

# Adversarial Training of Gradient-Boosted Decision Trees

Stefano Calzavara  
Università Ca' Foscari Venezia  
stefano.calzavara@unive.it

Claudio Lucchese  
Università Ca' Foscari Venezia  
claudio.lucchese@unive.it

Gabriele Tolomei  
Sapienza Università di Roma  
tolomei@di.uniroma1.it

## ABSTRACT

Adversarial training is a prominent approach to make machine learning (ML) models resilient to adversarial examples. Unfortunately, such approach assumes the use of differentiable learning models, hence it cannot be applied to relevant ML techniques, such as ensembles of decision trees. In this paper, we generalize adversarial training to gradient-boosted decision trees (GBDTs). Our experiments show that the performance of classifiers based on existing learning techniques either sharply decreases upon attack or is unsatisfactory in absence of attacks, while adversarial training provides a very good trade-off between resiliency to attacks and accuracy in the unattacked setting.

## CCS CONCEPTS

• **Information systems** → **Data mining**; • **Security and privacy** → *Formal methods and theory of security*.

## KEYWORDS

Adversarial learning; Decision trees; Tree ensembles

### ACM Reference Format:

Stefano Calzavara, Claudio Lucchese, and Gabriele Tolomei. 2019. Adversarial Training of Gradient-Boosted Decision Trees. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19)*, November 3–7, 2019, Beijing, China. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3357384.3358149>

## 1 INTRODUCTION

Machine learning (ML) has become a key component of many services we use on a daily basis, yet it is now acknowledged that ML is easily fooled by *adversarial examples*, i.e., carefully-crafted inputs designed to force prediction errors [5, 8, 10, 13]. Recall that traditional ML is based on the *empirical risk minimization* principle: given a training set  $\mathcal{D}_{\text{train}} = \{(\vec{x}_i, y_i)\}_{i=1}^n$  of correctly labeled instances and a set of hypotheses  $\mathcal{H}$ , ML identifies the function  $f \in \mathcal{H}$  which assigns to the training instances  $\vec{x}_i$  the best approximation of their labels  $y_i$ , with the understanding that  $f$  will generalize well to unseen data. Formally,  $f$  is found by minimizing a loss function  $\ell$ , which measures the cost of the prediction errors over  $\mathcal{D}_{\text{train}}$ :

$$f = \arg \min_{h \in \mathcal{H}} \sum_{i=1}^n \ell(h(\vec{x}_i), y_i). \quad (1)$$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or to publish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

CIKM'19, November 3–7, 2019, Beijing, China

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6976-3/19/11...\$15.00

<https://doi.org/10.1145/3357384.3358149>

Adversarial examples consist in malicious corruptions of existing instances that are crafted to be mis-classified by the ML model at test time. The main goal of *adversarial learning* research is making ML robust against adversarial examples, most notably through the development of novel attacker-aware learning algorithms [1, 6].

A prominent adversarial learning approach is called *adversarial training* [9]. Given a set of perturbations  $A$  defining the attacker's capabilities, adversarial training requires the minimization of the loss function  $\ell$  over  $\mathcal{D}_{\text{train}}$  under the conservative assumption that the attacker will always pick the perturbations in  $A$  which maximize the loss. More formally, adversarial training can be spelled out in terms of the following optimization problem:

$$f = \arg \min_{h \in \mathcal{H}} \sum_{i=1}^n \max_{\vec{z}_i \in A(\vec{x}_i)} \ell(h(\vec{z}_i), y_i). \quad (2)$$

Though the adversarial training formulation in Equation 2 is very elegant and appealing, it does not yield itself to standard optimization approaches, since it involves both a non-convex outer minimization problem and a non-concave inner maximization problem. Previous research tackled this issue by showing that, for deep neural networks, a pragmatically good approach is generating a single adversarial instance  $\vec{z}_i$  for each training instance  $\vec{x}_i$  by means of *gradient-based methods*, such as the Fast Gradient Sign Method [5] and its multi-step variants [8]. Unfortunately, such gradient-based approach assumes the use of differentiable learning models, hence it cannot be applied to relevant ML techniques like ensembles of *decision trees*, whose decision function is either flat or discontinuous everywhere [2]. This is a major shortcoming, since decision tree ensembles proved extremely effective for non-perceptual tasks and are one of the most successful techniques in Kaggle competitions [3].

*Contributions:* In this paper, we generalize adversarial training to decision tree ensembles. In particular we start from the observation that decision trees are based on known *thresholds*, which allow one to reduce the set of possible perturbations to a finite set without loss of generality. This enables the use of differentiable approximations of the maximum function in Equation 2, thus making the optimization problem tractable. We then implement our approach on top of the LightGBM framework for gradient-boosted decision trees (GBDTs) and compare its performance on a public dataset against two baselines: a traditional ensemble of GBDTs and an ensemble of trees trained through *adversarial boosting*, a state-of-the-art adversarial learning technique [7]. Our experiments show that the performance of classifiers based on existing learning techniques either sharply decreases upon attack or is unsatisfactory in absence of attacks, while adversarial training provides a very good trade-off between resiliency to attacks and accuracy in the unattacked setting.

## 2 ADVERSARIAL TREE LEARNING

Decision trees are binary trees which assign label predictions to instances by performing thresholding over feature values. A decision tree  $t$  is inductively defined as either a leaf containing a label prediction  $\hat{y}$  or a non-leaf node  $(f, v, t_l, t_r)$ , where  $f$  identifies a feature,  $v$  is the corresponding threshold and  $t_l, t_r$  are decision trees. At test time, the instance  $\vec{x}$  traverses the tree  $t$  until it reaches the leaf assigning its prediction. Specifically, for each visited non-leaf node  $(f, v, t_l, t_r)$ ,  $\vec{x}$  falls into the left tree  $t_l$  if  $x_f \leq v$ , while it falls into the right tree  $t_r$  otherwise.

Given a training set  $\mathcal{D}_{\text{train}} = \{(\vec{x}_i, y_i)\}_{i=1}^n$ , traditional algorithms for decision tree learning first compute the best label prediction on  $\mathcal{D}_{\text{train}}$  for a decision tree composed of a single leaf and then assess if it is possible to reduce the loss by replacing such leaf with a non-leaf node leading to two new leaves with predictions  $\hat{y}_l, \hat{y}_r$  respectively. The best replacement is found by an exhaustive search over all the possible features  $f$  and thresholds  $v$  occurring in  $\mathcal{D}_{\text{train}}$ , computing the predictions  $\hat{y}_l, \hat{y}_r$  which minimize the loss over  $\mathcal{D}_l = \{(\vec{x}_i, y_i) \in \mathcal{D}_{\text{train}} \mid x_f \leq v\}$  and  $\mathcal{D}_r = \mathcal{D}_{\text{train}} \setminus \mathcal{D}_l$ , respectively. The construction recursively proceeds on the new leaves and stops when it is not possible to further reduce the loss or a given termination criterion is met, e.g., the depth of the decision tree exceeds a fixed bound.

The key observation of our adversarial training approach is that, since it is possible to pre-compute all the possible thresholds which will occur in any decision tree built from a given training set  $\mathcal{D}_{\text{train}}$ , it is also possible to reduce the set of adversarial perturbations  $A$  to a finite set  $\mathbb{A}$  without loss of generality. Indeed, given any feature  $f$ , it is possible to identify a partition of its range in the intervals:

$$(-\infty, v_1], (v_1, v_2], \dots, (v_{k-1}, v_k], (v_k, +\infty),$$

where  $v_1, \dots, v_k$  are the possible values of the feature  $f$  observed in  $\mathcal{D}_{\text{train}}$ . By iterating such reasoning over all features, it is possible to identify a partition of the feature space in a finite set of *equivalence classes*, so that any two elements of the same class follow exactly the same path upon decision tree traversal. This allows us to replace the set of perturbations  $A$  in Equation 2 with the finite set  $\mathbb{A}$  of the (arbitrarily chosen) *representatives* of the equivalence classes.

One last issue in the optimization of Equation 2 is its non differentiability, due to maximum function, which prevents us to exploit any standard *gradient-based optimization* technique, e.g., back-propagation [12] or gradient boosting [4]. To overcome this issue in a classifier-independent way, we propose instead the use of the *LogSumExp* (LSE) function, which is a differentiable approximation of the maximum. Recall that, given a set of values  $\{w_1, \dots, w_n\}$ , we have  $LSE(\{w_1, \dots, w_n\}) = \log(\sum_{i=1}^n e^{w_i})$ . Hence, the adversarial training problem in Equation 2 can be approximated by the following differentiable optimization problem:

$$f = \arg \min_{h \in \mathcal{H}} \sum_{i=1}^n \log \left( \sum_{\vec{z}_i \in \mathbb{A}(\vec{x}_i)} e^{\ell(h(\vec{z}_i), y_i)} \right). \quad (3)$$

The loss in Eq. 3 is differentiable provided that the instance-level loss  $\ell$  is differentiable. This is the case for most instance-level losses. Remarkably, this method is agnostic from the choice of the specific machine learning algorithm, e.g., it can be applied to both deep

```
{
  "conditions": {
    "name": "hours_per_week",
    "operator": "greater_than",
    "value": 0 },
  "actions": {
    "name": "perturb_hours_per_week",
    "params": { "increment": 4,
               "cost": 4,
               "max-budget": 8 }
  }
}
```

**Table 1: Example of attack rule**

neural networks and decision trees. Indeed, in our experimental section, we apply the proposed technique to *gradient boosted decision trees* [4], which are a powerful machine learning approach largely ignored by the existing adversarial learning literature, despite its effectiveness at dealing with non-perceptual tasks.

## 3 CASE STUDY: CENSUS INCOME

### 3.1 Experimental Setup

We implemented the adversarial training approach formalized in Equation 3 on top of the LightGBM framework<sup>1</sup> for training gradient-boosted decision trees (GBDTs). We compare our proposal against two alternative approaches: (i) a traditional ensemble of GBDTs as provided by LightGBM, which does not implement any built-in protection against attacks at test time, but is supposed to provide optimal performance in absence of attacks; and (ii) the adversarial boosting approach, which makes GBDT classifiers attacker-aware by performing multiple boosting rounds over a training set extended with the adversarial examples which are most effective up to the previous boosting round. We implemented such approach on top of LightGBM, following the description in the original paper [7].

We implemented a simple domain specific language to express the attacker’s capabilities in terms of rewriting rules subject to a budget limitation. For example, the rule in Table 1 states that any person who actively works can cheat on her number of working hours by increasing their value by 4, provided that 4 units of budget are spent. Since the rule also enforces that no more than 8 units of budgets can be spent on such corruption attempts, adversarial perturbations can only add at most 8 working hours to the original feature value. Note that this simple model allows one to easily model both categorical and numerical features.

### 3.2 Training and Evaluation

In our experiments, we consider the Census Income dataset from the UCI ML Repository<sup>2</sup>, containing demographic information on approximately 48k people. The prediction task is to guess whether the yearly income of a person exceeds \$50,000 (positive class) or not (negative class), based on the following set of features: age, work class, sampling weight, education level, marital status, occupation, race, sex, capital gain, capital loss, working hours per week, and native country. After removing the limited number of instances including missing values, the dataset is split into 60%÷20%÷20% portions: a training set of ≈27k instances, a validation set and a test

<sup>1</sup><https://github.com/microsoft/LightGBM>

<sup>2</sup><https://archive.ics.uci.edu/ml/datasets/census+income>

	Precision		Recall		F <sub>1</sub>	
	No atk	Atk	No atk	Atk	No atk	Atk
GBDT	0.763	0.625	0.635	0.635	0.693	0.630
Adv. Boosting	<b>0.842</b>	<b>0.822</b>	0.505	0.505	0.631	0.626
Adv. Training	0.764	0.653	<b>0.663</b>	<b>0.663</b>	<b>0.710</b>	<b>0.658</b>

**Table 2: Experimental results (best results in boldface)**

set of  $\approx 9k$  instances each. These datasets are moderately skewed, with around 3/4 of the instances belonging to the negative class.

For each model of interest, we train a classifier over the training set and we use the validation set for hyperparameter tuning. We use the logistic loss  $\ell(h(\bar{x}), y) = \log(1 + e^{-yh(\bar{x})})$  as the underlying loss function and we eventually select the best performing classifier based on the value of its objective function on the validation set. Finally, we assess the performance of the resulting classifiers by computing the following standard measures on the test set: precision, recall, and  $F_1$ , both in absence and in presence of attacks. Evaluation under attack is performed by generating all the representatives of the equivalence classes mentioned in Section 2.

### 3.3 Threat Model

The goal of the attacker is fooling the classifier into mispredicting a yearly income higher than \$50,000 and improperly qualify for a loan, i.e., we let the attacker target just the negative instances. Specifically, we let the attacker corrupt the following features: *work class*: if the attacker never worked, she can pretend that she works without pay (cost = 1, max budget = 1); *marital status*: if the attacker is divorced or separated, she can pretend she never married (cost = 1, max budget = 1); *occupation*: any occupation can be presented as a generic “other service” (cost = 1, max budget = 1); *education level*: the attacker can cheat on her education level by lowering it at most twice (cost = 1, max budget = 2); *working hours per week*: the attacker can cheat on her working hours per week by adding at most 8 hours at chunks of 4 (cost = 4, max budget = 8); *capital gain*: the attacker can cheat on her capital gain by adding at most \$2,500 at chunks of \$500 (cost = 50, max budget = 250).

We assign different values of budget units to the attacker: 5, 15, 150 and 300. Notice that the two extreme values respectively model an attacker who can only corrupt the four categorical features and an attacker who has enough budget to fully run all the attacks.

### 3.4 Experimental Results

Table 2 shows our validity measures on the trained ML models, both in absence and in presence of attacks: for the sake of readability, we just report the results of the evaluation against the strongest attacker. Notice that the value of recall does not change upon attack, since the attacker targets just the negative instances.

The numbers provide several interesting insights: the first observation we make is that, although traditional GBDT was expected to provide the best performance in absence of attacks, it turns out that adversarial training actually works better in terms of  $F_1$ . We conjecture that this might be due to a phenomenon similar to *regularization* [11], i.e., the introduction of attacks in the optimization problem solved at training time provides better generalization power to the learned models, as if they had been exposed to additional training data. Though this might suggest that the attacker

is not powerful enough to perform effective attacks which significantly deviate from the training data, observe that the performance of standard GBDT sharply decreases upon attack, with the  $F_1$  score lowering from 0.693 to 0.630. As expected, adversarial training provides much better performance against attacks than standard GBDT, with the  $F_1$  score sitting at 0.658. Remarkably, adversarial training provides the best performing classifier in terms of  $F_1$  score both in absence and in presence of attacks, thus striking a very good trade-off between accuracy and security. The only measure where adversarial training is worse than its competitors is precision, where adversarial boosting provides a significant improvement. However, this is due to the fact that adversarial boosting completely sacrifices recall in the name of precision, as testified by the lowest value of  $F_1$ . The reason behind this behavior is that, since attacks only operate on the negative class, adversarial boosting sees a very large number of negative instances at training time, which push it into over-predicting the negative class to achieve a larger accuracy. This does not happen in the case of adversarial training, which optimizes a custom objective function and thus avoids such drawback of data augmentation techniques like adversarial boosting.

To provide a more complete picture of the performance of the trained models, we also plot their *security evaluation curves* for the  $F_1$  score in Figure 1. Since both adversarial boosting and adversarial training are parametric with respect to the choice of the attacker’s budget used for training, we provide one curve per training budget. We first highlight that adversarial boosting is quite sensitive to the chosen budget: when small training budgets are used its performance significantly decreases under stronger attacks, though this phenomenon essentially disappears when using large training budgets. In our experiments adversarial boosting exhibits close yet worse performance than standard GBDT in terms of  $F_1$  score, while the quality of the proposed adversarial training approach is consistently better than its competitors, independently of the budget given to the attacker. Last, we show in Figure 2 the value of precision and recall for the different models, when varying the attacker’s budget considered for training and for evaluation. The plots confirm that adversarial boosting is highly unbalanced towards precision, while GBDT and adversarial training provide a much better trade-off between the two measures; still, adversarial training is consistently better than GBDT in all the plots.

## 4 CONCLUSION

Adversarial training is a popular approach to make ML models resilient to adversarial examples, yet its original formulation cannot be applied to train decision tree ensembles. In this paper, we showed how to generalize such approach to this important class of models and we validated its effectiveness on a public dataset. As future work, we plan to extend our experimental evaluation to additional datasets and explore novel adversarial learning techniques based on revised algorithms for decision tree learning, which optimize the construction by avoiding the generation of the full set of the attack representatives.

## ACKNOWLEDGMENTS

This work was supported in part by the MIUR under grant “Dipartimenti di eccellenza 2018-2022” of the Department of Computer Science of Sapienza University.

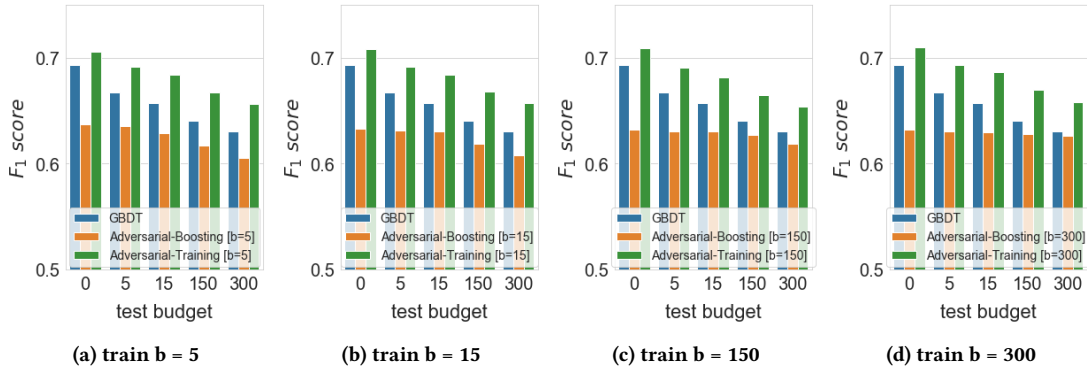


Figure 1:  $F_1$  score measured on  $\mathcal{D}_{\text{test}}$  against different train/test attacker’s budgets.

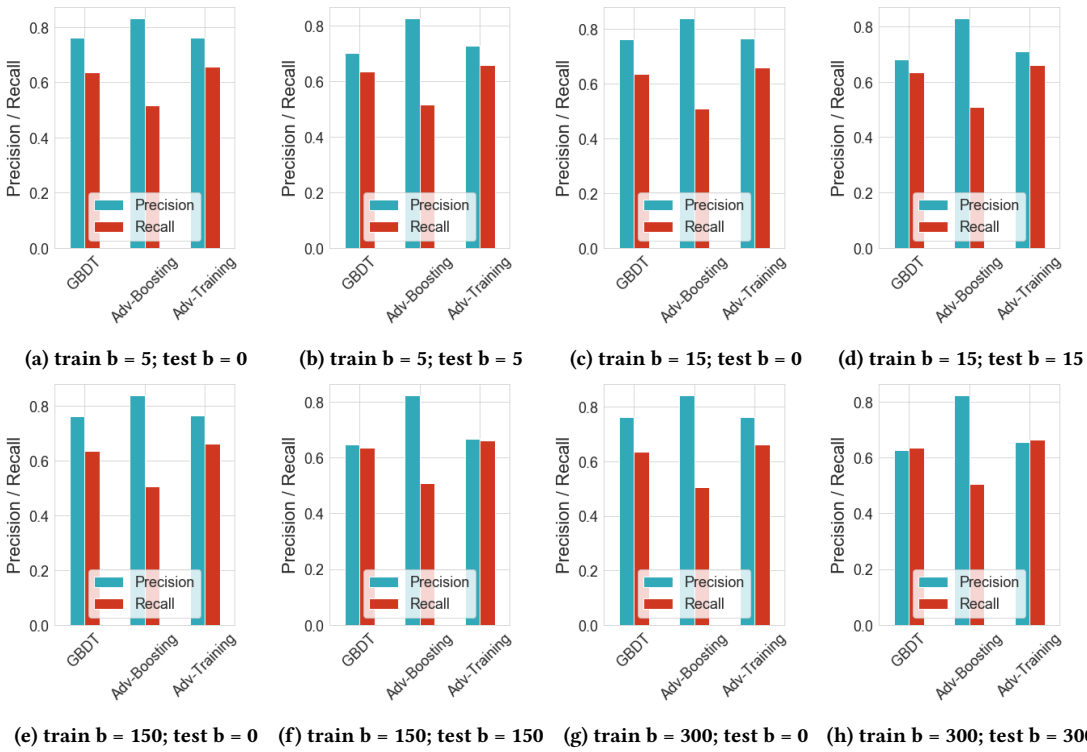


Figure 2: Precision and recall measured on  $\mathcal{D}_{\text{test}}$  against different train/test attacker’s budgets.

## REFERENCES

- [1] BIGGIO, B., AND ROLI, F. Wild patterns: Ten years after the rise of adversarial machine learning. *CoRR abs/1712.03141* (2017).
- [2] BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., AND STONE, C. J. *Classification and Regression Trees*. Wadsworth, 1984.
- [3] CHOLLET, F. *Deep Learning with Python*, 1st ed. Manning Publications Co., Greenwich, CT, USA, 2017.
- [4] FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.
- [5] GOODFELLOW, I. J., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples. *CoRR abs/1412.6572* (2014).
- [6] HUANG, L., JOSEPH, A. D., NELSON, B., RUBINSTEIN, B. I. P., AND TYGAR, J. D. Adversarial machine learning. In *AISeC* (2011), pp. 43–58.
- [7] KANTCHELIAN, A., TYGAR, J. D., AND JOSEPH, A. D. Evasion and hardening of tree ensemble classifiers. In *ICML* (2016), pp. 2387–2396.
- [8] KURAKIN, A., GOODFELLOW, I. J., AND BENGIO, S. Adversarial machine learning at scale. *CoRR abs/1611.01236* (2016).
- [9] MADRY, A., MAKELOV, A., SCHMIDT, L., TSIPRAS, D., AND VLADU, A. Towards deep learning models resistant to adversarial attacks. *CoRR abs/1706.06083* (2017).
- [10] MOOSAVI-DEZFOOLI, S., FAWZI, A., AND FROSSARD, P. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR* (2016), pp. 2574–2582.
- [11] NG, A. Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *ICML '04* (2004), ACM, pp. 78–.
- [12] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. Cambridge, MA, USA, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362.
- [13] SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I. J., AND FERGUS, R. Intriguing properties of neural networks. *CoRR abs/1312.6199* (2013).