Attackability Characterization of Adversarial Evasion Attack on Discrete Data

ABSTRACT

Evasion attack on discrete data is a challenging, while practically interesting research topic. It is intrinsically an NP-hard combinatorial optimization problem. Characterizing the conditions guaranteeing the solvability of an evasion attack task thus becomes the key to understand the adversarial threat and to design efficient attack methods. Our study is inspired by weak submodularity theory. We characterize attackability of a targeted classifier in evasion attack by bridging the attackability measurement and the regularity of the targeted classifier. Based on our attackability analysis, we propose a computationally efficient orthogonal matching pursuit-guided based attack method for evasion attack on discrete data. It provides provably computational efficiency and attack performances. Substantial experimental results on real-world datasets validate the proposed attackability conditions and the effectiveness of the proposed attack method.

1 INTRODUCTION

Despite fruitful progress of evasion attack on continuous data, such as images and videos [2, 3, 5, 6, 19, 21, 33, 34], how to design adversarial examples for discrete data remains a rarely investigated, but important research problem. Discrete data pervasively exists in real-world applications and analytic practitioners have been designed for financial fraud detection, spam email detection, gene analysis in bioinformatics and malware dynamic analysis, etc. Many of these applications are safety-critical, while eager to deploy machine learning technologies. Investigating the adversarial vulnerability on discrete data thus becomes a must in these industrial practices.

Real-world discrete data are usually located in a space spanned by physically meaningful dimensions. For example, the presence or absence of a keyword in a document, a gene in a genome or a function call of an executable file. The discrete attributes of an object form a highly structured and semantic description of the object. Compared to the low-level and continuous measurements like pixel intensities, discrete attributes and their combination encode high-level qualitative concepts. Pioneering work of this topic, such as evasion attack against spam filtering and NLP classifiers [11, 14, 18, 23, 25, 29, 31, 32, 34–36], depends heavily on the prior knowledge about feature extraction and domain-specific rules. In general, adversarial attacks on discrete data raise the following fundamental challenges to the research community.

First of all, designing adversarial modifications of discrete data is intrinsically **a combinatorial optimization task**, which is NP-hard and generally intractable. Similar to in the continuous domain, the solvability of an evasion attack task on discrete data (a.k.a. **attackability**) relates closely to the characteristics of the attack objective function and the targeted classifier. The key questions about characterizing the attackability are thus:

- **What conditions does the attack objective need to meet to define a tractable attack problem?**
- **Can we provide provably attack performances?**

Both questions are open according to the current research progress. Given the discontinuity nature of discrete data, modifying any of the attributes might cause a big leap in the discrete feature space. Gradient-based attack methods, though pervasively used for continuous data, cannot be applied directly. Classically, combinatorial optimization problems are [28] relaxed to continuous linear or quadratic programming tasks. However, such solutions require a simple form of the targeted classifier, like linear models. They become infeasible when a highly non-linear deep neural network based classifier is used. Recent works on graph poisoning attack [1, 9, 38] and evasion attack against NLP classifiers [24] use greedy search to solve the attack problem. Notably, both branches of research efforts explicitly or implicitly formulate evasion attack on discrete data as a problem of submodular function maximization. Benefited from submodularity of the attack objective, the greedy search based attack methods in these works enjoy strong performance guarantees. However, it is still unclear how to evaluate the attackability of a general evasion attack. Furthermore, enforcing strict submodularity limits the choice of the targeted classifier and/or harm the usability of the targeted classifier by introducing artifacts into the classifier.

Secondly, it is difficult to find out the physical meaning of the adversarial noise of continuous data. Especially for images, the intensity change of individual pixels is not necessarily associated with any meaningful characteristics of the visual contents. In contrast, a change of discrete attributes indicates a drift in the concept space spanned by the discrete attributes and implies the sensitivity of each attribute for the classification task. Therefore, beyond delivering a successful evasion from the targeted classifier, we expect to reveal the answer to the question, "**Can we associate the adversarial modification of the discrete attributes with any characteristics of the classification task, e.g. sensitivity of the objects?**"

We echo the challenges by providing attackability guarantees of white-box evasion attack against non-linear classifiers, such as deep neural nets, on discrete data. We aim at flipping the classification output while preserving the integrity of the data as much as possible. Notably, we save for the future to address the issue of maintaining the functionality of data during the attack, e.g., preserving the applicability of malware or readability of texts after adversarial perturbations. On one hand, defining the rules to preserve the data functionality depends heavily on domain-specific knowledge. On the other hand, it doesn't change the combinatorial optimization nature of the attack problem. Our attackability study can still apply by integrating a-priori functionality-preserving rules of adversarial attribute modifications.

We summarize our contributions as follows:

- We cast the evasion attack problem on discrete data as a set function based optimization task. We show that the attack problem is solvable if the targeted classifier follows certain regularity constraints. Benefited from these constraints, the objective function of the attack problem enjoys weak submodularity, which makes the generally NP-hard problem feasible to solve approximately with polynomial complexity by greedy search based methods. We instantiate the attackability study on recurrent neural network (RNN) based classifiers to demonstrate how to evaluate the attackability given a concrete classifier.

- We reveal that the approximation quality of the greedy search based solution depends on the regularity of the targeted classifier, which determines the submodularity ratio of the attack objectives. It indicates a balance between the modeling flexibility of the classifier and its attackability. Fewer regularity constraints enable broader choices of the classifiers' architectures. However, it makes the attack problem more difficult to solve simultaneously. Furthermore, to address the computational bottleneck of the primitive greedy search, we propose an orthogonal matching pursuit based greedy search to improve the efficiency of the solution. The proposed method provides a bounded estimator of the marginal gain of the attack objective with respect to each attribute. It only traverses the attributes of the largest marginal gain, in order to fasten the greedy search while providing a provable approximation quality.
- In addition, we explore the association between the evasion attack and sensitivity analysis of the attributes. We extend the evasion attack to flag commonly useful attributes to attack different instances. Despite a lack of theoretical explanation, we observe an interesting consistency between the most sensitive attributes and the most commonly selected attributes to attack.

2 RELATED WORK

Most research effort of evasion attack with discrete input studies attacking NLP classifiers [11, 14, 18, 23, 25, 29, 31, 32, 34–36], where the attack is conducted by modifying or replacing individual words or sentences, following heuristic rules, such as replacing words with synonyms or sentences with similar alternatives either defined manually or found by grouping entities with word-embedding techniques. These works borrow directly the key idea of the evasion attack in continuous domain. They choose the discrete word/sentence transformations, which are mostly aligned with the gradient vector of the attack objective. Similarly [18] proposes to conduct Carlini's gradient-guided evasion attack [5] on continuous word-embedding space first. It generates a target embedding representation of the text. After that, the method searches for the entities carrying the most similar embedding vector with the targeted embedding representation. These entities are then used to replace the corresponding words/sentences in the original text. These methods have no guarantee of delivering a successful attack. On one hand, the word embedding space could be non-smooth. The embedding vectors of synonyms could be so different that they don't appear as neighbors in the embedding space. On the other hand, nevertheless, the gradient vector of the attack objective approximates well only local variations around the input embeddings. Modifying even one word/phrase could violate this assumption in the embedding space. Therefore, the attack methods applied popularly on continuous data can't be used directly for discrete data.

Submodular function maximization has been introduced as an efficient solution to combinatorial optimization since 1970's [16, 17]. Early stage works [16, 17, 26] focuses on submodular function maximization with uniform matroid constraint, a.k.a. cardinality constraint. Calinescu et al further extend its applicability to more general matroid constraints. As pointed in [17], the theory of submodular functions offers a decent solution to the NP-hard combinatorial optimization problem. For monotone submodular function maximization, primitive greedy search with polynomial complexity can achieve an approximation ratio of $1 - 1/e$. Furthermore, [7, 15, 27] prove that non-monotone submodular functions subject to general and uniform matroid constraints can still enjoy an approximation ratio of 0.491 and 0.478. In evasion attack on discrete data, the use of submodular functions has been witnessed in [1, 9, 24, 38]. [1, 9, 38] propose to add or remove edges of a given graph, in order to conduct evasion attack against graph embedding based node classifier. Though it is not explicitly claimed, the attack objective function, defined as a sum of smallest eigenvalues of the adjacency matrix of the graph, is intrinsically a submodular function. Enjoying strict submodularity, the proposed graph poisoning method selects edges to flip with greedy search efficiently to evade from the graph embedding based classifier successfully. [24] introduces additional positiveness constraint over the CNN and RNN based classifier's parameters. The resultant classifier is proven to be submodular. A greedy search based attack method is then used to select words/sentences and replace them with feasible candidates.

Despite of the effectiveness, the use of submodular function is limited, especially in adversarial learning research. It has been observed that there are many practical settings inducing functions that deviate from submodularity. Greedy search based methods unfortunately perform arbitrarily poorly in general when the target function even slightly deviates from strict submodularity. Besides, enforcing extra constraints over the targeted classifiers of adversarial attack introduces artefacts into the classifiers' architectures and limits the model choice. It can cause degradation of classification accuracy, thus reduces the usability of the classifier. Therefore, such attack setting is still far from the adversarial threat witnessed in real-world practices. Recent research efforts break the hurdle by introducing γ-weakly submodular functions [8, 10]. Submodularity ratio γ measures the distance of the function from being strictly submodular. The standard greedy algorithm achieves a graceful approximation ratio of $1 - e^{-\gamma}$ for the problem of maximizing such functions subject to a cardinality constraint.

It is a brand new research problem to apply the theory of weak submodularity in adversarial attack on discrete data. Intuitively, weak submodularity can broaden the choice of feasible attacks, which helps to formulate more realistic scenarios of the adversarial threat. Nevertheless, it is still unclear how to design a weakly submodular objective function for the attack. The celebrated work by [12] sets up an equivalence between the smoothness and strongly concavity of log-likelihood functions and the weak submodularity of the objective functions in a feature subset selection problem. Inspired by this work, we reveal that the evasion attack can be formulated as a problem of weakly submodular function maximization, if the targeted classifier follows even less tight regularity constraints. We argue that enforcing strong concavity over the classifier is not obliged to guarantee the weak submodularity. Enjoying the framework of weak submodularity, we can characterize the attackability of evasion attack given the regularity measurement of the targeted classifier.

3 ATTACKABILITY ANALYSIS

We use $\mathbf{x} = \{x_1, x_2, x_3, ..., x_n\}$ to represent an instance with n discrete features. Each x_i is a cell of m ($m \geq 1$) categorical attributes. For example, an x_i can be a code segment in a malware file associated with one unique function or a medical examination output linked to the m different biological characteristics of human body. In practice, we cast each categorical attribute to a D-dimensional pre-trained embedding vector, e.g., $\mathbf{e}^j \in R^D$, $j = 1, 2, ..., m$.

To represent instance $\mathbf{x}$ by the embedding vectors of its attribute values, we introduce binary variables $\mathbf{b} = \{b_i^j\}$, i=1, 2, ..., n, j=1, 2, ..., m, where $b_i^j = 1$ when the j-th attribute value is selected for x_i, and $b_i^j = 0$ otherwise. One instance $\mathbf{x}$ encoded by the embedding vectors can then be represented as an R^{n*m*D} tensor with $\mathbf{x}_{\{i,j,:\}} = b_i^j e_i^j$.

We use $\hat{\mathbf{b}} = \{\hat{b}_i^j\}$ as the adversarially tuned variable modified from $\mathbf{b}$. If no modification, $\hat{b}_i^j = b_i^j$. Otherwise, $\hat{b}_i^j \neq b_i^j$. Depending on the type of attacks to implement, e.g., *insertion*, *deletion* or *substitution*, $\hat{b}_i^j$ can have different values. *Insertion* is to let $\hat{b}_i^j = 1$, given $b_i^j = 0$, $\forall j = 1, ..., m$. *Deletion* is to let $\hat{b}_i^j = 0$, given $b_i^j = 1$. *Substitution* is to let $\hat{b}_i^j = 1, \hat{b}_i^{j'} = 0$, given $b_i^j = 0, b_i^{j'} = 1, j \neq j'$. A modified instance $\hat{\mathbf{x}}$ can thus be written as $\hat{\mathbf{x}}_{\{i,j,:\}} = \hat{b}_i^j e_i^j$.

Let $\mathbf{y}$ be the target class label of the evasion attack. The goal is to make $f_y(\hat{\mathbf{x}})$ deviate from $f_y(\mathbf{x}) = 0$. In other words, we aim at maximizing $f_y(\hat{\mathbf{x}})$, so that $\mathbf{x}$ is modified to get $\mathbf{y}$ assigned to it. The evasion attack task can then be formulated as a process of set function optimization, defined as

$$\begin{aligned} S^* &= \underset{|S| \leq K}{\arg\max}\ g(S) \\ \text{where} \quad g(S) &= \max_{l \subset S}\ f_y(\hat{\mathbf{x}}), \quad l = \mathit{diff}(\mathbf{b}, \hat{\mathbf{b}}) \end{aligned} \tag{1}$$

where $|S|$ is the cardinality of set S. Function $g(S)$ is a set function measuring the maximum extent to which the attacks in S can change the classification result. The *diff* function reports the set of the indices where $\mathbf{b}$ and $\hat{\mathbf{b}}$ are different. In other words, l denotes the set of modification to make for attacking $\mathbf{x}$. To preserve data integrity, the modification made on $\hat{\mathbf{b}}$ should be as small as possible ($|S| \leq K$).

Obviously $g(S)$ is a non-decreasing monotone set function as $g(S) \leq g(T)$ if $S \subseteq T$. By solving Eq. (1), we pursue to find the optimal set of adversarial discrete attribute modification, including adding, deleting and replacing values of discrete attributes in the original data instance.

3.1 Weak Submodularity Based Solvability Conditions

To solve the attack problem in Eq.(1), we first evaluate its solvability conditions based on weak submodularity [10]: we require the attack objective to be **weakly submodular** and show it can be solved approximately with polynomial complexity given this constraint. We introduce **submodularity ratio** [10] to measure the attackability of the evasion attack problem defined by Eq.(1). For a weakly-submodular attack objective, the higher the submodularity ratio is, the better attack solution we can obtain via maximizing the attack objective, while the less constraints we need to enforce over the classification model.

We elaborate our solution by first introducing the definition of weak submodularity and submodularity ratio.

Definition 1. *Given a set cardinality threshold $k \geq 1$, the submodularity ratio γ_k of a set function $g(.)$ w.r.t. a set H:,*

$$\gamma_k = \min_{S \subseteq H, A: |A| \leq k, S \cap A = \emptyset} \frac{\sum_{a \in A} g(S \cup a) - g(S)}{g(S \cup A) - g(S)} \tag{2}$$

If $\gamma = \min_k \gamma_k < 1$, the set function $g(S)$ is weakly submodular. Otherwise, $g(S)$ is submodular when $\gamma \geq 1$ for any S.

We define the regularization constraint over the classification function f_y by extending *Restricted Strong Convexity (RSC)* in Theorem.1 of [12] to apply to non-concave functions. It guarantees further **weak submodularity** of the attack objective of broader classes of function forms.

Definition 2. *Let $\Omega = (x, y), x, y \in R^n$ and $\ell: R^n \rightarrow R$ be a continuously differentiable function. A function f is (m_Ω, M_Ω)-bounded on Ω, if for any $(x, y) \in \Omega$, $m_\Omega \in R$ and $M_\Omega \in R^+$:*

$$\begin{aligned} f(y) - f(x) - \langle \nabla f(x), y - x \rangle &\geq -\frac{M_\Omega}{2} \|y - x\|_2^2 \\ f(y) - f(x) - \langle \nabla f(x), y - x \rangle &\leq -\frac{m_\Omega}{2} \|y - x\|_2^2 \end{aligned} \tag{3}$$

Remark 1. *If $m_\Omega > 0$, f is then strongly concave. If $m_\Omega \leq 0$, f may violate the concavity constraint and presents linearity or convexity. It is easy to find that, larger M_Ω and smaller m_Ω relax the bounded constraint enforced to the function f, which allows f to choose among a broader class of functions.*

We then present the regularity constraint enforced on f_y to guarantee weak submodularity of the attack objective in Theorem 1. We then link the lower bound of the submodularity ratio (how "weakly submodular" the attack objective is) to the approximation quality of Eq. (1) in Theorem 2. Both theorems form the base of the attackability study of the evasion attack on discrete data.

Theorem 1. *Let b as the unchanged original binary indicator defined in Eq.1. Let $\Omega_k = \{(\hat{b}, \hat{b'}) : |\mathit{diff}(b, \hat{b})| \leq k, |\mathit{diff}(b, \hat{b'})| \leq k, |\mathit{diff}(\hat{b}, \hat{b'})| \leq k\}$, where $\hat{b}$ and $\hat{b'}$ denote two sets of selected discrete attributes to be modified adversarially. If the classifier f_y is $(m_{\Omega_k}, M_{\Omega_k})$-regularized on Ω_k, the $g(S)$ defined by Eq.1 is weakly submodular. Its submodularity ratio γ_k on Ω_k is bounded from below:*

$$\begin{aligned} \gamma_k &\geq \frac{1}{2\psi_k M_{\Omega_k}} \\ \psi_k &= 1 + \frac{k^2 |m_{\Omega_k}|}{2\|\nabla f_y(b)_s\|_2^2}, \quad \mathit{If}\ m_{\Omega_k} \leq 0 \\ \psi_k &= \frac{1}{2m_{\Omega_k}}, \quad \mathit{If}\ m_{\Omega_k} > 0 \end{aligned} \tag{4}$$

where $\nabla f_y(b)_v$ denotes the elements of $\nabla f_y(b)$ corresponding to the difference between the index sets l_b and $l_{b'}$, where $v = l_{b'} \setminus l_b + l_b \setminus l_{b'}$.

```
Input: The attack budget $K$, the set function $g(S)$ defined by
       Eq. (1) and the set $H = \{(i,j), i = 1...n, j = 1...m\}$ of
       all the modifiable discrete values
Output: selected support set $S_k$, with $|S_k| \leq K$; $g(S_k)$ and
        the optimal subset of $S_k$ achieving the attack goal
$S_0 \leftarrow \emptyset$
for $k = 1$ to $K$ do
    $s \leftarrow \arg\max_{j \in H/S_{k-1}} g(S_{k-1} \cup j) - g(S_{k-1})$
    $S_k \leftarrow S_{k-1} \cup \{s\}$
end
```

Algorithm 1: Forward Stepwise Greedy Search (FSGS)

According to Theorem.1, if the targeted classifier f_y has a well regularized function form as required by Eq.3, the objective function of the evasion attack defined by Eq.1 then enjoys weak submodularity with a bounded submodularity ratio. As proved in [10], such a weakly submodular attack objectivecan be maximized by incrementally changing discrete attributes. This primitive greedy search based solution is also known as *Forward Stepwise Greedy Search* (*FSGS*) [12]. More specifically, let S denote the set of modified attributes in the previous iterations. In next iteration, *FSGS* finds $s = (i, j)$ to add to S, if $\hat{b}_i^j$ is set to be different from b_i^j to achieve the largest marginal gain of $g(S \cup s)$ over $g(S)$. The algorithm is represented in Algorithm.1.

This primitive greedy search method is computationally expensive, because in each iteration, *FSGS* evaluates all the unchanged values to select the next candidate to modify. That's to say, s is selected from all the unchanged values to see which makes maximum improvement of $g(S \cup s)$ over $g(S)$. In each iteration, *FSGS* needs $O((\sum_{i=1}^{k-1} C_k^i)(|H| - \kappa))$ objective function evaluations to search the optimal candidate attribute s, where $|H|$ denotes the size of all feasible candidate attributes to change and $\kappa = |S_k|$. As seen, the computational cost of *FSGS* becomes quickly expensive with increasingly larger $|S_k|$. It is the major bottleneck of *FSGS*. We discuss how to address this issue and deliver faster while successful attack later in Section.4.

Theorem 2. *[**Theorem 3** in [12]] Let the evasion attack problem defined by Eq.(1) be with the classification function f_y that is $(m_{\Omega_k}, M_{\Omega_k})$-bounded. Let S_k be the set of the values selected by FSGS and S_k^* be the underlying optimal value set following the support size constraint. The corresponding attack objective values reached by S_k and S_k^* are g^{FSGS} and g^{OPT}, respectively. Then g^{FSGS} is bounded:*

$$g^{FSGS} \geq (1 - e^{-\gamma_{S_k}})g^{OPT} \tag{5}$$

where γ_{S_k} is the submodularity ratio of $g(S)$ defined on the selected set S_k. Especially, if $g(S)$ is submodular, the lower bound gives as:

$$g^{FSGS} \geq (1 - e^{-1})g^{OPT} \tag{6}$$

We can find explicitly the relation between the regularity of the classification f_y and solvability of the attack objective as follows:

- **Submodularity Ratio v.s. Regularity of f_y**: According to Theorem 1, a classification function f_y following the bound conditions is guaranteed to be weakly submodular. A smoother function f_y (smaller M_Ω) with higher m_Ω (closer to concavity) can give higher bound of the submodularity ratio of $g(S)$. It thus makes Eq. (1) closer to submodularity. In contrary, a less regularized f_y, espeically when $m_{\Omega_k} \leq 0$, the submodularity ratio of the attack objective deteriorates significantly.
- **Solution quality v.s. Regularity of f_y**: According to Theorem 2, the higher submodularity ratio the attack objective has, the better approximation quality the solution derived reaches, compared to the underlyingly optimal solution to the attack problem. In summary, with a more regularized function f_y, it is more likely to solve the NP-hard attack problem approximately with greedy search based solution.

3.2 Attackability of RNN

We instantiate the attackability condition proposed in Theorem 1 to the case where recurrent neural network (RNN) based classifier is applied. It is worth noting that the attackability analysis is not limited to RNN. We use this case study to demonstrate how to characterize the attackability for a given classifier on discrete data.

We consider a simple but generalizable RNN model $f_y(\mathbf{x})$ with n time steps, taking input of the n discrete values in $\mathbf{x}$. The hidden state h_t of RNN by taking x_t is:

$$h_t = \phi(\alpha^T h_{t-1} + \beta(\mathbf{b}_t \odot \mathbf{E}) + \theta) \tag{7}$$

where α, β and θ are the parameters of the unit, and $\mathbf{b}_t \odot \mathbf{E} = [b_t^1 \mathbf{e}^1; b_t^2 \mathbf{e}^2; ...; b_t^k \mathbf{e}^m]$ is the representation of x_t in its attribute value embedding form. The output of the classifier is given based on the hidden unit of the last time step, as $f_y(\mathbf{x}) = \phi_y(w_y^T h_n + \theta_y)$, where w_y and θ_y are the parameters of the classification layer, ϕ and ϕ_y are the activation function of each hidden unit and the classification layer. Notably, [22] defines deep submodular functions for text classification. Nevertheless, they are not applicable in the setting of adversarial attack.

According to Theorem 1, the attackability condition of evasion attack against the general form of **RNN classifier** can be given as:

Corollary 2.1. ***Weakly Submodular Attack Objective.*** *Given the classifier as $f_y(\mathbf{x})$ and the domain $\Omega_k \in R^n * R^n$, ϕ and ϕ_y can be chosen as Sigmoid and Tahn functions. The activation function of f_y is thus non-concave and m_{Ω_k} of f_y is negative. The corresponding attack objective function $g(S)$ defined by Eq. (1) is weakly submodular.*

Remark 2. *Let ρ be the set of values selected to attack and $|\rho|$ be the attack budget. The submodularity ratio γ of $g(S)$ on $\Omega_{|\rho|}$ is bounded from below as:*

$$\begin{aligned}
&\gamma_{\Omega_{|\rho|}} \geq \frac{|\rho|^2 |\nabla f_y(\mathbf{b})_\rho|}{4(2 + |\nabla f_y(\mathbf{b})_\rho|)(2 + |\nabla f_y(\mathbf{b})_\rho| + \|\nabla f_y(\mathbf{b})_\rho\|_2^2)} \\
&(Tanh\ activation\ function) \\
&\gamma_{\Omega_{|\rho|}} \geq \frac{|\rho|^2 |\nabla f_y(\mathbf{b})_\rho|}{4(1 + |\nabla f_y(\mathbf{b})_\rho|)(1 + |\nabla f_y(\mathbf{b})_\rho| + \|\nabla f_y(\mathbf{b})_\rho\|_2^2)} \\
&(Sigmoid\ activation\ function)
\end{aligned} \tag{8}$$

where $|\nabla y_f(\mathbf{b})_\rho|$ denotes the L1-norm of the gradient computed w.r.t. the original binary vector $\mathbf{b}$ without adversarial perturbation.

Enforcing additional constraints over the regularity of f_y can make the attack objective function closer to submodularity. An example can be found as proposed in [24], where the parameters of the RNN-based classifier f_y are forced to be **positive** to guarantee the hidden layer output to be positive. The resultant attack objective $g(S)$ is thus proved to be submodular. It can be considered as a special case of the attackability framework proposed by Theorem.1.

Corollary 2.2. ***Submodular Attack Objective.*** *Let f_y be the RNN classifier defined by Eq. (7). We assume that all the parameters of f_y are positively valued. Let ρ be the set of attributes selected to attack. Given the positiveness constraint, the activation function of f_y is strictly concave and $m_{\Omega_{|\rho|}}$ of f_y is positive. We can further derive $\frac{2(1+|\nabla f_y(\mathbf{b})_\rho|)}{|\rho|^2} \geq m_{\Omega_{|\rho|}} > 0$ and $M_{\Omega_{|\rho|}} \geq \frac{2(1+|\nabla f_y(\mathbf{b})_\rho|)}{|\rho|^2}$. The submodularity ratio $\gamma_{\Omega_{|\rho|}} \geq 1$ of the corresponding $g(S)$. The attack objective is thus submodular.*

Comparing Corollary 2.1 and Corollary 2.2, we can observe balance between **model usability** and **attackability**. On one hand, though submodularity can improve the approximation quality of the greedy search based solution, enforcing the positiveness constraint inevitably introduces unnecessary bias into the classifier design. It can reduce the classification power of the classifier. On the other hand, as shown by Corollary.2.1, f_y is free of the additional positiveness constraint. It loosens the bound condition by allowing m_Ω to be negative, which gives more freedom in choosing the form of f_y. However, the attack objective loses submodularity as the lower bound of the submodularity ratio degrades significantly.

3.3 Link to Sensitivity Analysis of Attributes

Though conducting **sensitivity analysis** of the attributes is beyond our scope, it is also interesting to find out that the evasion attack on discrete attributes can unveil sensitivity information of different attributes, other than just cheating the classifier. For a given classification task, the adversarial attacker can gain the task-specific knowledge, such as, **the classifier's output might be highly sensitive to the perturbation over a specific subset of the discrete attributes.** These attributes are the origin of adversarial vulnerability of the systems running in the given task. They can potentially be informative features for classification at the same time. **On one side**, unveiling the useful information can help us interpret physical meaning of the selected attributes. **On the other side**, it also helps the adversarial attacker understand the statistical characteristics of the training data for better attack and/or information stealing. From the perspective of information security, such threat can be categorized as **data privacy breach**.

We can explore the risk induced by the evasion attack, that is, searching for the optimal support set of the attributes that can be used to *successfully attack the classifier over multiple discrete data instances.* We extend Eq.1 to apply it over a set of data instances:

$$S^* = \arg\max_{|S|\leq K} \sum_{r=1}^{R} g_r(S) \quad \text{where} \quad g_r(S) = \max_{l\subset S} f_y(\hat{\mathbf{x}}_r) \tag{9}$$

where $g_r(S)$ is the attack objective function defined based on each discrete data instance $\mathbf{x}_r$. The notation of $g_r(S)$ follow those used in Eq. (1). As each $g_r(S)$ is at least weakly submodular, the sum of $g_r(S)$ is also weakly submodular. The submodularity ratio of the attack objective function over multiple data instances gives as $min\{\gamma^1, \gamma^2, ..., \gamma^R\}$, where γ^r is the submodularity ratio of each $g^r(S)$. We can still use greedy search to solve the attack problem and find out *the set of discrete attributes that are generally useful for attacking multiple data instances.* Intuitively, the output of the classifier f_y should be statistically sensitive to adversarial perturbation over these identified attributes. In the empirical study, we confirm our intuition by first measuring attribute-wise sensitivity with the one-factor-at-a-time method [4]. We then observe the overlapping between the top-ranked attributes according to their sensitivity scores and the attributes selected for adversarial attack in Section.5.

4 PROVABLE APPROXIMATION QUALITY OF GRADIENT-GUIDED GREEDY SEARCH

The computational complexity of *FSGS* increases drastically as the number of the modifiable attributes in discrete instance $\mathbf{x}$ becomes larger. In our application datasets presented in next section, there can be more than 10k modifiable attribute values. This is the major bottleneck for applying *FSGS* practically. To address the issue, we propose an efficient attack solution based on orthogonal matching pursuit [30], which is referred as *Orthogonal Matching Pursuit-based Greedy Search* (OMPGS). This method narrows down the candidate attribute value modifications to those that correlate the best with the orthogonal complement of the attributes already selected to modify. These attribute values are identified by first calculating the gradient of $f_y(\mathbf{x})$ with respect to $\mathbf{b}$, denoted as $\nabla(f_y(\mathbf{b}))$. The candidate attributes to modify are chosen as the ones corresponding to the gradient elements with the largest magnitudes. We conduct greedy search only within these candidate features to choose the optimal attribute for adversarial tuning.

Algorithm 2 presents the pseudo codes of *OMPGS*. In each iteration, for each subset l' of the previous support set S_{k-1}, we compute the gradient of f_y with respect to $\mathbf{b}_{l'}$ (binary matrix $\mathbf{b}$ with the entries in l' changed). The top-k' attributes with the largest magnitudes in the gradient vector are selected to form a candidate set s'. We can then find out the optimal attribute s to extend for each subset l' and record the optimal attack objective value $g(S_{k-1} \cup s)$. In the outer iteration, we choose finally the attribute j^* producing the largest marginal gain to add into S_{k-1}.

The worst case cost of objective function evaluation in each iteration of *OMPGS* is bounded by $O((\sum_{i=1}^{k-1} C_k^i))|k'|$. We adjust k' to achieve a trade-off between enlarging the search range of greedy search and the cost of function evaluation. Usually k' is much less than $|H| - |S_k|$, espeically when H is significantly large (e.g. H >10k). Thus *OMPGS* runs significantly faster than *FSGS*.

To further illustrate the rationality of the proposed *OMPGS*, we show that the magnitude of each element in $\nabla(f_y(\mathbf{b}))$ can be used to measure the marginal contribution of adding the corresponding attributes in the candidate set for adversarial attack.

Theorem 3. ***Gradient as Indicator.*** *We assume S and T as two independent sets of the discrete attributes. $S \cap T = \emptyset$. $|T| \leq k$. Modification $\hat{\mathbf{b}}_{S\cup T}$ and $\hat{\mathbf{b}}_S$ are made by the optimal solution of the attack given the support set as $S \cup T$ and S, resulting in $\hat{\mathbf{x}}^{S\cup T}$ and $\hat{\mathbf{x}}^S$, respectively. Following the notations in Theorem 1, the lower bound of the marginal gain by adding extending the set S to $S \cup T$ gives as:*

$$f_y(\hat{\mathbf{x}}^{S\cup T}) - f_y(\hat{\mathbf{x}}^S) \geq \frac{1}{2M_{|S|+k}} \|\nabla f_y(\hat{\mathbf{b}}_S)_T\|_2^2 \tag{10}$$

```
Input: The attack budget $K$, the set function $g(S)$ defined by
        Eq. (1) and the set $H = \{(i, j), i = 1...n, j = 1...m\}$ of
        all the modifiable discrete attributes
Output: selected support set $S_k$, with $|S_k| \leq K$; $g(S_k)$ and
        the optimal subset of $S_k$ achieving the attack goal
$S_0 \leftarrow \emptyset$
for $k = 1$ to $K$ do
    $T = \emptyset$
    for $l' \subset S_{k-1}$ do
        $r \leftarrow \nabla f_y(\mathbf{b}_{l'})$
        $\mathbf{s}' = \{s'_1, s'_2, ..., s'_{k'}\} \leftarrow \arg\max_{j \in \{H/S_{k-1}\}} | < e_j, r > |$
        $s \leftarrow \arg\max_{j \in \mathbf{s}'} g(S_{k-1} \cup \{j\})$
        $T = T \cup \{(s, g(S_{k-1} \cup s))\}$
    end
    $j^* \leftarrow \arg\max_{j \in T} g(S_{i-1} \cup \{j\})$
    $S_k \leftarrow S_{k-1} \cup \{j^*\}$
end
```

Algorithm 2: Orthogonal Matching Pursuit based Greedy Search (OMPGS)

*where $M_{|S|+k}$ is the bound condition parameter defined on the domain $\Omega_{|S|+k}$ according to Eq.3. When $m_{|S|+k} > 0$ in Eq.3, the attack objective $g(S)$ is **submodular**. The upper bound of the marginal gain can be further formulated as:*

$$f_y(\hat{\mathbf{x}}^{S \cup T}) - f_y(\hat{\mathbf{x}}^S) \leq \frac{1}{2m_{|S|+k}} \|\nabla f_y(\hat{b}_S)_T\|_2^2 \tag{11}$$

*If $m_{|s|+k} \leq 0$, the attack objective is **weakly-submodular**. The upper bound of the marginal gain gives as:*

$$f_y(\hat{\mathbf{x}}^{S \cup T}) - f_y(\hat{\mathbf{x}}^S) \leq \tilde{\psi} \|\nabla f_y(\hat{b}_S)_T\|_2^2 \tag{12}$$

where $\tilde{\psi} = \frac{|\nabla f_y(b_S)_T|}{\|\nabla f_y(b_S)_T\|_2^2} + \frac{m_{|S|+k} k^2}{2\|\nabla f_y(b_S)_T\|_2^2}$

According to Theorem 3, the magnitude of each element in the gradient vector derived in each iteration of Algorithm 2 provides a bounded estimate of the marginal gain by adding the corresponding attribute into the support set to attack. Intuitively, with the bounded estimate, we can shrink the candidate set for the attack in order to focus on the attributes that can potentially bring more improvement of the attack objective than the others. In this way, we reduce the number of attributes to traverse during the greedy search of each iteration, while preserving the solution quality as much as possible. Indeed, we can provide a provably lower bound of the approximation quality of *OMPGS* as follows:

Theorem 4. *With ψ defined in Theorem 3, let S_k be the support set of the selected attributes to attack with OMPGS, the corresponding objective value g^{OMPGS} is bounded from below as:*

$$\begin{aligned} g^{OMPGS} &\geq (1 - e^{m_{\Omega_{|S_k^*|}}/M_{\Omega_{|S_k^*|}}}) g^{OPT} \quad, (m_{\Omega_{|S_k^*|}} > 0) \\ g^{OMPGS} &\geq (1 - e^{1/(2\tilde{\psi} M_{\Omega_{|S_k^*|}})}) g^{OPT} \quad, (m_{\Omega_{|S_k^*|}} \leq 0) \end{aligned} \tag{13}$$

Theorem 4 gives the intuition that the performance of *FSGS* will be better than that produced by *OMPGS*, when the attack objective is submodular. *FSGS* and *OMPGS* have the same lower bound of the approximation quality. In addition, we find that the upper and lower bound of the marginal gain are tighter when the classifier f_y is tightly bounded and the attack objectgive is submodular, compared to the weakly submodular opponent. If the classifier y_f is well bounded to gurantee the submodularity, the gradient magnitude of y_f provides a more accurate estimate of the marginal contribution of each attribute. *OMPGS* thus achieves a good balance between economic computing and effective greedy exploration.

5 EXPERIMENTS

5.1 Dataset Information

We include two real-world evaluation datasets, **cyber security** and **electronic medical service**, which are briefly introduced due to space limits. More information can be found in Appendix A.

Intrusion Prevention System Dataset (IPS) in Cyber Security. We collect one day of IPS records from 242,467 endpoint devices containing 29,641 intrusion events. Each intrusion instance is composed by 20 intrusion steps. On each intrusion step, one of 1,103 different malicious actions can be selected. We embed each action as a 70-dim vector. Then one IPS data instance is given as $\mathbf{x} \in R^{20*1103*70}$ according to the definition given in Section 3. A classifier is built to predict if, given an intrusion, the next action would fall into 2 highly malicious actions and a third class for all others. The evasion attack is non-targeted. We aim at mis-classifying an x to any of the two classes other than the true class label, by replacing the original action at a given intrusion step with a new action.

Electronic Health Records (EHR) [13]. The real-world EHR dataset consists of time-ordered medical visit records of 7314 patients. One patient data instance is organized as a tensor $\mathbf{x} \in R^{200*4130*70}$, with 200 medical visits[1], 4130 diagnosis codes[2] each of which is embedded as a 70-dim vector. A classifier is built for a binary task: predicting the risk of a patient suffering a heart disease. The evasion attack is simply to flip the binary classification output by changing the presence of a diagnosis codes in one visit (presence to absence, or absence to presence).

5.2 Experimental Setup

We instantiate the attackability study to the standard Long Short-Term Memory (LSTM) model. To demonstrate the link between the regularity of the classifier and its attackability, we train three LSTM classifiers following different regularity constraints for each dataset: **1) LSTM**: standard LSTM without any additional constraints, whose attack objective is *weakly submodular with low submodularity ratio*; **2) LSTM-Sub**: LSTM with positiveness constraints proposed in [24], whose attack objective is *strictly submodular* according to Corollary 2.2; **3) LSTM-Noise**: LSTM with parameter truncation, i.e., any parameters with their values less than -1 are truncated and assigned randomized positive values.

To show how the attack methods perform given different levels of attack difficulty, we require each attack method to cause misclassification with a classification probability of 0.5 and 0.7, noted as *Attack Confidence Threshold* in the results.

[1]For the patients with less than 200 visits, we pad the empty observations by setting the corresponding $b_i^j = 0$.

[2]One code is the occurrence of a disease, a symptom, or an abnormal finding, seehttp://www.icd9data.com/.

Table 1: Attack Performances on IPS dataset

Model	Attack Confidence = 0.5												
	Attack Algorithm	**k** = top 2			**k** = top 4			**k** = top 6			**k** = top 10		
		ANC ↓	AI ↓	SR ↑	ANC ↓	AI ↓	SR ↑	ANC ↓	AI ↓	SR ↑	ANC ↓	AI ↓	SR ↑
LSTM	SGS	2.93	3.87	0.89	2.41	3.03	0.91	2.12	2.55	0.90	1.72	1.96	0.89
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	1.51	1.00	1.00	3.23	1.00	1.00	3.74	1.00	1.00	3.86	1.00	1.00
	OMPGS-Rand	2.75	3.03	0.98	2.98	3.42	0.98	3.12	3.66	0.96	3.37	4.15	0.96
	OMPGS	**2.12**	**2.15**	**0.98**	**1.71**	**1.75**	**0.98**	**1.60**	**1.65**	**0.98**	**1.52**	**1.57**	**0.98**
LSTM-Noise	SGS	3.39	4.45	0.90	2.76	3.39	0.86	2.47	2.98	0.89	1.98	2.25	0.89
	FSGS	ANC = 1.20 AI = 1.22 SR = 1.00											
	GradAttack	1.59	1.00	1.00	2.43	1.00	1.00	3.62	1.00	1.00	5.97	1.0	1.00
	OMPGS-Rand	2.33	2.56	0.99	2.70	3.19	0.99	2.93	3.55	1.0	3.23	4.12	0.99
	OMPGS	**1.78**	**1.82**	**0.99**	**1.57**	**1.61**	**1.00**	**1.47**	**1.49**	**1.00**	**1.40**	**1.41**	**1.00**
LSTM-Sub	SGS	2.96	3.96	0.96	2.47	3.03	0.93	2.18	2.51	0.94	1.78	1.93	0.94
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	1.20	1.00	1.00	1.34	1.00	1.00	1.34	1.00	1.00	1.40	1.00	1.00
	OMPGS-Rand	1.69	1.78	0.99	2.29	2.62	0.99	2.59	3.12	0.99	3.02	3.84	0.99
	OMPGS	**1.01**	**1.01**	**0.99**	**1.01**	**1.01**	**0.99**	**1.01**	**1.01**	**0.99**	**1.01**	**1.01**	**0.99**
Attack Confidence = 0.7													
LSTM	SGS	3.03	4.03	0.91	2.52	3.15	0.91	2.11	2.56	0.90	1.67	1.91	0.90
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	1.50	1.00	0.99	3.23	1.00	0.99	3.76	1.00	0.99	3.88	1.00	0.99
	OMPGS-Rand	2.78	3.08	0.99	3.11	3.61	0.99	3.23	3.87	0.97	3.41	4.26	0.91
	OMPGS	**2.18**	**2.22**	**0.98**	**1.75**	**1.79**	**0.98**	**1.69**	**1.75**	**0.98**	**1.56**	**1.64**	**0.96**
LSTM-Noise	SGS	3.34	4.38	0.87	2.79	3.56	0.90	2.45	2.97	0.89	2.45	2.97	0.89
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	1.59	1.00	0.99	2.44	1.00	0.99	3.58	1.00	0.99	5.96	1.00	0.99
	OMPGS-Rand	2.45	2.71	0.99	2.83	3.40	0.99	3.06	3.72	1.00	3.24	4.02	0.91
	OMPGS	**1.95**	**2.02**	**0.99**	**1.70**	**1.75**	**1.00**	**1.63**	**1.65**	**1.00**	**1.58**	**1.59**	**0.99**
LSTM-Sub	SGS	3.07	4.16	0.95	2.56	3.20	0.97	2.29	2.73	0.96	1.96	2.17	0.95
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	1.22	1.00	0.98	1.36	1.00	0.97	1.35	1.00	0.97	1.45	1.00	0.97
	OMPGS-Rand	1.70	1.80	1.00	2.27	2.64	1.00	2.65	3.20	1.00	3.05	3.92	1.00
	OMPGS	**1.04**	**1.04**	**1.00**	**1.02**	**1.02**	**1.00**	**1.01**	**1.01**	**1.00**	**1.02**	**1.02**	**1.00**

We include 4 state-of-the-art greedy-search based attack methods, as well as the proposed *OMPGS* in the study.
1. Stochastic Greedy Search (SGS) [20]: SGS selects randomly a subset of attributes as the candidate of the greedy search in each iteration. Compared to *FSGS*, SGS is more efficient since it doesn't traverse every unselected attribute. As a price to pay, the approximation ratio of SGS degrades.
2. Graident-based Attack (GradAttack) [24]. It follows the same attack objective definition in Eq. (1) and also uses gradients to guide the attribute selection. However, it only considers the new attributes contributing largest marginal gain. According to Eq.1, it only searches a subset of the potentially feasible candidates, which inevitably cause loss of approximation quality.
3. OMPGS-Rand, a variant of **OMPGS**. It selects randomly one attribute from those with largest gradient magnitude in each iteration. It borrows the random sampling spirit from *SGS* to reduce the computational cost. It is evaluated to demonstrate the necessity of combining the gradient's guidance and the greedy search.
4. FSGS. It is presented in Algorithm 1.

Comparison of these methods to our proposed *OMPGS* targets on 1) showing the computational efficiency and attack performances of our proposed *OMPGS*; and 2) the attackability of an evasion attack task is independent from the choice of specific attack methods. It depends on the functional characteristics of the classifier and the characteristics of the data on which the classifier is applied. For *SGS* and *OMPGS*, we vary the number of attributes **k** in the candidate set in each iteration (**k**=2,4,6,10 in Table.1 and Table.2). Higher **k** indicates a larger exploration range of greedy search, thus usually brings better attack performances. In contrast, *GradAttack* prefers smaller **k**. In each iteration, *GradAttack* selects **k** candidates of greatest gradient magnitudes. It is designed to traverse all possible combinations of the **k** attributes and take the best combination. Larger **k** usually introduces unnecessary attribute changes or changes with adverse effects on attack.

Benchmark Metric We measure **Accuracy Score**, **F1 score** and **AUC score** to evaluate the usability of the trained *LSTM*-based classifiers. To illustrate the computational efficiency of different methods, we force all the methods except *FSGS* to halt after 60s and compare the **success rate** (noted as **SR**) of evasion attacks. Higher *SR* denotes faster attack. We allow *FSGS* to run for 3600s and report the metrics after it stops. Furthermore, we record the **average number of attribute (*ANC*)** and the **average number of iteration (*AI*)** of each method to achieve the attack goal. Both metrics are used to measure **attack efficiency**. Note that *ANC* does not necessarily equal to *AI*. According to Eq. (1), maximizing the set function based attack objective is not obliged to increase the support set S in each iteration. A successful evasion attack with **lower ANC** indicates better preserving data integrity, thus more invisible to data sanitary check. *ANC* is thus the most important indicator measuring effectiveness of attack methods. In contrast, **AI** is used to show **the computational cost**, as more iterations require more objective function evaluations. More details about the datasets and experimental set up can be found in the supplements.

5.3 Results on IPS and EHR Datasets

Table 1 and Table 2 illustrate the performances of all the attack methods on the 3 LSTM models. Table 3 and Table 4 show the

Table 2: Attack Performances on EHR dataset

Model	Attack Confidence = 0.5												
	Attack Algorithm	**k** = top 2			**k** = top 4			**k** = top 6			**k** = top 10		
		ANC ↓	AI ↓	SR ↑	ANC ↓	AI ↓	SR ↑	ANC ↓	AI ↓	SR ↑	ANC ↓	AI ↓	SR ↑
LSTM	SGS	3.65	5.13	0.10	3.35	4.72	0.14	2.98	4.07	0.12	3.63	5.19	0.25
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	3.06	1.84	0.98	4.26	1.27	0.99	5.30	1.06	0.98	6.35	1.00	0.98
	OMPGS-Rand	2.38	2.45	0.97	2.58	2.73	0.97	2.85	3.06	0.97	2.91	3.25	0.97
	OMPGS	**2.06**	**2.08**	**0.98**	**1.92**	**1.94**	**0.98**	**1.89**	**1.91**	**0.98**	**1.82**	**1.83**	**0.98**
LSTM-Noise	SGS	4.43	6.5	0.06	3.79	5.59	0.09	3.28	4.68	0.08	2.78	3.48	0.07
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	3.74	2.74	0.97	5.03	1.40	0.97	5.81	1.08	0.97	8.4	1.0	0.97
	OMPGS-Rand	3.21	3.30	0.97	3.47	3.67	0.97	3.67	3.90	0.97	3.90	4.24	0.96
	OMPGS	**2.97**	**3.00**	**0.98**	**2.96**	**3.0**	**0.98**	**3.01**	**3.05**	**0.98**	**3.06**	**3.12**	**0.98**
LSTM-Sub	SGS	4.61	6.82	0.45	4.25	6.18	0.48	3.66	0.48	0.44	3.46	4.78	0.45
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	1.96	1.79	0.99	3.48	1.11	0.99	5.01	1.03	0.99	7.67	1.00	0.99
	OMPGS-Rand	1.91	1.98	0.98	2.21	2.33	0.98	2.32	2.52	0.98	2.52	2.83	0.98
	OMPGS	**1.64**	**1.67**	**0.99**	**1.44**	**1.46**	**0.99**	**1.47**	**1.50**	**0.99**	**1.47**	**1.48**	**0.99**
Attack Confidence = 0.7													
LSTM	SGS	4.56	7.32	0.15	4.36	6.46	0.16	4.14	6.02	0.16	4.18	6.37	0.28
	FSGS	ANC = 1.00 AI = 1.00 SR = 1.00											
	GradAttack	3.27	2.01	0.98	4.33	1.28	0.99	5.44	1.15	0.99	6.28	1.0	0.97
	OMPGS-Rand	2.55	2.64	0.97	2.81	2.98	0.96	2.98	3.23	0.97	3.25	3.58	0.97
	OMPGS	**2.31**	**2.32**	**0.98**	**2.05**	**2.06**	**0.98**	**2.02**	**2.05**	**0.98**	**2.02**	**2.05**	**0.98**
LSTM-Noise	SGS	5.16	7.58	0.10	4.35	6.45	0.08	4.30	6.69	0.10	4.09	6.23	0.17
	FSGS	ANC = 1.00 AI = 1.20 SR = 1.00											
	GradAttack	3.86	2.74	0.97	5.35	1.53	0.97	6.03	1.10	0.97	8.57	1.00	0.96
	OMPGS-Rand	3.36	3.47	0.97	3.88	4.09	0.97	3.90	4.16	0.97	4.22	4.54	0.94
	OMPGS	**3.15**	**3.20**	**0.97**	**3.16**	**3.21**	**0.97**	**3.18**	**3.24**	**0.98**	**3.23**	**3.33**	**0.98**
LSTM-Sub	SGS	5.19	8.00	0.55	4.97	7.62	0.63	4.55	6.83	0.54	4.13	6.11	0.54
	FSGS	ANC = 1.03 AI = 1.04 SR = 1.00											
	GradAttack	1.94	1.55	0.98	3.36	1.12	0.99	5.02	1.01	0.98	7.63	1.00	0.99
	OMPGS-Rand	2.03	2.08	0.98	2.31	2.47	0.98	2.48	2.73	0.98	2.63	2.99	0.98
	OMPGS	**1.78**	**1.82**	**0.99**	**1.60**	**1.63**	**0.99**	**1.55**	**1.56**	**0.99**	**1.56**	**1.57**	**0.99**

Table 3: Usability of the classifier on IPS dataset

Classifier	**Accuary**	**Macro F1**	**AUC**
LSTM	0.9597	0.9432	0.9872
LSTM-Noise	0.9668	0.9533	0.9870
LSTM-Sub	0.8867	0.8687	0.9204

Table 4: Usability of the classifier on EHR dataset

Classifier	**Accuary**	**F1**	**AUC**
LSTM	0.9321	0.8831	0.9096
LSTM-Noise	0.9277	0.8710	0.8948
LSTM-Sub	0.9295	0.8730	0.8944

classification accuracy of different LSTM-based classifiers on both datasets. We can summarize the observations as follows:

First, the utility scores indicate the usability of the real-world classification tasks. It is not surprising to see that *LSTM* enjoys the highest accuracy on both datasets. In contrast, *LSTM-Sub*, due to the positiveness constraint, performs relatively poorly. As expected, truncation noise doesn't impact much the classification of *LSTM-Noise* thanks to the highly redundant network architecture.

Second, ***LSTM-sub* is easier to attack (or more "attackable") than *LSTM* and *LSTM-Noise***, because it needs the least number of attribute changes and objective function queries. This observation confirms what we reveal earlier: *for evasion attack on discrete data, a better regularized and bounded function form of the targeted classifier can improve the attackability of the evasion attack task.* In contrast, attack against *LSTM-Noise* is slightly easier than *LSTM* on IPS data set but significantly more difficult on EHR data set. The explanation behind the inconsistency is as follows. The classifier becomes generally less smooth due to the truncation noise, which hence makes the corresponding attack objective further from strict submodularity. As a result, the attack performances against *LSTM-Noise* become less tightly bounded by the regularity of the classifier according to Theorem 1, while more data-dependent.

Third, ***FSGS* runs slowly** on both datasets, while presents the best attack performances. This observation is highly consistent with the quality bound of *FSGS* given by Theorem 2. *FSGS* can provide the superior approximation quality over any greedy-search based variant, no matter whether the attack objective is submodular or weakly-submodular. Both *SGS* and *GradAttack* improve computational complexity by reducing the number of candidate attributes in each iteration. Nevertheless, they don't consider explicitly how to minimize the loss of the quality of the solution.

Last, our proposed ***OMPGS* method always has high success rate and finds quickly the valid attacks** (1-2 changes to make in one instance), because it combines the merit of orthogonal matching pursuit (OMP) and greedy search. On one hand, OMP helps to narrow down the greedy search to the attributes that are the most likely to be useful in the attack. On the other hand, *OMPGS* still conducts greedy search to explore the feasible subset of attributes improving the attack objective. *OMPGS* provides a good trade-off between exploration and exploitation in this sense. Without greedy search, the attack performances of *OMPGS-Rand* are significantly worse than *OMPGS*. It confirms further the necessity of integrating OMP-based guidance and greedy exploration to deliver a fast while effective search in the attribute space.

5.4 Sensitivity of the Selected Attributes

For each data set, we conduct one-factor-at-a-time sensitivity analysis [4] over the discrete attributes. Given a data instance, we change each attribute while keeping all the others fixed. The averaged change of the probabilistic classification output over all the testing instances is used as the attribute-wise sensitivity measurement. A larger average change indicates that the classifier's output is more sensitive to the change over the corresponding attribute. Following Eq.9, we launch *OMPGS* based attack against *LSTM-Sub* on both datasets. The attack spins till the goal, *Attack Confidence*= 0.5, is reached.

For attacking *LSTM-Sub*, we finally select 5 attributes on IPS data and 7 attributes on EHR data. On IPS data set (1103 attributes), **3** of the **5** attributes also appear as the top 50% sensitive attributes. Especially, **2** of them are ranked as the top 12% sensitive attributes. On EHR data set (4130 attributes), **5** of the **7** attributes show up in the list of the top 50% sensitive attributes. Furthermore, **3** of them are ranked as top 12% sensitive attributes. The interesting overlapping between the attributes useful for attack and the top-ranked sensitive attributes confirms our intuition about stability of the classification model. Though highly sensitive attributes can help better capture difference between different classes, they can also increase adversarial vulnerability of the classifier [37]. Moreover, the observation unveils that evasion attack can also be used to exploit the statistical characteristics (e.g. attribute sensitivity) of the training data of the classification system. It can not only harm usability of the classifier, but also steal privacy-sensitive information of the training data.

6 CONCLUSION

In this paper, we explore attackability guarantees of evasion attack against a targeted classifier on discrete input. We unveil the regularity constraints over the function form of the targeted classifier. Despite of the general NP-hard complexity, evasion attack targeted at a classifier bounded by the constraints enjoys the approximately diminishing return property of weak submodularity. It can thus deliver an efficient attack via fast greedy search and provide provable attack performances. Furthermore, following the framework of the attackability study, we propose an orthogonal matching pursuit guided greedy search to achieve a good balance between economic candidate attribute search and efficient attack. Both theoretical and empirical study confirm the merits of the proposed method. For a classifier with discrete inputs, our work reveals theoretically a link between functional characteristics of the classification function and its adversarial vulnerability. Our study is thus useful for improving robustness of the classifier facing hostile adversarial threats.

REFERENCES

[1] Amir Akbarnejad and Stephen Günnemann. Adversarial attacks on node embedding via graph poisoning. In *ICML*, 2019.
[2] Blaine Nelson Battista Biggio and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.
[3] Davide Maiorca Blaine Nelson Nedim Šrndić Pavel Laskov Giorgio Giacinto Battista Biggio, Igino Corona and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML PKDD*, 2013.
[4] Chai T Mena-Carrasco M Tang Y Blake DR Blake NJ Vay SA Collatz GJ Baker I Berry JA Montzka SA Sweeney C Schnoor JL Campbell JE, Carmichael GR and Stanier CO. Photosynthetic control of atmospheric carbonyl sulfide during the growing season. *Science*, 322, 2008.
[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE S&P*, 2017.
[6] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *SPW*, 2018.
[7] Jan. Vondrák Chandra Chekuri and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SICOMP*, 43(6), 2014.
[8] Lin Chen, Moran Feldman, and Amin Karbasi. Weakly submodular maximization beyond cardinality constraints: Does randomization help greedy? In *ICML*, 2017.
[9] Amir Akbarnejad Daniel Zügner and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *KDD*, 2018.
[10] Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, 2011.
[11] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *ACL*, 2018.
[12] Alexandros G. Dimakis Ethan R. Elenberg, Rajiv Khanna and Sahand Negahban. Restricted strong convexity implies weak submodularity. *Annuals of Statistics*, 2016.
[13] Qiuling Suo Quanzeng You Jing Zhou Fenglong Ma, Jing Gao and Aidong Zhang. Risk prediction on electronic health records with prior medical knowledge. In *KDD*, 2018.
[14] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *SPW*, pages 50–56, 2018.
[15] Shayan Oveis Gharan and Jan Jan Vondrák. Submodular maximization by simulated annealing. In *SODA*, 2011.
[16] G.Nemhauser and L.A.Wolsey. An anlaysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14, 1978.
[17] G.Nemhauser and L.A.Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3, 1978.
[18] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. Adversarial texts with gradient methods. *ArXiv*, abs/1801.07175, 2018.
[19] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.
[20] Avinatan Hassidim and Yaron Singer. Robust guarantees of stochastic greedy algorithms. In *ICLR*, 2017.
[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
[22] J.Bilmes and W.Bai. Deep submodular functions. *arXiv preprint arXiv:1701.08939*, 2017.
[23] Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. Adversarial examples for natural language classification problems. In *ICLR*, 2018.
[24] Qi Lei, Lingfei Wu, Pin-Yu Chen, Alexandros Dimakis, Inderjit Dhillon, and Michael Witbrock. Discrete attacks and submodular optimization with applications to text classification. In *SysML*, 2019.
[25] Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. Adversarial training methods for semi-supervised text classification. In *ICLR*, 2016.
[26] G.Nemhauser M.L.Fisher and L.A.Wolsey. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 8, 1978.
[27] Joseph Naor Niv Buchbinder, Moran Feldman and Roy Schwartz. Submodular maximization with cardinality constraints. In *SODA*, 2014.
[28] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., 1982.
[29] Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM*, 2016.
[30] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *ACSSC*, 1993.
[31] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging NLP models. In *ACL*, 2018.
[32] Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *CoRR*, abs/1707.02812, 2017.
[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2013.
[34] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review, 2019.
[35] Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I. Jordan. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *ArXiv*, abs/1805.12316, 2018.
[36] Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y. Zhao. Automated crowdturfing attacks and defenses in online review systems. In *ACM CCS*, 2017.
[37] Fei Zhang, Patrick P. K. Chan, Battista Biggio, Daniel S. Yeung, and Fabio Roli. Adversarial feature selection against evasion attacks. *IEEE Transactions on Cybernetics*, 46, 2016.
[38] Daniel Zugner, Amir Akbarnejad, and Stephan Gunnemann. Adversarial attacks on neural networks for graph data. In *IJCAI*, 2019.

A DATASET INFORMATION

We include two datasets collected from real-world classification applications on **cyber security** and **electronic medical service**, as summarized as follows:

Intrusion Prevention System Dataset (IPS). Modern cyberattacks have reached high levels of complexity. An attacker who is trying to compromise a computer system has to perform a series of attack steps (i.e., reconnaissance, exploitation, and persistence) to achieve such goal. For each of the attack steps, attackers have a choice of executing a series of malicious actions, such as exploiting Apache Struts or exploiting Wordpress file download , which are usually scripted and automated. We collect one day of IPS records from 242,467 endpoint devices containing 29,641 time series of attack events. Each sequence instance is composed by 20 attack steps. On each attack step, the attacker can choose one of 1103 different malicious actions registered as highly threatening ones. Thus one data instance of IPS data is given as $\mathbf{x} \in R^{20*1103*70}$ according to the definition given in Section.3. Each of the 1103 malicious actions is projected to a 70-dimensional embedding vector. In our study, each sequence instance is used as input to the classification system, which predicts the most likely attack operation conducted at the immediately successive step of the sequence. Based on the prediction output, security analysts can proactively take prevention actions. We focus on predicting the occurrence of the two most threatening actions related to recently uncovered vulnerability. We thus study a 3-class classification task: the 2 highly malicious actions and all the others as the third class.

Electronic Health Records (EHR) [13]. The real-world EHR dataset consists of time-ordered medical visit records of 7314 patients. Each patient has from 4 to 200 medical visits. Each visit record is composed by a subset of 4130 discrete ICD9 diagnosis codes[3]. Each diagnosis code represents occurrence of a disease, a symptom, or an abnormal finding. Using the historical EHR data of patients, we can predict the risk of patients suffering the target diseases. In this experiment, our target is a binary classification task: we forecast whether a patient will suffer heart failure disease in the future. In our experiments, a data instance of EHR data set is organized as a tensor $\mathbf{x} \in R^{200*4130*70}$ with each of the 4130 diagnosis codes projected to a 70-dimensional embedding vector. For the patients with less than 200 visits, we pad the empty observations by setting the corresponding $b_i^j = 0$.

We split randomly each dataset into 3 non-overlapped subsets for training, testing and evaluating the attack performances. For IPS data, 21,214 and 7,427 sequence instances are used to train and evaluate the classification accuracy of the classifier. The left 1,000 instances are used for benchmarking the performances of the evasion attack. On IPS data, we focus on the adversarial attribute change by replacing the original action at a given attack step with a new action. For EHR data, we use 5,679 and 1,135 patient visit instances to train and test the classifier. The rest 500 visit instances are used to attack. As the diagnosis code in each visit instance is binary, we conduct the attack by simply flipping the codes.

[3]http://www.icd9data.com/

B EXPERIMENTAL SETUP

We instantiate the attackability study to a popularly adopted RNN based classifier, standard Long Short-Term Memory (LSTM) model in the experiments. Without loss of generality, we use *Tanh* activation function in the LSTM classifier and exclude the dropout module. Theorem.1 applies to LSTM similarly as the simpler RNN architecture. To demonstrate the link between the regularity of the classifer and its attackability, we train three LSTM classifiers following different regularity constraints for each dataset. First, we use the standard LSTM mode without any additional constraints over the models' parameter. According to Corollary 2.1, the evasion attack objective targeted at the classifier is weakly submodular with low submodularity ratio. This classifier is referred as *LSTM* in the experiments. Similarly, we enforce the positiveness constraints on the classifier proposed in [24]. The resultant LSTM is strictly submodular according to Corollary 2.2. It is noted as *LSTM-Sub* in the followings. Though *LSTM* and *LSTM-Sub* share the similar level of smoothness, the activation function of *LSTM-Sub* is truncated to be positive and thus presents strongly concavity. In contrast, the activation function of *LSTM* is convex on the negative input and in general is not concave any more. As shown by Defintion.1, the regularity parameter m of *LSTM-Sub* is large than that of *LSTM*. In other words, *LSTM-Sub* is more regularized than *LSTM*. We compare the attack performances against them to confirm our theoretical discussion and intuition: more regularized and bounded classifier is easier to be attacked with greedy-search based methods. Furthermore, we add additional parameter perturbations to *LSTM* by parameter truncation: any parameters with their values less than -1 are truncated and assigned randomized positive values. As per our empirical observation, most of *LSTM*'s parameters are larger than -1. Therefore, the parameter perturbation causes little changes of classification performances. But it can reduce the classifier's smoothness as the randomized parameter perturbation introduces unpredictable change to the first-order derivative of the classifier. We refer the noisy version of *LSTM* as *LSTM-Noise*. Comparing attack performances against *LSTM* and *LSTM-Noise* can demonstrate further that attackability of a less consistently regularized classifier is more difficult to be estimated.

The evasion attack task on IPS data is non-targeted. We aim at mis-classifying the attack sequence to any of the two classes other than the true class label. The evasion attack on EHR data simply flips the binary classification output. To show how the attack methods perform given different levels of attack difficulty, we require each attack method to cause misclassification with a classification probability of 0.5 and 0.7 respectively, as noted as *Attack Confidence Threshold* in the results. We include 4 state-of-the-art greedy-search based attack methods, as well as the proposed *OMPGS* in the study. The purpose is two-fold. Firstly we involve the baselines for comparative study, in order to show the computational efficiency and attack performances of our proposed *OMPGS*. Secondly, we show that the attackability of an evasion attack task is independent from the choice of specific attack methods. It depends on the functional characteristics of the classifier and the characteristics of the data on which the classifier is applied. Except *FSGS*, the other involved baseline approaches are as follows:

- **Stochastic Greedy Search (SGS)** [20]: SGS selects randomly a subset of attributes as the candidate of the greedy search in each iteration. Compared to *FSGS*, SGS is computationally more efficient since it doesn't traverse every unselected attribute. As a price to pay, the approximation ratio of SGS degrades, as shown in [20]. We believe that the proposed *OMPGS* performs better. Benefited from the gradient information used in *OMPGS* the candidate set of the greedy search greatly shrinks while preserving attack effectivity.
- **Graident-based Attack (GradAttack)** [24]. GradAttack follows the same attack objective definition in Eq.1 and also uses gradients to guide the attribute selection. However, it only considers the new attributes contributing largest marginal gain with respect to the currently best combination of attribute changes. According to Eq.1, GradAttack only searches a subset of the potentially feasible candidates in each iteration, which inevitably cause loss of approximation quality of the attack solution.

Besides, we also involve a variant of *OMPGS*, named as *OMPGS-Rand*. It works by selecting randomly one attributes from the attributes with largest gradient magnitude in each iteration. It is designed by borrowing the random sampling spirit from *SGS* to reduce the computational cost. The purpose of involving *OMPGS-Rand* is to demonstrate the necessity of combining the gradient's guidance and the greedy search within the candidate attributes. For all the attack methods except *FSGS*, we vary the size of the candidate attribute set for greedy search in each iteration from 2 to 10, noted as *top2*, *top4*, *top6* and *top10*. We implement all the attack methods and the 3 LSTM based classifiers using the Python library PyTorch and conduct all the experiments on Linux server with 2 GPUs (GeForce 1080Ti) and 16-core CPU (Intel Xeon).

We release the source codes for experimental study at https://github.com/X8GWRFJT/Attackability-Characterization-of-Adversarial-Evasion-Attack-on-Discrete-Data.