indirect attacks and random attacks given the same budget. From Figure 2, we can see that direct attacks significantly degrade the learning performance on all datasets. For example, on Sarcos dataset, direct attacks with 30% malicious data injected leads to about 50% higher averaged NMSE. However, note that in some scenarios, attackers may have larger budget for launching indirect attacks. Take the recommender system for example, attackers can provide arbitrary number of training data through the malicious accounts created by themselves. In such cases, indirect attacks are also big threats to the learning system.
- Both direct attacks and indirect attacks computed by PATOM significantly outperform random attacks, respectively, which demonstrates that the real-world attackers can do much better than just launching random attacks.
- Different choices of pairs (T_{tar}, T_{att}) influence the attacks' performance. For example, we can see from the second figure of the first row of Figure 2, the indirect attacks lead to a higher loss than random direct attacks. However, in the third figure of the first row, the random direct attacks lead to a higher loss than indirect attacks.
- Indirect attacks almost have no effect on MNIST dataset. Since we can easily learn good classifiers on MNIST dataset using only hundreds of training examples, each task does not need much help from other tasks and the task correlations are relatively low. Consequently, it is hard for the attacker to launch effective indirect attacks by exploiting task relationships.
- The AUC slightly increases after budget=0.1 on MNIST dataset. Since our formulation maximizes the empirical loss on the training data but the AUC is computed based on the test data, we think the AUC on test data reaches the lower bound near budget=0.1 and slightly increases after that due to the distribution discrepancy between training data and test data. The intuition behind this is similar to 'overfitting' in the convention of machine learning.


(a) No attack (b) $T_{att} = \{T_2, ..., T_5\}$ (c) $T_{att} = \{T_3, ..., T_6\}$ (d) $T_{att} = \{T_4, ..., T_7\}$

(e) No attack (f) $T_{att} = \{T_5, ..., T_{19}\}$ (g) $T_{att} = \{T_{11}, ..., T_{25}\}$ (h) $T_{att} = \{T_{15}, ..., T_{29}\}$

Figure 3: Visualization of task correlations under attacks. (a) - (d) are the results on Sarcos dataset and (e) - (f) are the results on Landmine dataset. The color of each grid represents the value of the correlation matrix, ranging from -1 (blue) to $+1$ (yellow) as shown in the color bar at the right side of each figure. The first figure of each row is the ground-truth task correlations learned with clean data. The target task set is set to be $T_{tar} = \{T_1, T_2, T_3\}$ on Sarcos dataset and $T_{tar} = \{T_1, ..., T_{15}\}$ for Landmine dataset and remains the same under different attacks.

Evaluating Task Relationships

Our third set of experiments study the task relationships under different attacks. We fix the target task set as $T_{tar} = \{T_1, T_2, T_3\}$ on Sarcos dataset and $T_{tar} = \{T_1, ..., T_{15}\}$ on Landmine dataset. Then, for each dataset, we select three different attacking task sets and compute three hybrid attacks. We set the amount of injected data to be 30% of the clean data with respect to each task. We convert the learned covariance matrices to correlation matrices and visualize them in Figure 3. Since the Sarcos dataset has 7 tasks and the Landmine dataset has 29 tasks, the learned correlation matrix is a 7×7 symmetric matrix on Sarcos dataset and 29×29 symmetric matrix on Landmine dataset.

Figure 3 shows that on both datasets the ground-truth task correlations are significantly subverted by the data poisoning attacks. For example, in Figure 3(a), the ground-truth correlation between tasks 2 and 3 is -0.99, which suggests that the two tasks are highly negatively correlated. However, in Figure 3(b), the correlation between tasks 2 and 3 becomes 0.99, meaning that the two tasks are highly positively correlated. Similar results can be found on Landmine dataset. Moreover, from Figures 3(b) - 3(d) and 3(f) - 3(h), we observe that the attacking tasks are usually highly positive correlated, in contrast with other tasks. This suggests that the machine learner needs to be aware of a group of tasks that form strong local correlations.

Conclusion and Future Work

This paper studies the data poisoning attacks on MTRL models. To the best of our knowledge, we are the first to study the vulnerability of MTL. We categorize the data poisoning attacks into direct attacks, indirect attacks and hybrid attacks based on the real-world scenarios. We propose a bilevel formulation that includes the three kinds of attacks to analyze the optimal attack problems. We propose PATOM, a stochastic gradient ascent based approach that leverages the optimality conditions of MTRL to compute the gradients. We evaluate PATOM in terms of convergence and solution quality on real-world datasets. Experimental results show that PATOM converges to local optima in less than 10 iterations and the attack strategies computed by PATOM significantly outperform baselines. We also study the task correlations under data poisoning attacks.

The ultimate goal of studying data poisoning attacks is to develop effective defense strategies against such attacks. In future work, we will consider two classes of potential defense strategies for protecting MTL: data sanitization and improving the robustness of MTL. First, as shown in our experiments of evaluating task relationships, the tasks under attacking show a strong correlation with only 30% data injected. Therefore, the machine learner can examine the data from tasks that form strong local correlations, perhaps through human verifications. Moreover, once a task is demonstrated to be malicious, the learner can examine the tasks that strongly correlate to it, which will significantly reduce the learner's effort in examining the data. Second, improving the robustness of MTL could also be an effective approach to defend against data poisoning attacks. MTL exploits the task relatedness to improve the performance of individual tasks, where such relatedness can also be exploited by the attacker to launch indirect attacks. A possible approach to improve the robustness of MTL is to differentiate the helpful relatedness and the harmful task relatedness, so that we can preserve the helpful relatedness and reduce the harmful relatedness during learning.

Acknowledgements

This research is supported by NRF2015 NCR-NCR003-004 and NCR2016NCR-NCR001-002.

References

Alfeld, S.; Zhu, X.; and Barford, P. 2016. Data poisoning attacks against autoregressive models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 1452–1458.

Ando, R. K., and Zhang, T. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research* 6:1817–1853.

Argyriou, A.; Evgeniou, T.; and Pontil, M. 2007. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, 41–48.

Barreno, M.; Nelson, B.; Sears, R.; Joseph, A. D.; and Tygar, J. D. 2006. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, 16–25.

Barreno, M.; Nelson, B.; Joseph, A. D.; and Tygar, J. 2010. The security of machine learning. *Machine Learning* 81(2):121–148.

Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2013. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 387–402.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning*, 1807–1814.

Evgeniou, T., and Pontil, M. 2004. Regularized multi–task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 109–117.

Evgeniou, T.; Micchelli, C. A.; and Pontil, M. 2005. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research* 6:615–637.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research* 9(Aug):1871–1874.

Greitzer, F. L.; Moore, A. P.; Cappelli, D. M.; Andrews, D. H.; Carroll, L. A.; and Hull, T. D. 2008. Combating the insider cyber threat. *IEEE Security & Privacy* 6(1).

Huang, L.; Joseph, A. D.; Nelson, B.; Rubinstein, B. I.; and Tygar, J. 2011. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, 43–58.

Jacob, L.; Vert, J.-p.; and Bach, F. R. 2009. Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems*, 745–752.

Kahn, D. 1998. Codebreakers: The comprehensive history of secret communication from ancient times to the Internet. *Naval War College Review* 51(4):153–155.

Kato, T.; Kashima, H.; Sugiyama, M.; and Asai, K. 2008. Multi-task learning via conic programming. In *Advances in Neural Information Processing Systems*, 737–744.

Kloft, M., and Laskov, P. 2010. Online anomaly detection under adversarial impact. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 405–412.

Li, B., and Vorobeychik, Y. 2014. Feature cross-substitution in adversarial classification. In *Advances in Neural Information Processing Systems*, 2087–2095.

Li, B.; Wang, Y.; Singh, A.; and Vorobeychik, Y. 2016. Data poisoning attacks on factorization-based collaborative filtering. In *Prodeedings of the 30th Annual Conference on Neural Information Processing Systems*, 1885–1893.

Liu, S.; Pan, S. J.; and Ho, Q. 2017. Distributed multi-task relationship learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 937–946.

Lowd, D., and Meek, C. 2005. Adversarial learning. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 641–647.

Mei, S., and Zhu, X. 2015a. The security of latent dirichlet allocation. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 681–689.

Mei, S., and Zhu, X. 2015b. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the 29st AAAI Conference on Artificial Intelligence*, 2871–2877.

Thrun, S., and O'Sullivan, J. 1996. Discovering structure in multiple learning tasks: The TC algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, 489–497.

Xiao, H.; Biggio, B.; Brown, G.; Fumera, G.; Eckert, C.; and Roli, F. 2015. Is feature selection secure against training data poisoning? In *Proceedings of the 32th International Conference on Machine Learning*, 1689–1698.

Zhang, Y., and Yang, Q. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.

Zhang, Y., and Yeung, D.-Y. 2010. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 733–742.

Zhao, M.; An, B.; Gao, W.; and Zhang, T. 2017. Efficient label contamination attacks against black-box learning models. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 3945–3951.