2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

ON THE ADVERSARIAL ROBUSTNESS OF FEATURE SELECTION USING LASSO

Fuwei Li[*]*, Lifeng Lai*[*]*, and Shuguang Cui*[†]

[*]Department of ECE, University of California, Davis
[†] Chinese University of Hong Kong, Shenzhen, China
Email: [*]{fli, lflai}@ucdavis.edu, [†]shuguangcui@cuhk.edu.cn

ABSTRACT

In this paper, we investigate the adversarial robustness of feature selection based on the ℓ_1 regularized linear regression method, named LASSO. In the considered problem, there is an adversary who can observe the whole data set. After seeing the data, the adversary will carefully modify the response values and the feature matrix in order to manipulate the selected features. We formulate this problem as a bi-level optimization problem and cast the ℓ_1 regularized linear regression problem as a linear inequality constrained quadratic programming problem to mitigate the issue caused by non-differentiability of the ℓ_1 norm. We then use the projected gradient descent to design the modification strategy. Numerical examples based on synthetic data and real data both indicate that the feature selection is very vulnerable to this kind of attacks.

***Index Terms*—** Linear regression, sparse learning, LASSO, adversarial machine learning, bi-level optimization.

1. INTRODUCTION

Feature selection plays an important role in machine learning. It selects the most important features while pruning redundant and irrelevant features. By doing so, it dramatically reduces the dimension of the inputs; thus reduces the complexity of machine learning algorithms, speeds up the training and inference process, and improves the generality of the learning machine. Among a variety of the feature selection methods, LASSO [1] is one of the most widely used methods. It imposes ℓ_1 norm constraints on the regression coefficients and performs feature selection and regression simultaneously. Due to its simplicity and efficiency, it has a broad range of applications in biomedical information processing [2], compressed sensing [3], image processing [4], etc. In this paper, we assume the feature selection method is LASSO.

Machine learning is increasingly used in security and safety critical areas [5, 6, 7] that play a vital role in our daily lives. Hence, it is urgent and necessary to study the security of the machine learning algorithms. Even though there are existing works on the robustness of feature selection against outliers and small dense noise [8, 9], the behavior of feature selection algorithms in the presence of adversaries is unclear. The goal of this paper is to fill this void and investigate the adversarial robustness of feature selection methods.

In the considered model, we assume that there exists an adversary who can observe the whole data set and carefully modify the response values and the feature matrix. The goal of the adversary is to make the system select the features which are not chosen originally or make the system discard important features; in the meantime, it tries to keep the change of other features as small as possible to minimize the possibility of being detected by the system. We formulate this problem as a bi-level optimization problem. To address the computational complexity of this complicated optimization problem, we resort to the projected gradient descent method. However, the non-differentiability of the ℓ_1 norm hinders us from computing the gradients of the objective function. To circumvent this, we reformulate the LASSO problem as a linear inequality constrained quadratic programming problem. Using the implicit function theorem [10], we obtain the gradients from the Karush–Kuhn–Tucker (KKT) conditions of the reformulated LASSO problem. Compared with [11], which requires the objective function of the feature selection method being differentiable, our method can be used in any ℓ_1 regularized feature selection methods. Our numerical examples on the synthetic data and real data both indicate that small changes of the response values or the feature matrix will lead to a huge difference of the selected set of features. Hence, feature selection is very vulnerable to adversarial attacks.

The remainder of the paper is organized as follows. In Section 2, we describe the precise problem formulation. In Section 3, we introduce our method to solve this problem. In Section 4, we provide numerical experiments with both synthetic data and real data to illustrate the results obtained in this paper. Finally, we offer concluding remarks in Section 5.

2. PROBLEM FORMULATION

Given the data set $\{(y_0^i, \mathbf{x}_0^i)\}_{i=1}^n$, where n is the number of data samples, y_0^i is the response value of data sample i, $\mathbf{x}_0^i \in$

The work was partially supported by the National Science Foundation with Grants CNS-1824553, CCF-1717943, and CCF-1908258.

978-1-7281-6662-9/20/$31.00

$\mathbb{R}^m$ denotes the feature vector of data sample i, and each element of $\mathbf{x}_0^i$ is called a feature of the data sample. Through the data samples, we attempt to learn a sparse representation of the response values from the features. The LASSO algorithm learns a sparse regression coefficient, $\boldsymbol{\beta}_0$, by

$$\boldsymbol{\beta}_0 = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y}_0 - \mathbf{X}_0\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_{\ell_1}, \tag{1}$$

where the response vector $\mathbf{y}_0 = [y_0^1, y_0^2, \ldots, y_0^n]^\top$, the feature matrix $\mathbf{X}_0 = [\mathbf{x}_0^1, \mathbf{x}_0^2, \ldots, \mathbf{x}_0^n]^\top$, $\|\cdot\|_{\ell_1}$ denotes the ℓ_1 norm, and λ is the trade-off parameter to determine the relative goodness of fitting and sparsity of $\boldsymbol{\beta}_0$ [1]. The location of the non-zero elements of the sparse regression coefficients indicates the corresponding selected features.

In this paper, we assume that there is an adversary who is trying to manipulate the learned regression coefficients, and thus maneuver the selected features by carefully modifying the response values or the feature matrix. We denote the modified response value vector as $\mathbf{y}$ and denote the modified feature matrix as $\mathbf{X}$. Further, we assume that the adversary's modification is constrained by the ℓ_p norm ($p \geq 1$). This means we have $\|\mathbf{y} - \mathbf{y}_0\|_{\ell_p} \leq \eta_y$, and $\|\mathbf{X} - \mathbf{X}_0\|_{\ell_p} \leq \eta_x$, where η_y is the energy budget for the modification of the response values, and η_x is the energy budget for the modification of the feature matrix. For a vector, $\|\cdot\|_{\ell_p}$ denotes the ℓ_p norm of the vector and for a matrix, $\|\cdot\|_{\ell_p}$ denotes the ℓ_p norm of the vectorization of the matrix. As the result, the manipulated regression coefficients, $\hat{\boldsymbol{\beta}}$, are learned from the modified data set $(\mathbf{y}, \mathbf{X})$ by solving the following LASSO problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_{\ell_1}. \tag{2}$$

The goal of the adversary is to suppress or promote some of the regression coefficients while keeping the change of the remaining coefficients to be minimum. If it wants to suppress the ith regression coefficient, we minimize $s_i \cdot \hat{\beta}_i^2$, where $s_i > 0$ is the predefined weight parameter. If it aims to promote the ith regression coefficients, we minimize $e_i \cdot \hat{\beta}_i^2$, where $e_i < 0$ is the weight parameter. To keep the ith regression coefficients not changed so much, we minimize $\mu_i \cdot (\hat{\beta}_i - \beta_0^i)^2$, where $\mu_i > 0$ is a user defined parameter to measure how much effort we put on keeping the ith regression coefficients not changing. Moreover, we denote the set of indices of coefficients which are suppressed, promoted, unchanged as S, E, and U, respectively. In summary, the objective of the adversary is:

$$\min \frac{1}{2}(\hat{\boldsymbol{\beta}} - \boldsymbol{\nu})^\top \mathbf{H}(\hat{\boldsymbol{\beta}} - \boldsymbol{\nu}), \tag{3}$$

where $\nu_i = \beta_0^i$ if $i \in U$, otherwise $\nu_i = 0$, and $\mathbf{H} = \operatorname{diag}(\mathbf{h})$ as $h_i = \mu_i$ for $i \in U$, $h_i = s_i$ for $i \in S$ and $h_i = e_i$ for $i \in E$.

Considering the energy constraints of the adversary, we have the bi-level optimization problem:

$$\min_{\mathbf{y}\in\mathcal{C}_y, \mathbf{X}\in\mathcal{C}_x} \quad \frac{1}{2}(\hat{\boldsymbol{\beta}} - \boldsymbol{\nu})^\top \mathbf{H}(\hat{\boldsymbol{\beta}} - \boldsymbol{\nu}) \tag{4}$$

$$\text{s.t.} \quad \hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_{\ell_1}, \tag{5}$$

where $\mathcal{C}_y = \{\mathbf{y} \mid \|\mathbf{y} - \mathbf{y}_0\|_{\ell_p} \leq \eta_y\}$ and $\mathcal{C}_x = \{\mathbf{X} \mid \|\mathbf{X} - \mathbf{X}_0\|_{\ell_p} \leq \eta_x\}$.

3. PROBLEM ANALYSIS

In this section, we investigate problem (4) and present our projected gradient descent method to solve this problem.

In problem (4), in addition to the constraints $\mathcal{C}_y$ and $\mathcal{C}_x$, the lower-level optimization problem (5) further imposes implicit constraints on the feasible $(\mathbf{y}, \mathbf{X})$ through $\hat{\boldsymbol{\beta}}$. Due to the non-convexity of the feasible set, the bi-level optimization problem is NP-hard in general. To solve this bi-level optimization problem, we need to first solve the lower-level optimization problem to determine the dependence between $(\mathbf{y}, \mathbf{X})$ and $\hat{\boldsymbol{\beta}}$. Then, we can use the gradient descent method to solve this bi-level optimization problem. Since the lower-level problem is convex [1], it can be presented by its KKT conditions. The corresponding KKT conditions with respect to the lower-level optimization problem is:

$$\mathbf{0} \in 2\mathbf{X}^\top(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}) + \lambda\partial\|\boldsymbol{\beta}\|_{\ell_1}, \tag{6}$$

where, $\partial\|\cdot\|_{\ell_1}$ is the subgradient of the ℓ_1 norm. We denote the right hand of (6) as $q(\boldsymbol{\beta}, \mathbf{y}, \mathbf{X})$. The KKT conditions define a one to one mapping from $(\mathbf{y}, \mathbf{X})$ to $\boldsymbol{\beta}$ if $q(\boldsymbol{\beta}, \mathbf{y}, \mathbf{X})$ is a continuous differentiable function and its Jacobian matrix with respect to $\boldsymbol{\beta}$ is invertible. Thus, by the implicit function theorem [10], we can calculate the gradient of $\boldsymbol{\beta}$ with respect to $\mathbf{y}$ and $\mathbf{X}$. However, $q(\boldsymbol{\beta}, \mathbf{y}, \mathbf{X})$ is not differentiable at the point with $\beta_i = 0$. Moreover, (5) does not always determine a single valued mapping from $(\mathbf{y}, \mathbf{X})$ to $\boldsymbol{\beta}$. For example, when $\lambda \geq \|\mathbf{X}^\top\mathbf{y}\|_\infty$, we always have $\boldsymbol{\beta} = \mathbf{0}$.

To circumvent this difficulty, we transform the lower level optimization problem to the following equivalent linear inequality constrained quadratic programming [12]:

$$\underset{\boldsymbol{\beta}, \mathbf{u}}{\operatorname{argmin}} \quad \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\sum_{i=1}^m u_i \tag{7}$$

$$\text{s.t.} \quad -u_i \leq \beta_i \leq u_i,\ i = 1, 2, \ldots, m. \tag{8}$$

Following [12], we can apply the interior-point method to solve this problem, which means we solve the penalized problem:

$$\underset{\boldsymbol{\beta}, \mathbf{u}}{\operatorname{argmin}} \quad t\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + t\lambda\sum_{i=1}^m u_i + \Phi(\boldsymbol{\beta}, \mathbf{u}), \tag{9}$$

where $\Phi(\boldsymbol{\beta}, \mathbf{u}) = -\sum_{i=1}^{m} \log(u_i^2 - \beta_i^2)$ is the penalty function for the constraints of (7) and t is the penalty prameter. Solution of problem (9) converges to (2) if we follow the central path as t varies from 0 to ∞.

Instead of using the KKT conditions of (6), we utilize the KKT conditions of (9), which are

$$2t\mathbf{X}^\top(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}) + \begin{bmatrix} 2\beta_1/(u_1^2 - \beta_1^2), \\ \vdots \\ 2\beta_m/(u_m^2 - \beta_m^2) \end{bmatrix} = \mathbf{0}, \tag{10}$$

$$t\lambda\mathbf{1} - \begin{bmatrix} 2u_1/(u_1^2 - \beta_1^2) \\ \vdots \\ 2u_m/(u_m^2 - \beta_m^2) \end{bmatrix} = \mathbf{0}. \tag{11}$$

Let us denote this KKT conditions as $g(\mathbf{y}, \mathbf{X}, \boldsymbol{\beta}, \mathbf{u}) = \mathbf{0}$. According to the implicit function theorem, the derivative of $\boldsymbol{\beta}$ with respect to $\mathbf{y}$ can be computed as the first m rows of

$$-\mathbf{J}^{-1}\frac{\partial g}{\partial \mathbf{y}}, \tag{12}$$

where $\mathbf{J} = [\frac{\partial g}{\partial \boldsymbol{\beta}}, \frac{\partial g}{\partial \mathbf{u}}]$ is the Jacobian matrix of $g(\mathbf{y}, \mathbf{X}, \boldsymbol{\beta}, \mathbf{u})$ with respect to $\boldsymbol{\beta}$ and $\mathbf{u}$,

$$\frac{\partial g}{\partial \mathbf{y}} = \begin{bmatrix} -2t\mathbf{X}^\top \\ \mathbf{0} \end{bmatrix}, \tag{13}$$

$$\frac{\partial g}{\partial \boldsymbol{\beta}} = \begin{bmatrix} 2t\mathbf{X}^\top\mathbf{X} + \mathbf{D}_1 \\ \mathbf{D}_2 \end{bmatrix}, \tag{14}$$

$$\frac{\partial g}{\partial \mathbf{u}} = \begin{bmatrix} \mathbf{D}_2 \\ \mathbf{D}_1 \end{bmatrix}, \tag{15}$$

with

$$\mathbf{D}_1 = \operatorname{diag}\big(2(u_1^2 + \beta_1^2)/(u_1^2 - \beta_1^2)^2, \ldots, 2(u_m^2 + \beta_m^2)/(u_m^2 - \beta_m^2)^2\big),$$
$$\mathbf{D}_2 = \operatorname{diag}\big(-4u_1\beta_1/(u_1^2 - \beta_1^2)^2, \ldots, -4u_m\beta_m/(u_m^2 - \beta_m^2)^2\big).$$

Also, according to (10), (11), and the implicit function theorem, the derivative of $\boldsymbol{\beta}$ with respect to $\mathbf{X}$ can be calculated as the first m rows of

$$-\mathbf{J}^{-1}\frac{\partial g}{\partial \mathbf{X}}, \tag{16}$$

where $\frac{\partial g}{\partial \mathbf{X}} \in \mathbb{R}^{2m \times (mn)}$ and

$$\frac{\partial g_i}{\partial X_{kl}} = \begin{cases} 2t\delta_{li}(\mathbf{X}\boldsymbol{\beta} - \mathbf{y})_k + 2tX_{ki}\beta_l, & \text{if } i \le m \\ 0, & \text{if } i > m \end{cases} \tag{17}$$

with δ_{li} being the Kronecker delta function

$$\delta_{li} = \begin{cases} 1, & \text{if } i = l \\ 0, & \text{if } i \ne l \end{cases} \tag{18}$$

and $(\mathbf{X}\boldsymbol{\beta} - \mathbf{y})_k$ being the kth element of the vector $(\mathbf{X}\boldsymbol{\beta} - \mathbf{y})$.

To calculate the gradient of $\boldsymbol{\beta}$ with respect to $\mathbf{y}$ and $\mathbf{X}$, we first need to find the inverse of the Jacobian matrix. The Jacobian matrix is a 2×2 block matrix,

$$\mathbf{J} = \begin{bmatrix} 2t\mathbf{X}^\top\mathbf{X} + \mathbf{D}_1 & \mathbf{D}_2 \\ \mathbf{D}_2 & \mathbf{D}_1 \end{bmatrix}.$$

This block structure makes the inverse of $\mathbf{J}$ admit a simple form [13]:

$$\mathbf{J}^{-1} = \begin{bmatrix} (2t\mathbf{X}^\top\mathbf{X} + 2\mathbf{D})^{-1} & * \\ * & * \end{bmatrix}, \tag{19}$$

where $\mathbf{D} = \operatorname{diag}\big(1/(u_1^2 + \beta_1^2), \ldots, 1/(u_m^2 + \beta_m^2)\big)$. Since the gradient of $\boldsymbol{\beta}$ with respect $\mathbf{y}$ and $\mathbf{X}$ only depends on the first m rows of (12) and (16) respectively and the elements from $m+1$ to $2m$ are zero both for $\partial g/\partial \mathbf{y}$ and $\partial g/\partial \mathbf{X}$, we omit unused elements in (19) in later computation. With this explicit expression of the Jacobian matrix, we have

$$\frac{\partial \boldsymbol{\beta}}{\partial \mathbf{y}} = \big(\mathbf{X}^\top\mathbf{X} + 1/t \cdot \mathbf{D}\big)^{-1}\mathbf{X}^\top, \tag{20}$$

and

$$\frac{\partial \boldsymbol{\beta}}{\partial X_{kl}} = \left[\frac{\partial \beta_1}{\partial X_{kl}}, \frac{\partial \beta_2}{\partial X_{kl}}, \ldots, \frac{\partial \beta_m}{\partial X_{kl}}\right]^\top, \tag{21}$$

with

$$\frac{\partial \beta_i}{\partial X_{kl}} = \sum_j -(\mathbf{X}^\top\mathbf{X} + 1/t \cdot \mathbf{D})^{-1}_{ij}\frac{\partial g_j}{\partial X_{kl}}.$$

Let us denote the objective of (4) as $f(\mathbf{y}, \mathbf{X})$. Using the chain rule, we have the gradient of f with respect to $\mathbf{y}$ and $\mathbf{X}$:

$$\nabla_{\mathbf{y}} f(\mathbf{y}, \mathbf{X}) = \left(\frac{\partial \boldsymbol{\beta}}{\partial \mathbf{y}}\right)^\top \mathbf{H}(\boldsymbol{\beta} - \boldsymbol{\nu})\Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} \tag{22}$$

and

$$\frac{\partial f(\mathbf{y}, \mathbf{X})}{\partial X_{kl}} = (\boldsymbol{\beta} - \boldsymbol{\nu})^\top \mathbf{H}\frac{\partial \boldsymbol{\beta}}{\partial X_{kl}}\Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}}. \tag{23}$$

Now, we know the gradients of our objective function (4). With the help of this gradient information, we can use a variety of gradient based optimization methods. Since our problem is a constrained optimization problem, we resort to the projected gradient descent method. We have summarized it in Algorithm 1. The main concept of the projected gradient descent algorithm is that we first take a gradient step, project it onto our feasible set, and we take an α_t step along the projected point. In this algorithm, $\text{Porj}_{\mathcal{C}_y}(\cdot)$ and $\text{Proj}_{\mathcal{C}_x}(\cdot)$ represent the projection operator which projects a point onto the feasible set $\mathcal{C}_y$ and $\mathcal{C}_x$, respectively. $\mathcal{C}_y$ and $\mathcal{C}_x$ are ℓ_p balls with radius η_y and η_x respectively. In the following, we will discuss the expressions of the projection onto three commonly

Algorithm 1 The Projected Gradient Descent Algorithm

Input: data set $\{(y_0^i, \mathbf{x}_0^i)\}_{i=1}^n$, trade off parameter λ in (1), energy budget η_y, η_x, ℓ_p norm, and step size parameter α_t.
solve $\boldsymbol{\beta}_0$ via (1), set up feature set S, E, U and its corresponding parameters $\mathbf{s}$, $\mathbf{e}$, $\boldsymbol{\mu}$; use those parameters define our objective function $f(\mathbf{y}, \mathbf{X})$ in (4).
Initialize set the number of iterations $t = 0$ and $\mathbf{y}_t = \mathbf{y}_0$, $\mathbf{X}_t = \mathbf{X}_0$.
Do
solve $\hat{\boldsymbol{\beta}}$ according to (9),
compute the gradients: $\nabla_{\mathbf{y}} f(\mathbf{y}_t, \mathbf{X}_t)$ according to (22) and $\nabla_{\mathbf{X}} f(\mathbf{y}_t, \mathbf{X}_t)$ according to (23),
update:
$\mathbf{y}_{t+1} = (1-\alpha_t)\mathbf{y}_t + \alpha_t \cdot \text{Proj}_{\mathcal{C}_y}\big(\mathbf{y}_t - \nabla_{\mathbf{y}} f(\mathbf{y}_t, \mathbf{X}_t)\big)$,
update:
$\mathbf{X}_{t+1} = (1-\alpha_t)\mathbf{X}_t + \alpha_t \cdot \text{Proj}_{\mathcal{C}_x}\big(\mathbf{X}_t - \nabla_{\mathbf{X}} f(\mathbf{y}_t, \mathbf{X}_t)\big)$,
set $t = t + 1$,
While convergence conditions are not meet.
Output: $\mathbf{y}_t, \mathbf{X}_t$.

used ℓ_p norm balls, where $p = 1, 2, \infty$ with radius of the norm ball being η and its center being the origin.
Case 1: Project onto the ℓ_1 norm ball. It involves solving the following convex problem

$$\begin{aligned} \min_{\mathbf{z}} \quad & \|\mathbf{z} - \mathbf{x}\|_2 \\ \text{s.t.} \quad & \|\mathbf{x}\|_{\ell_1} \leq \eta, \end{aligned}$$

which can be efficiently solved via its dual with complexity $\mathcal{O}(m)$ [14].
Case 2: Project onto the ℓ_2 norm ball. In this case, we have a very simple closed form solution

$$\text{Proj}(\mathbf{x}) = \mathbf{x}./\max\{1, \|\mathbf{x}\|_2\}, \tag{24}$$

where '$./$' denotes the element-wise division.
Case 3: Project onto the ℓ_∞ norm ball. In this case, we also have a very simple close-form solution:

$$\text{Proj}(\mathbf{x}) = \mathbf{z}, \tag{25}$$

where $\mathbf{z} = [z_1, \ldots, z_m]^\top$ and

$$z_i = \begin{cases} -1, & \text{if } x_i \leq -1, \\ x_i, & \text{if } |x_i| < 1, \\ 1, & \text{if } x_i \geq 1. \end{cases}$$

With these expressions of the projection, we can easily obtain the expressions of $\text{Proj}_{\mathcal{C}_y}(\cdot)$ and $\text{Proj}_{\mathcal{C}_x}(\cdot)$ by simply doing geometric translation.

4. NUMERICAL EXAMPLES

In this section, we use several experiments to demonstrate the results obtained in this paper.

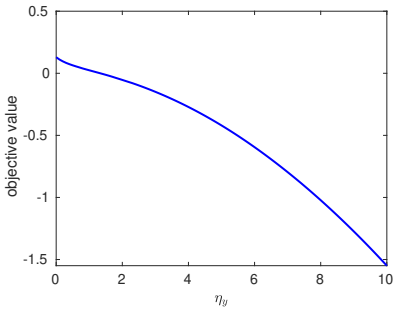

Fig. 1. The objective value changes with the energy budget.

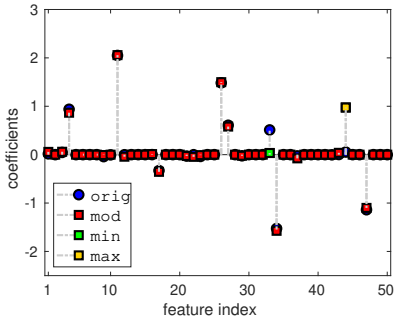

Fig. 2. The original regression coefficients and the regression coefficients after our attacks.

In the first numerical example, we test our algorithm on a synthetic data set. Firstly, we generate a 30×50 feature matrix $\mathbf{X}_0$. Each entry of the feature matrix is i.i.d. generated from a standard normal distribution. Then, we generate the response values, $\mathbf{y}_0$, through the model $\mathbf{y}_0 = \mathbf{X}_0\mathbf{v} + \mathbf{n}$, where $\mathbf{v}$ is the sparse vector in which only ten randomly selected positions are non-zero and each of the non-zero entry is i.i.d. drawn from the standard normal distribution; $\mathbf{n}$ is the noise vector where each entry is i.i.d. generated according to a normal distribution with zero mean and 0.1 variance. Then, we set the LASSO trade-off parameter $\lambda = 2$ and use (7) to estimate the regression coefficients $\boldsymbol{\beta}_0$. We randomly select one regression coefficient as the desired coefficient to be boosted and another as our coefficient to be suppressed. Also, we set the suppress parameter $s_i = 1$ for $i \in S$, set boost parameter $e_i = -1$ for $i \in E$, and set the unchanged parameter $\mu_i = 5$ for $i \in U$. We set the stepsize parameter $\alpha_t = 2/\sqrt{t}$ in Algorithm 1.

In the first experiment, we set $\eta_x = 0$, which means that we do not modify the feature matrix, and impose ℓ_2 norm constraint on the modification of the response values. Then, we vary the energy budget, η_y, to see how energy budget influence our objective value. Fig. 1 illustrates that the objective value decreases as the energy budget increases. This is expected because larger energy budget provides larger feasible region, and thus lower objective value. Fig. 2 demonstrates the recovered regression coefficients when $\eta_y = 5$ along with the original regression coefficients. In this figure, 'orig' de-

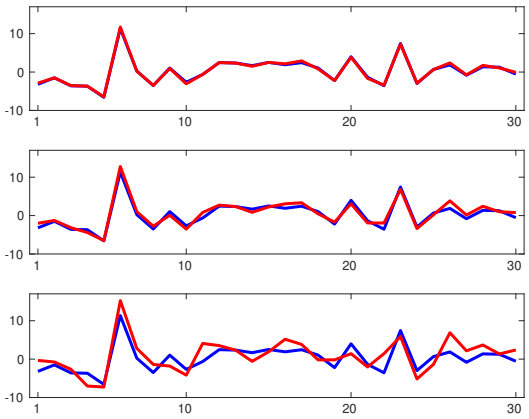

Fig. 3. The blue line demonstrates the original response values and the red line is the modified response values with different attack constraints. From top to bottom are the modified response values with ℓ_1, ℓ_2, and ℓ_∞ norm constraints, respectively.

notes the original regression coefficients, 'mod' represents the regression coefficients after our attack, 'min' is the regression coefficient we want to suppress, and 'max' denotes the regression coefficient we want to promote. As the figure demonstrated, we have successfully suppressed and promoted the corresponding coefficients while keeping other regression coefficients almost unchanged.

In the second experiment, we also attack the response values. We fix the energy budget $\eta_y = 5$ and test different ℓ_p norm constraints on the modification of the response values as $p = 1, 2, \infty$. Fig. 3 shows the original and modified response values under different ℓ_p norm constraints. The x-axis denotes the index of each response value and the y-axis denotes the value of the response vector. As the figure illustrated, the ℓ_1 norm constraint provides the smallest modification on the response values and the ℓ_∞ norm constraint provides the most significant modification, which result in objective value 0.0095 with the ℓ_1 norm constraint, objective value -0.4199 with the ℓ_2 norm constraint, and objective value -2.8813 with the ℓ_∞ norm constraint. That is because with the same radius, ℓ_1 norm ball is contained in the ℓ_2 norm ball and ℓ_2 norm ball belongs to the ℓ_∞ norm ball.

In the third experiment, we compare the modification on the response values and on the feature matrix with the ℓ_1 constraints. First, we only attack the response values with $\eta_y = 5$, which results in objective value 0.0095. Second, we only attack the feature matrix with the same energy budget $\eta_x = 5$, which results in objective value -0.0969. Finally, we attack both the response values and the feature matrix with $\eta_y = 5$ and $\eta_x = 5$, which results in objective value -0.2291. These results indicate that both the modifications of the response values and feature matrix are effective. However, modification of the feature matrix seems more efficient than that of the

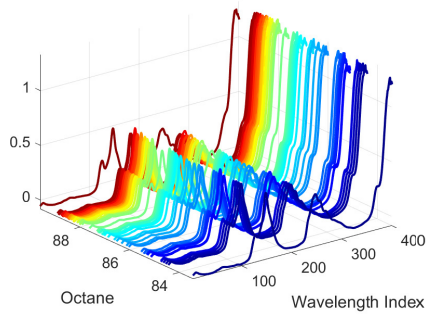

Fig. 4. Overview of the octane data set.

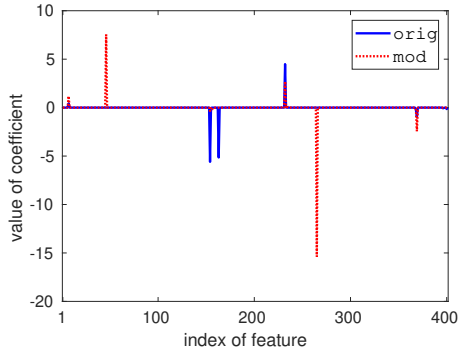

Fig. 5. The regression coefficients before and after our attack.

response values.

We now test our attack strategy using real datasets. In this task, we use the spectral intensity of the gasoline to predict its octane rating [15]. It consists of 60 samples of gasoline at 401 wavelength and their octane ratings. Fig. 4 provides an overview of the data samples. From the figure we can see that there are very high correlations among different wavelengths. Hence, if we use the ordinary linear regression method, it will have large errors. Thus, we use the LASSO method to complete the regression task. We randomly choose 80% of the data samples as our training data and the rest as our test data. We do cross-validation on the training data to decide the trade-off parameter in LASSO, and it gives $\lambda = 0.5$. Using this parameter, we compute the regression coefficients. Using this regression coefficients on the test data set, we have r-squared value 0.979. The blue line in Fig. 5 shows the original regression coefficient. From this figure, we can see that there are several important features.

In the next step, we modify the response values and the feature matrix with the energy budget $\eta_y = 5$ and $\eta_x = 5$ to suppress the 154th and 163th regression coefficients, unchange the 232th and 369th regression coefficients, and promote the rest of the regression coefficients. In our algorithm, we set $s_i = 1$ for $i \in S$, $e_i = -1$ for $i \in E$, $\mu_i = 50$ for $i \in U$, and stepsize parameter $\alpha_t = 1/t$. The red-dashed line in Fig. 5 shows the regression coefficients after our attacks.

As is shown, we successfully promoted two regression coefficients which were zero-valued before attack. We also suppressed the 154th and 163th regression coefficients and made the 232th and 369th regression coefficients change very little. Using this regression coefficients on the test data set, we got the r-squared value 0.694. Hence, by changing the response values and the feature matrix, we can easily make the system choose the wrong features.

5. CONCLUSION

In this paper, we have investigated the adversarial robustness of the LASSO based feature selection algorithm. We have provided an algorithm to mitigate the non-differentiability of the ℓ_1 norm in the feature selection method. The numerical examples both on the synthetic data and real data have shown that feature selection based on LASSO is very vulnerable to the adversarial attacks. It is of interest to study the defense strategy against this kind of attacks in the future.

6. REFERENCES

[1] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[2] S. Ma, X. Song, and J. Huang, "Supervised group lasso with applications to microarray data analysis," *BMC Bioinformatics*, vol. 8, no. 1, p. 60, Dec. 2007.

[3] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[4] D. Yang and W. Bao, "Group lasso-based band selection for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 12, pp. 2438–2442, Nov. 2017.

[5] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, Jun. 2018.

[6] F. Li, L. Lai, and S. Cui, "On the adversarial robustness of subspace learning," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1470–1483, Mar. 2020.

[7] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287–1289, Mar. 2019.

[8] J. Jeong and C. Kim, "Effect of outliers on the variable selection by the regularized regression," *Communications for Statistical Applications and Methods*, vol. 25, no. 2, pp. 235–243, Mar. 2018.

[9] P.-L. Loh and M. J. Wainwright, "High-dimensional regression with noisy and missing data: Provable guarantees with non-convexity," in *Advances in Neural Information Processing Systems*, Granada, Spain, Dec. 2011, pp. 2726–2734.

[10] A. L. Dontchev and R. T. Rockafellar, "Implicit functions and solution mappings," *Springer Monographs in Mathematics. Springer*, vol. 208, Feb. 2009.

[11] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Proc. AAAI Conference on Artificial Intelligence*, Austin Texas, Jan. 2015, pp. 2871–2877.

[12] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale ℓ_1-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, Dec. 2007.

[13] T.-T. Lu and S.-H. Shiou, "Inverses of $2\times$ 2 block matrices," *Computers & Mathematics with Applications*, vol. 43, no. 1-2, pp. 119–129, 2002.

[14] L. Condat, "Fast projection onto the simplex and the ℓ_1 ball," *Mathematical Programming*, vol. 158, no. 1-2, pp. 575–585, Sep. 2015.

[15] J. H. Kalivas, "Two data sets of near infrared spectra," *Chemometrics and Intelligent Laboratory Systems*, vol. 37, no. 2, pp. 255–259, Jun. 1997.