Discretization Inspired Defence Algorithm Against Adversarial Attacks on Tabular Data

Jiahui Zhou[1], Nayyar Zaidi[2], Yishuo Zhang[2], and Gang Li[2(✉)]

[1] College of Computer Science, Xi'an Shiyou University, Shaanxi 710065, China
[2] School of I.T., Deakin University, Burwood, VIC 3216, Australia
{nayyar.zaidi,zhangyis,gang.li}@deakin.edu.au

Abstract. Deep learning methods are usually trained via a gradient-descent based procedure, which can be efficient as it is not only end-to-end but also suitable for large quantities of data. However, gradient-based learning is vulnerable to adversarial attacks – which account for unperceivable changes in the input data to misguide a trained model. Though a plethora of work explored the adversarial learning (attacks and defences) in image datasets, the exploration of adversarial learning in tabular datasets has seen little attention. In this work, we study adversarial learning in tabular datasets. We investigate the role of discretization and demonstrate that discretizing numeric attributes offers a strong defence mechanism. The main contribution of this work is the proposition of two new defence algorithms for numeric tabular datasets, that utilize cut-points obtained from discretization, to forge a defence against various forms of adversarial attacks. We evaluate the effectiveness of our proposed method on a wide range of machine learning datasets and demonstrate that the proposed algorithms lead to a state-of-the-art defence strategy on tabular datasets.

1 Introduction

At the heart of deep learning is a parametric model in the form of `Artificial Neural Network` (`ANN`), which is trained by optimizing a differentiable objective function. The error is propagated back through the network, and each weight of the model is updated in an iterative gradient-descent optimization manner. This end-to-end training process, as it is known, is efficient as it can process notably large quantities of data in a strictly online or in some batch processing manner. However, this gradient-based learning has a fundamental weakness – it opens the door to adversarial attacks. The idea behind adversarial learning is that any malicious entity, if, has access to model weights/parameters and can obtain the respective gradients, then it can modify the input in a way, such that the desired output can be obtained from the model [7]. E.g., for an input $\mathbf{x}$ to a given model

J. Zhou and N. Zaidi—Equal Contribution.

J. Gama et al. (Eds.): PAKDD 2022, LNAI 13281, pp. 367–379, 2022.
https://doi.org/10.1007/978-3-031-05936-0_29

$f(\mathbf{x})$, $\mathbf{r}$ is an adversarial noise if $f(\mathbf{x}+\mathbf{r}) \neq f(\mathbf{x})$, where $|\mathbf{r}| \leq \epsilon$. It can be seen that the only challenge for the attacker is that the noise ($\mathbf{r}$) should be unperceivable. Well, for high-resolution images, one can easily make unnoticeable changes in the input data to fool the model. Therefore, in computer vision, adversarial attacks are considered a serious threat, and a lot of research has focused on building effective defence mechanisms [15].

Tabular dataset has some characteristics that challenge the plain application of adversarial attacks. They can have categorical features, leading to values that are unique and distinct, that is `Y` or `N` or `High` and `Low`, etc. Note, these values are represented as whole numbers: 0, 1 or 2, etc. – i.e., either encoded as bin numbers, or used in the one-hot-encoding format. Therefore, any changes to these values can easily be detected. Let us formalize adversarial attacks on tabular data in the following. We can denote the original unadulterated data as: $\mathcal{S}_{\text{data}} = [(\mathbf{x}^1, \mathbf{y}^1), \cdots, (\mathbf{x}^m, \mathbf{y}^m)]$, where $\mathbf{x} \in \mathcal{R}^n$ and $\mathbf{y} \in \mathcal{R}$. They are used to train the model: $f(\cdot)_{\boldsymbol{\theta}_s, \nabla_s}$. When an adversarial attack occurs, the adversarial sample $\mathcal{S}_{\texttt{adv}}$ is generated based on $f(\cdot)_{\boldsymbol{\theta}_s, \nabla_s}$, such that for each adversarial sample $\tilde{\mathbf{x}} \in \mathcal{S}_{\texttt{adv}}$, we have $f(\mathbf{x})_{\boldsymbol{\theta}_s, \nabla_s} = \mathbf{y} \neq \mathbf{f}(\mathbf{x}+\mathbf{r})_{\boldsymbol{\theta}_s, \nabla_s}$. The goal of the adversarial attack is defined as maximizing the following objective function: $\mathcal{L}_{\text{adv}}(f(\mathbf{x}+\mathbf{r})_{\boldsymbol{\theta}_s, \nabla_s}, \mathbf{y})$, where $|\mathbf{r}| \leq \epsilon$. Here, $\mathcal{L}_{\text{adv}}(f(\mathbf{x}+\mathbf{r})_{\boldsymbol{\theta}_s, \nabla_s}, \mathbf{y})$ is known as the adversarial risk on $\mathbf{x}$. We define adversarial risk over model $f(\cdot)$ as: $\mathbf{E}_{\mathbf{x} \in \mathcal{S}_{\text{data}}}(\mathcal{L}_{\text{adv}}(f(\mathbf{x}+\mathbf{r})_{\boldsymbol{\theta}_s, \nabla_s}, \mathbf{y}))$.

Let us suppose that every feature in our dataset is categorical. As discussed earlier, the final representation that we get for the features will only consist of well-round numbers. Any adversary aiming to choose a small value of ϵ, can easily be spotted[1]. Therefore, a simple strategy to defend against the adversarial samples for datasets with categorical data only is:

- Perform `Ceil` or `Floor` operations as advocated in [2]. E.g., if x_i is 3 and $\epsilon = 0.15$, the adversarial sample will have a value of 3.15, which will be converted back to 3 with `Floor` operation. Note, an adversary can also set $\epsilon = -0.15$, leading to $x_i = 2.85$. Now, if we perform the `Floor` operation, we obtain a value of 2. Note, if the value 2 is allowed, it is fine; otherwise, if the feature can only take values ≥ 3, the value 2 will violate the validity constraint of the feature and can be detected easily.

What if a tabular dataset has continuous features? In our previous example, if an adversarial sample has the values 3.15 or 2.85 – there is no way we can determine if it is not a legitimate value. And, therefore, adversarial attacks on numeric data can easily evade a manual inspection. Clearly, there is a need to determine whether a numeric value is adversarial or not. How about discretizing the feature and representing it as a categorical feature, or determining whether it is adversarial based on its distance from the discretization cut-point? These two questions will form the basis of our two proposed algorithms in this work. Generally, discretization is only employed to convert continuous features into categorical if a model can not handle continuous features. However, it has been

[1] This is one reason, why adversarial attacks against tabular data are not prevalent as compared to against image datasets.

shown recently [11], that discretization can lead to significantly better performance as well. In this paper, we show that it can be equally useful as a defence mechanism for adversarial attacks. Though related techniques such as quantisation have been used in adversarial defence on image datasets [2], their efficacy on tabular datasets is not well studied. Exploring what is the role, discretization can play in warding-off adversarial attack on tabular datasets has been the main motivation of this work.

In this work, we will devise defence strategies under two scenarios. The first scenario is where we are allowed to modify the input data while training the defence model. Here, we discretize the input continuous data – $\mathcal{S}_{\text{data}}$, and then adversarially train the model on this new discretized data – $\mathcal{S}_{\text{d-data}}$, as well as adversarially generated data – $\mathcal{S}_{\texttt{adv}}$. We demonstrate the efficacy of this strategy by formalizing it in form of our first proposed algorithm named `D2A3` – ***D**iscretized-based **D**efence **A**gainst **A**dversarial **A**ttacks*. The second scenario is where we are not allowed to modify the input features, and the input to the model has to be original continuous features. For this case, we believe, again discretization can offer an excellent defence mechanism, but rather implicitly. In this work, we have proposed a new defence algorithm named `D2A3N` – ***D**iscretized-based **D**efence **A**gainst **A**dversarial **A**ttacks with **N**umeric Input* – which leverages the cut-points (boundaries) definition obtained from discretization on original data and exploit the distance to cut-points to determine if a data point is adversarial or not. We summarize the main contributions of this work as follows:

- We highlight the importance of discretization as a defence mechanism for attacks on tabular datasets, and demonstrate that a simple discretization of continuous features can be very effective towards multiple forms of adversarial attacks.
- We propose two algorithms – `D2A3` and `D2A3N`, which utilize discretization to develop defence strategies for continuous features in tabular datasets. We evaluate the effectiveness of our proposed algorithms on a wide range of datasets, and against various forms of attacks.

The rest of this paper is organized as follows. We will discuss the related work in Sect. 2. Our proposed algorithms are presented and discussed in Sect. 3. We do an empirical evaluation of our proposed algorithms in Sect. 4, and conclude in Sect. 5, with some pointers to future works.

2 Background and Related Work

2.1 Tabular Data Adversarial Defence and Attack

There are three kinds of adversarial attacks that are common for tabular datasets. `LowProFool` [1] is a white-box attack method in the tabular domain for generating imperceptible adversarial examples. It is based on minimizing the addition of (imperceptible) adversarial noise on the features via the gradient descent approach. The gradients of the adversarial noise are used to guide the

updates towards the opposite target class of the clean sample. At the same time, the penalty of the perturbation is set proportionally to the feature importance for confirming the minimal perceptibility of the adversarial noise. The benefit of the `LowProFool` is that the success rate is largely guaranteed, even the adversarial noise is imperceptible compared to many other white-box attack methods. It is the state-of-the-art white-box attack method on tabular data [1]. `DeepFool` [6] is another white-box attack method, and it works by adding adversarial noise to the clean sample by finding the distance between the sample and the model decision boundary. In `DeepFool`'s formulation, the smallest adversarial noise can be considered as the orthogonal projection between the sample and the class decision boundary (affine hyperplane). The advantage of the `DeepFool` compared to other classical gradient-based white box attacks, such as `FGSM`, is that the adversarial noise is more reliable and efficient as `DeepFool` always finds the generated adversarial sample close to the decision boundary and therefore the target class can be changed. The limitation of `DeepFool` is that the adversarial noise can be large when the sample is far away from the model decision boundary [1]. `FGSM` [3] is a classical white box attack method for image and typical numerical tabular datasets. The idea behind `FGSM` is quite simple and straightforward. It relies on adding the gradients into the original sample to create the adversarial sample.

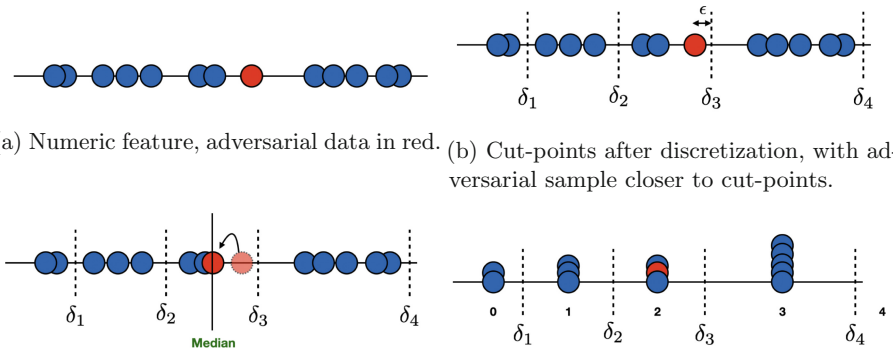
Defence methods against tabular data adversarial attacks are still limited in the current literature. A commonly used defence method for adversarial attack on continuous data is `Madry` [5]. It leverages the adversarial training to minimize the adversarial risk of the model. `Trade` [12] is another commonly used defence method for continuous data which minimizes the regularized surrogate loss instead of directly training adversarially.

Finally, `Thermometer` [2] is another defence mechanism that relies on idea similar to discretization. The `Thermometer` discretization for tabular data on $\mathbf{x}_i$ with k cut-points can be expressed as: $t(\alpha = \phi_{\text{ew}}(f_{\text{scale}}(\mathbf{x}_i)))_l = 1, \quad l \geq \alpha$. The $f_{\text{scale}}(\cdot)$ is the min-max scaler to scale the value of $\mathbf{x}_i$ into the range $[0, 1]$. The $\phi_{\text{ew}}(f_{\text{scale}}(\mathbf{x}_i))$ is the quantization function that uses the `equal-frequency` to obtain bin α on k cut-points. The array $t(\alpha)_l$ has k dimensions and l is the l-th dimension of the array. It can be seen that the `Thermometer` discretization is similar to one-hot-encoding after `equal-width` discretization. However, in contrast to one-hot encoding, `Thermometer` discretization can ensure that the order remains the same after discretization.

2.2 Discretization as Defence and Discretization Methods

Discretization is a commonly used and well-studied technique in machine learning [13]. It is to convert a continuous feature into a set of discrete values, which is usually done by sorting the data and then identifying some cut-points (also known as boundaries), and placing the continuous data point based on which bin does it fall into [14]. Each bin is labelled a number in range: $\{0, 1, \ldots, k\}$, where k is the total number of bins.

A simple illustration of how discretization can lead to a defence is shown in Fig. 1, where one of the data points (in red) is maliciously tampered (Fig. 1a). It



(a) Numeric feature, adversarial data in red.

(b) Cut-points after discretization, with adversarial sample closer to cut-points.

(c) Replacing the adversarial sample with the median of the bin.

(d) Data after discretization.

Fig. 1. Illustration of discretization as a defence against adversarial attack. (Color figure online)

can be seen that if we have the feature in a continuous format, there is no way for us to differentiate between these red data points and the others. However, let us suppose that we have obtained some boundaries $(\delta_1, \ldots, \delta_4)$ after running a discretization algorithm. We conjecture that the proximity of the data points to the boundaries can be an indication of an adversarial example. This can be seen in Fig. 1b, where this proximity is measured as a distance, ϵ. In practice, once the boundaries are identified, the data is discretized as the bin-number or represented with one-hot-encoding. In our example, the malicious red data will be assigned a value of 2, as shown in Fig. 1d. Now, if we just use bin-number (and train the original model) or do a one-hot-encoding (and train a modified model that takes in many more features as input) – the adversarial data is neutralized, which means that whatever the malicious intent of the attacker was, we have scaled it back to a value that our model expects (in our example, that is $\{0, 1, 2, 3\}$). Additionally, instead of using the bin-number, one can only replace the value of the adversarial data to the median of the bin, and keep all other data points the same – as depicted in Fig. 1c. This will help us in training a model that still takes as input the data in original format.

In Support of Discretization as Defence. We know that in `ANN` models, with all linear activation functions, the loss function tends to be linear with respect to the inputs as well. In such case, when the input $\mathbf{x}$ with the model $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$ is under adversarial attack with $\mathbf{x} + \mathbf{r}$, we have:

$$f(\mathbf{x} + \mathbf{r}) = \sigma(\mathbf{w}^\top (\mathbf{x} + \mathbf{r})) = \sigma(\mathbf{w}^\top \mathbf{x} + \mathbf{w}^\top \mathbf{r})$$

It can be seen that $\mathbf{w}^\top \mathbf{r}$ determines the success of adversarial attack. Even though, non-linear functions are typically used in deep `ANN` models, such as `Relu`, they are only piece-wise linear. Much of the work in designing a defence against

Algorithm 1: Algorithm D2A3 and D2A3N Training

Input: $\mathcal{S}_{\text{data}} = [(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^m, \mathbf{y}^m)]$, Discretization type – $\mathcal{C}$

Initial parameter $\boldsymbol{\theta}_s$ of the model $f(\cdot)$
Run discretization method $\mathcal{C}$ to obtain cut-points for each feature with $\Phi(\cdot)$
for iteration $q \subset Q$ in training model $f(\cdot)$ **do**
  for sample $\mathbf{x}^t$ in batch $\mathcal{X} \subset \mathcal{S}_{\text{data}}$ and $\mathbf{y}^t$ **do**
    if *D2A3* **then**
      Discretize with one-hot encoding $\Phi(\mathbf{x}^t)$
      Train $f(\cdot)$ with $\Phi(\mathbf{x}^t)$ and $\mathbf{y^t}$ via gradient descent to minimize:
        $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\mathbf{x}^t)), \mathbf{y}^t)$; `// Training Loss`
      Obtain adversarial sample $\tilde{\mathbf{x}}^t \in \mathcal{S}_{\text{adv}}$; `// Adversarial Training`
      Discretize with one-hot encoding $\Phi(\tilde{\mathbf{x}}^t)$
      Train $f(\cdot)$ with $\Phi(\tilde{\mathbf{x}}^t)$ via gradient descent to minimize:
        $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\tilde{\mathbf{x}}^t)), \mathbf{y}^t)$
    else
      Train $f(\cdot)$ with $\mathbf{x}^t$ and $\mathbf{y}^t$ via gradient descent to minimize:
        $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\mathbf{x}^t), \mathbf{y}^t)$
      Obtain adversarial sample $\tilde{\mathbf{x}}^t \in \mathcal{S}_{\text{adv}}$; `// Adversarial Training`
      Obtain data transformation: $\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))$
      Train $f(\cdot)$ with $\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))$ via gradient descent to minimize:
        $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))), \mathbf{y}^t)$; `// Adversarial Training Loss`
return $f(\cdot)$, $\Phi(\cdot)$, $\mathcal{M}(\cdot)$

adversarial attacks focus on *how to break the linearity between inputs and the output?* [2]. Well, discretization followed by one-hot-encoding leads to a non-linear model, and we claim that it can break the linearity between the input and the output, and hence, can provide an effective defence mechanism against adversarial attacks.

3 Methodology

3.1 D2A3 – Model Discretization as Defence

Our proposed algorithm D2A3, relies on discretizing the continuous features to categorical features as input, and therefore, instead of using a model with continuous input, the discretized model is trained and used in D2A3. One can utilize any discretization method. The detailed pseudocode of D2A3 is given in Algorithm 1, where it takes as input the training data $\mathcal{S}_{\text{data}}$, as well as the discretization method $\mathcal{C}$ – `equal-frequency, equal-width, MDL`[2]. Of course, changing the input data format can be considered as a limitation. We will discuss D2A3N in the next section, which addresses this issue.

[2] Note, we have combined the two proposed algorithms into one due to space constraints.

Algorithm 2: Algorithm D2A3 and D2A3N Defence

Input: $\mathcal{S}_{\text{adv}}$, Algorithm 1 Output: $f(\cdot)$, $\Phi(\cdot)$
, $\mathcal{M}(\cdot)$ **while** In defence **do**
  Load $f(\cdot)$
  for *sample* $\tilde{\mathbf{x}} \sim \mathcal{S}_{adv}$ **do**
    Discretize $\Phi(\tilde{\mathbf{x}})$
    if *D2A3* **then**
      One-hot encoding on $\Phi(\tilde{\mathbf{x}})$
      return $f(\Phi(\tilde{\mathbf{x}}))_{\boldsymbol{\theta}_s, \nabla_s}$
    else
      Obtain data transformation: $\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))$
      return $f(M(\Phi(\tilde{\mathbf{x}}^t))_{\boldsymbol{\theta}_s, \nabla_s}$

Let us discuss another salient feature of our D2A3, i.e., it relies on exploiting adversarial learning to enhance its defence capability, by optimizing the following objective function:

$$\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}_{\text{D2A3}} = \arg\min_{\boldsymbol{\theta}_s, \nabla_s} \overbrace{\mathbf{E}_{\mathbf{x}^t \sim \mathcal{S}_{\text{data}}} \mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\mathbf{x}^t)), \mathbf{y}^t)}^{\text{Training Loss}} + \overbrace{\mathbf{E}_{\tilde{\mathbf{x}}^t \sim \mathcal{S}_{\text{adv}}} \mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\tilde{\mathbf{x}}^t)), \mathbf{y}^t)}^{\text{Adversarial Training Loss}}. \quad (1)$$

It can be seen that our proposed objective function is composed of two parts: $\min \mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\mathbf{x}^t)), \mathbf{y}^t)$ and $\min \mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\tilde{\mathbf{x}}^t), \mathbf{y}^t)$. The motivation for Equation 1 is to achieve better performance for minimizing the empirical loss and also add robustness to the model as recommended by the work of [4,9].[3] As we discussed earlier, minimization of training loss – $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\mathbf{x}^t)), \mathbf{y}^t)$, can leverage the non-linearity of the discretization process and obtain a more accurate model $f(\cdot)$ [10]. However, only adding discretization into model $f(\cdot)$ is not enough as advocated in [4]. The adversarial training loss – $\min \mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\tilde{\mathbf{x}}^t), \mathbf{y}^t)$, is further added to add robustness to the model. It is based on the optimization problem which is to minimize the adversarial training loss given by the inner attack type (i.e., maximizing the adversarial loss by finding the adversarial version of input) [5]. During the defence, the input samples $\tilde{\mathbf{x}}^t$ will be firstly discretized and format with one-hot encoding as $\Phi(\tilde{\mathbf{x}}^t)$. The outline of D2A3 in defence mode is given in Algorithm 2.

3.2 D2A3N – Input Discretization as Defence

Unlike D2A3, where the input of the model $f(\cdot)$ is discretized, in D2A3N, the input to the model is the same as the original data format (continuous or categorical). It still obtains the cut-point $\phi^\alpha \in \Phi(\cdot)$ for each feature, using discretization method $\mathcal{C}$ on original data $\mathcal{S}_{\text{data}}$. After discretization, we use $\Phi(\cdot)$ to build a

[3] The effectiveness of adversarial training loss component is also studied in ablation study in Sect. 4.4.

defence strategy. The strategy revolves around finding the closest data points to the cut-points. These data points are replaced with the median of the bin. This is achieved by implementing the transformation function: $\mathcal{M}(.)$ as:

$$\mathcal{M}(\tilde{\mathbf{x}}^t) = \begin{cases} \arg\min_{\alpha} \|\tilde{\mathbf{x}}^t - \mu(\Phi(\cdot))^\alpha\|, & |\tilde{\mathbf{x}}^t - \phi^\alpha| < \epsilon, \\ \tilde{\mathbf{x}}^t, & |\tilde{\mathbf{x}}^t - \phi^\alpha| \geq \epsilon. \end{cases} \tag{2}$$

Here, $\mu(\Phi(\cdot))^\alpha$ denotes the median value of bin α and ϕ_i^α is its cut-points. In addition, we maintain a constraint, i.e., the absolute value between the data and the corresponding cut-point has to be smaller than the tiny constant value ϵ (minimum threshold value to change the bin and fixed as constant during the defence). Additionally, just like `D2A3`, the adversarial training is also augmented here to add the robustness to the model to minimize the adversarial risk as:

$$\mathcal{L}_{\texttt{D2A3N}}^{\boldsymbol{\theta}_s, \nabla_s} = \arg\min_{\boldsymbol{\theta}_s, \nabla_s} \overbrace{\mathbf{E}_{\mathbf{x}^t \sim \mathcal{S}_{\text{data}}} \mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\mathbf{x}^t)), \mathbf{y}^t)}^{\text{Training Loss}} + \overbrace{\mathbf{E}_{\tilde{\mathbf{x}}^t \sim \mathcal{S}_{\text{adv}}} \mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))), \mathbf{y}^t)}^{\text{Adversarial Training Loss}} \tag{3}$$

When the `D2A3N` is used in defence (Algorithm 2), the input is discretized first and then processed by the transformation function $\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))$ to map it back to a numeric value.

4 Experiments

In this section, let us empirically verify the effectiveness of `D2A3` and `D2A3N`. We will first implement white-box attacks to compare the robustness of both `D2A3` and `D2A3N` with state-of-the-art methods. Later, we will conduct an ablation study to determine the effect of adding adversarial training to `D2A3` and `D2A3N`.

4.1 Experimental Set-up

Datasets. We have used a total of 12 `UCI` classification datasets in our experiment. Note, all datasets are numeric in our experiments. Out of 12 datasets, 5 datasets have more than 10K samples and are denoted as `Large`, whereas 3 datasets have between $5-10$K samples and are denoted as `Medium`. The remaining 4 datasets have less than 5K samples and are denoted as `Small`. The statistics of the data are summarized in Sect. 4.1.

Adversarial Attack Setting and Evaluation Metric. We have made use of 3 commonly used white-box attack methods, which are common for tabular datasets, i.e., – `FGSM`, `Deepfool` and `LowProfool`. For each attack method, the architecture and the parameters of the target model are available, and the adversarial samples are directly generated. The step size of the `FGSM` is 0.1, and the maximum iteration for `Deepfool` and `LowProFool` is 50, which is the same as the default setting in their original implementation.

Table 1. Description of datasets.

Dataset	m	n	c	size	Dataset	m	n	c	size
SkinSegmentation	245057	4	2	Large	page-blocks	5473	11	5	Medium
connect-4	67557	43	3	Large	wall-following	5456	25	4	Medium
letter-recog	20000	17	26	Large	spambase	4601	58	2	Small
magic	19020	11	2	Large	diabetes	768	9	2	Small
sign	12546	9	3	Large	pid	768	9	2	Small
satellite	6435	37	6	Medium	credit-a	690	16	2	Small

Regarding the evaluation metrics, we have made use of **Accuracy** which generally determines the level of resistance of a defence mechanism against an adversarial attack. Notably, the **Standard Accuracy** for normal manner (no attack has occurred) and the **Robust Accuracy** for attack manner are used separately [9]. Moreover, the **Success Rate** of an adversarial attack is also used to measure the efficacy of the attack. According to the work [1], we define the **Success Rate** as:

$$\text{SR} = \frac{\sum_{i=1}^{\mathcal{Z}} |f(\mathbf{x}_i + \mathbf{r}) \neq f(\mathbf{x}_i)|}{\mathcal{Z}} \tag{4}$$

which is a ratio of successfully crafted adversarial samples (e.g., the samples are able to fool the model) and the total targeted samples ($\mathcal{Z}$).

Experiment Configuration and Baselines. Each dataset is split into train and test via 3-fold cross-validation. The models in `D2A3` and `D2A3N` are trained with 300 epochs and consist of a fully connected Artificial Neural Network (`ANN`) with `Relu` activation on 5 hidden layers and 1 `Softmax` for the output layer. We have used `Thermometer` encoding as a baseline for `D2A3`. Also, `D2A3` is implemented with three commonly used discretization methods i.e., `Equal-Width` (`D2A3-EW`), `Equal-Frequency` (`D2A3-EF`), and the `MDL` (`D2A3-MDL`).

For `D2A3N`, as discussed in Sect. 3.2, we have used two commonly used defence methods as baselines – `Madry` [5] and `Trades` [12]. Again, just like `D2A3`. we have used three commonly used discretization methods for `D2A3N`, leading to `D2A3N-EW`, `D2A3N-EF` and `D2A3N-MDL` respectively.

Other than baselines for `D2A3` and `D2A3N`, we have presented results with `Clean` model, which is the model without any defence mechanism. our code will be released in https://github.com/tulip-lab/open-code.

4.2 Model Discretization (`D2A3`) Results

Let us compare the performance of `D2A3` method with baseline methods by considering the three commonly used white-box attack methods. Of course, these attack methods have the full access to the architecture and parameters of the model $f(\cdot)$, and can directly generate the adversarial samples depending on any

Table 2. Performance comparison for D2A3.

Models	Standard Accuracy (Avg)	Robust Accuracy (Avg)			Success Rate(SR) (Avg)		
		FGSM	DeepFool	LowProFool	FGSM	DeepFool	LowProFool
Clean	0.895	0.813	0.272	0.282	0.135	0.793	0.769
Thermometer	0.836	0.759	0.761	0.726	0.196	0.160	0.194
D2A3-EF	0.870	0.792	0.768	0.722	0.153	0.165	0.217
D2A3-EW	0.854	0.793	0.788	0.753	0.136	0.143	0.161
D2A3-MDL	**0.918**	**0.897**	**0.884**	**0.869**	**0.000**	**0.025**	**0.030**

attack method. This is one reason why the **robust accuracy** of the clean model can deteriorate easily.

Table 2 shows the averaged comparison results on all datasets between D2A3 and other baseline methods, including the Clean model. Our experiment suggests that FGSM as an attack method is not particularly effective for tabular datasets. On the other hand, DeepFool and LowProFool are quite effective attack methods which result in lowering the accuracy of the model quite significantly.

It can be seen that Thermometer method as a defence is quite effective against the three forms of attacks. However, the difference between **standard accuracy** and **robust accuracy** is quite large.

It can be seen that three variants of D2A3 lead to an effective defence mechanism against three forms of attacks. It is important to note that since we are discretizing differently, hence the quality of model in terms of Standard Accuracy is different. As expected, discretization based on mdl leads to the best results in terms of **standard accuracy** (91.8%). Note, the more accurate the trained model is, the higher the quality of the adversarial samples it can generate. Now, it is encouraging to see that D2A3-MDL not only achieve a higher performance on **robust accuracy** but also has the smallest difference between **standard accuracy** and **robust accuracy**. A similar pattern can be seen in terms of the **success rate**, where the success rates of the three attacks are 0%, 2.5% and 3% respectively. These results are extremely encouraging, as they demonstrate that the model discretization algorithm D2A3 is an effective defence method against white-box attacks for tabular datasets. In Sect. 4.4, we will discuss the advantage of incorporating adversarial training loss within D2A3.

4.3 Input Discretization (D2A3N) Results

It can be seen from Table 3 that the overall performances of D2A3N trained with three forms of discretization – D2A3N-EF, D2A3N-EW, and D2A3N-MDL is better than the two standard baselines namely Madry and Trades. It is encouraging to see that without changing the dimensionality of the model $f(\cdot)$, D2A3N can largely help to resist the 3 kinds of adversarial attack. Particularly, the D2A3N-EW has shown higher **robust accuracy** and lower **success rate** compared to D2A3N-EF and D2A3N-MDL.

Table 3. Performance comparison for D2A3N.

Models	Standard Accuracy (Avg)	Robust Accuracy (Avg)			Success Rate(SR) (Avg)		
		FGSM	DeepFool	LowProFool	FGSM	DeepFool	LowProFool
Clean	**0.895**	0.813	0.272	0.282	0.135	0.793	0.769
Madry	0.892	0.808	0.459	0.496	0.129	0.553	0.500
Trades	0.895	0.817	0.456	0.307	0.127	0.549	0.739
D2A3N-EW	0.881	0.839	**0.683**	**0.536**	**0.103**	**0.291**	**0.458**
D2A3N-EF	0.871	0.807	0.558	0.532	0.164	0.449	0.487
D2A3N-MDL	0.874	**0.860**	0.588	0.488	0.138	0.416	0.514

Table 4. Ablation study on adversarial training.

Model		Robust Accuracy (Avg)			Success Rate(SR) (Avg)		
		FGSM	DeepFool	LowProFool	FGSM	DeepFool	LowProFool
D2A3	Adv. Train	**0.897**	**0.884**	**0.869**	**0.000**	**0.025**	**0.030**
	With Out Adv. Train	0.862	0.883	0.847	0.039	0.025	0.064
D2A3N	Adv. Train	**0.835**	**0.609**	**0.518**	**0.115**	**0.385**	**0.486**
	With Out Adv. Train	0.753	0.579	0.495	0.218	0.429	0.519

The better performance of D2A3N-EW discretization is surprising but can be explained. It is well known that EW discretization is more robust to the skewness of the data [8]. Since adversarial training leads to the original data being skewed, EW discretization based D2A3N is more robust to the adversarial attack.

4.4 Ablation Study on Adversarial Training

In this section, we conduct an ablation study to verify the effectiveness of the adversarial training step.

In Table 4, both D2A3 and D2A3N are tested to obtain the **robust accuracy** and **success rate** with and without the adversarial training, denoted as Adv.Train and With Out Adv. Train. It can be seen that without the adversarial training, both D2A3 and D2A3N leads to a slightly worse **robust accuracy** and **success rate** than with the adversarial training. This highlights the importance and necessity of the adversarial training within both D2A3 and D2A3N. It is also interesting to see that even without adversarial training, the performance of D2A3 and D2A3N is mostly better than the corresponding baselines in Tables 2 and 3.

5 Conclusion

In this paper, we studied the role of discretization in devising a defence strategy against adversarial attacks on tabular datasets. We showed that not only discretization can be effective, but it can also lead to better performance than

existing baselines. We proposed two algorithms namely `D2A3` and `D2A3N`, which leverages the cut-points obtained from discretization to devise a defence strategy. We evaluated the effectiveness of our proposed methods on 12 standard datasets and compared them against standard baselines of `Thermometer, Madry` and `Trades`. The effectiveness of `D2A3` and `D2A3N` clearly demonstrate the importance of discretization in warding-off adversarial attacks on tabular datasets. As future work, we are keen to explore theoretical justification over why `mdl` discretization is effective for `D2A3` and why `EW` is better than the other two forms of discretization for `D2A3N`. We are keen to extend our proposed methods to image datasets as well.

Acknowledgement. This research is partially supported by the National Natural Science Fund of China (Project No. 71871090).

References

1. Ballet, V., Renard, X., Aigrain, J., Laugel, T., Frossard, P., Detyniecki, M.: Imperceptible adversarial attacks on tabular data. arXiv preprint arXiv:1911.03274 (2019)
2. Buckman, J., Roy, A., Raffel, C., Goodfellow, I.: Thermometer encoding: one hot way to resist adversarial examples. In: International Conference on Learning Representations (2018)
3. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
4. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236 (2016)
5. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
6. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: DeepFool: a simple and accurate method to fool deep neural networks. In: CVPR (2016)
7. Qiu, S., Liu, Q., Zhou, S., Wu, C.: Review of artificial intelligence adversarial attack and defense technologies. Appl. Sci. **9**(5), 909 (2019)
8. Sulewski, P.: Equal-bin-width histogram versus equal-bin-count histogram. J. Appl. Stat. **48**(12), 2092–2111 (2021)
9. Yang, S., Guo, T., Wang, Y., Xu, C.: Adversarial robustness through disentangled representations. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 3145–3153 (2021)
10. Yang, Y., Webb, G.I.: Discretization for Naive-Bayes learning: managing discretization bias and variance. Mach. Learn. **74**(1), 39–74 (2009)
11. Zaidi, N.A., Du, Y., Webb, G.I.: On the effectiveness of discretizing quantitative attributes in linear classifiers. IEEE Access **8**, 198856–198871 (2020). https://doi.org/10.1109/ACCESS.2020.3034955
12. Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., Jordan, M.: Theoretically principled trade-off between robustness and accuracy. In: ICML (2019)
13. Zhang, Y., Zaidi, N.A., Zhou, J., Li, G.: GANBLR: a tabular data generation model. In: 2021 IEEE International Conference on Data Mining (ICDM), pp. 181–190. IEEE (2021)

14. Zhang, Y., Zaidi, N.A., Zhou, J., Li, G.: GANBLR++: incorporating capacity to generate numeric attributes and leveraging unrestricted Bayesian networks. In: Proceedings of the 2022 SIAM International Conference on Data Mining (2022)
15. Zhou, M., Wu, J., Liu, Y., Liu, S., Zhu, C.: DAST: data-free substitute training for adversarial attacks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 234–243 (2020)