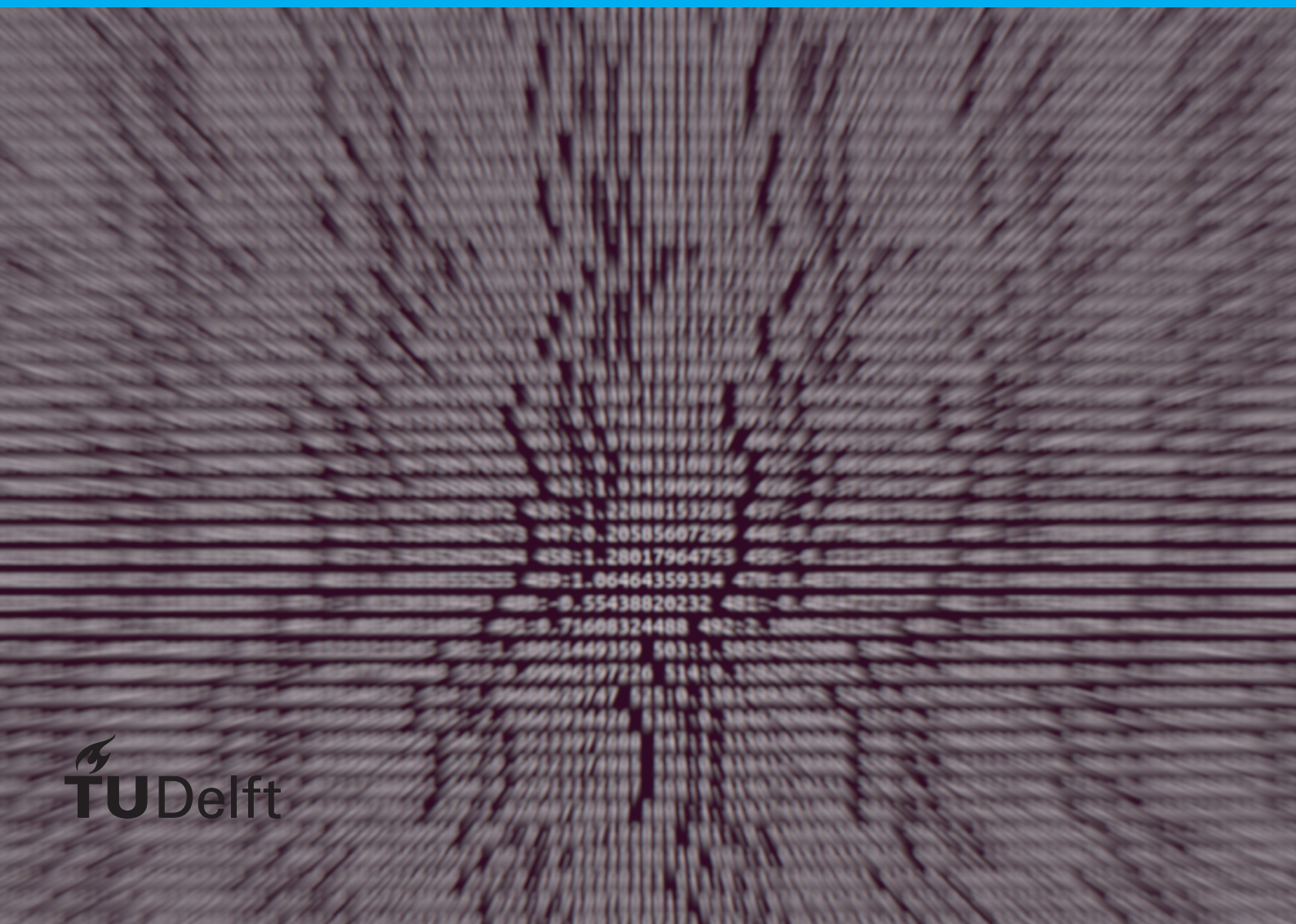
Improving Adversarial Attacks on Decision Tree Ensembles

Exploring the impact of starting points on attack performance

Max Pigmans



TU Delft

Improving Adversarial Attacks on Decision Tree Ensembles

Exploring the impact of starting points on attack performance

by

Max Pigmans

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday April 15, 2024 at 11:00 AM.

Student number:	4483669		
Project duration:	November, 2020 – April 15, 2024		
Thesis committee:	S. Verwer,	Associate Professor,	TU Delft, supervisor
	A. Anand,	Associate Professor,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

You are reading my masters thesis, Improving Adversarial Attacks on Decision Tree Ensembles: Exploring the impact of starting points on attack performance. This thesis was written to complete my masters in Computer Science with a Cybersecurity specialization, at the Delft University of Technology. The thesis investigates the effects of starting point choices in adversarial attacks on tree ensemble models.

I am very thankful to my supervisor Sicco Verwer for his support during my thesis. Even when things were not going as planned, you always showed support and provided the flexibility that was dearly needed at times. Thank you. I would also like to thank Leonie Boortman and Michel Rodrigues for their help; it was always helpful to talk to you.

It's been a long ride and there have been some very difficult periods in it for me. I've learnt a lot of things about myself that I didn't expect I would during my thesis. I would like to thank all of my friends, my girlfriend, and my family for their support.

I hope you enjoy reading my work.

Max Pigmans
Delft, April 2024

Abstract

Most of the adversarial attacks suitable for attacking decision tree ensembles work by doing multiple local searches from randomly selected starting points, around the to be attacked victim. In this thesis we investigate the impact of these starting points on the performance of the attack, and find that the starting points significantly impact the performance: some do much better than others. However, we do find that this is not the case for all attacked points, as there are large differences between points in how difficult they are to attack and for all datasets some points are always optimally attacked.

We compare the baseline randomly selected points to three alternative strategies. First, we try alternate random distributions, playing with both the standard deviation, to create a more narrow cone around the victim point, and mean, creating bimodal distributions further away from the victim point. We find that for some datasets these can give up to 5-7% improved performance on subsets of the dataset, but these improvements do not generalize to the remainder of the dataset. In general, as long as the distribution is wide enough to successfully find starting points we do not find a substantial performance change.

Secondly, we try to remove the randomness and attack from a fixed direction. For the simpler datasets we find it is possible for a starting direction to perform better than random starting points, but for larger datasets performance becomes much worse. We also try an attack from all main directions around the victim point, which we find performs much worse than 5-20 times fewer random points.

Lastly, we create an attack strategy where we select the closest points that scored well on previously attacked victims. We find that on smaller test sets this gets outperformed by the baseline, but when we extend the attack and give more possible previously well performing starting points we match or outperform the baseline slightly.

Contents

1 Introduction

As machine learning is becoming more and more popular and applied in more and more fields, the amount of sensitive and important decisions made by these algorithms grows too. From banks, healthcare, to the government, machine learning tools are increasingly used to make decisions that can have massive impacts on our lives. It is therefore important that we can trust these models to be reliable and robust, and to not be easily fooled by any malicious parties who might stand to gain from tricking our systems. Unfortunately, researchers have discovered [13] that machine learning models are susceptible to 'adversarial examples', inputs intentionally changed by a very small amount in order to fool the model.

As an example, think of a spam filter that might rely (partially) on which country the email is being sent from to determine how likely it is that an email is spam. By only changing what country the spam is being sent from, spammers might be able to bypass the filter easily.

While most research in the field of adversarial examples has focused on machine learning models such as neural networks, decision trees and decision tree ensembles have received much less work. Decision trees are characterized by their interpretability, since it is much easier to explain why the model comes to a certain conclusion than with neural networks. This interpretability makes them especially likely to be used for key decision making.

This makes research into the security of these decision trees and decision trees ensembles especially relevant today, as more and more companies and government bodies need to be able to trust in the reliability of these systems. One of the key ways we have to defend our models against such adversarial attacks involves the creation of adversarial examples during the training process, to learn from and to improve the robustness of our model. This process relies on fast creation of high-quality adversarial examples. Most current work in the field of adversarial attacks on decision tree ensembles involves randomly generated starting points for the attack. We believe this introduces large variance into the attack and our intuition says that it should be possible to find a smarter method of finding starting points, that can speed up the process of generating adversarial examples.

1.1. Research questions

The main goal of this thesis is to answer the following question:

How can we improve adversarial attacks on tree ensemble models by changing the way we generate the starting points of the attack

In order to better answer this question we will divide it into three parts. First we will look at whether this is actually an area that could lead to improvement by checking whether the starting points are impactful for performance. Second, we will investigate what happens when we tweak the random generation. Lastly, we look at whether we can create points using a completely different method than random generation.

1. What is the impact of the starting points of attacks on their performance

2. What is the effect of different random generation methods on the performance of random starting points
3. Can we improve on randomly generated starting points by using non-random distribution based methods of selecting starting points

1.2. Overview

Our main contributions are:

- A quantification of the impact of the starting points on the attacks performance.
- An investigation in how the specific parameters of the random distribution used impact the performance.
- New strategies for generating starting points with no/less randomness, by attacking from a fixed direction and by attacking from previously well-performing points, and an evaluation of their performances.

The code used and created during the thesis will be made available online.

1.3. Outline

The rest of this thesis is structured as follows. In chapter 2 we give background information on machine learning, especially regarding decision tree (ensembles) and adversarial attacks. In chapter 3 we investigate the impact of the starting points on the performance of the attack, both on a dataset wide level and per individual point. In chapter 4 we investigate different random generation methods and altogether different strategies for generating starting points. Finally, in chapters 5 and 6 we discuss and conclude our work.

2 Background

In this chapter we provide the background knowledge on machine learning, decision tree learning and adversarial attacks on machine learning models that might be needed for the rest of the thesis. We first introduce machine learning and decision trees & decision tree ensembles. Next, we introduce the different distance norms or distance metrics used in the field to calculate adversarial distance. Then, we give a summary of important works in the field of adversarial examples and adversarial examples for tree ensembles, and finally we determine the research gaps that we are focusing on.

2.1. Machine learning

Machine learning is a hot topic with a lot of ongoing research, and machine learning is becoming more and more widespread and applied in more and more fields. At its core, machine learning is about learning models from data using one of many various techniques, in order to give some output or prediction new previously unseen data. Rather than directly telling the computer how to interpret certain data, we let the machine decide what parts of the data to look at in order to draw certain conclusions. While there are many types of machine learning models, such as logistic regression, decision trees, and neural networks, we focus in this work on decision trees and more specifically decision tree ensembles.

2.1.1. Decision trees

Decision trees are relatively simple to understand machine learning models, that are often used when interpretability of the model predictions is important. Decision trees are tree models consisting of multiple layers of nodes, each of which does a simple test on the input and depending on the result goes to the left child or the right child of the node. Once a leaf node is reached an outcome is predicted. We give a simple example of a decision tree in figure 2.1, that could be used to detect whether an incoming email should be marked as spam or not. We start at the top node (square) of the model, and ask if the sender of the email is unknown. If we know the sender, we take the left path to the leaf node (circle), and mark the email as safe. If the sender is unknown we take the right path, and go to the next node and ask whether the email subject contains the word 'sale'. If it does, we take the right path to the leaf node and mark the email as spam, and if it does not we take the left path to the next node, and continue until we reach a leaf node.

The example decision tree has a depth of 4, as it contains 4 layers of leaves and/or nodes. In general, the deeper the model the less interpretable the model gets, as we need to consider more and more splitting decisions to determine why we have reached a certain leaf node.

Tree ensembles

Decision trees can offer good interpretability depending on their depth, but are prone to overfitting and struggle with high-dimensional data. In order to address this and make the model more effective on new data decision trees are often grouped in ensembles. The two main approaches are random forests [1] and gradient boosting [7]. Ensemble methods lose interpretability in exchange for this performance gain however, as reasoning over the decision made by a large ensemble of distinct trees quickly becomes difficult.

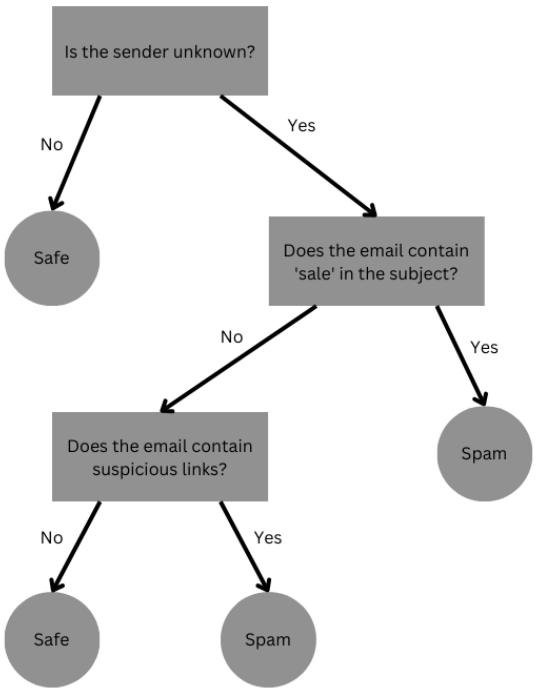


Figure 2.1: An example decision tree for the detection of spam. The tree starts at the top, with each node posing a question on the input data. Based on the answer a different node (square) or leaf (circle) is reached, and whenever we reach a leaf node we take its result as our answer, either 'safe' or 'spam'.

Random forests work by creating many diverse trees and getting the final result via majority voting, i.e. the predicted class is the most predicted class across all the decision trees. Each tree in the forest is trained on smaller training sets, each a subset from the original training set. By using bagging (bootstrap aggregation) and random feature selection the trees are made diverse, meaning that they do not all contain the same decision. Averaging the result over all the trees also helps to reduce noise, making the model more robust.

Gradient boosting works differently from random forests in that it is built in a stage-wise fashion. Trees are created one by one in such a way that each new tree tries to correct the weakness (being the prediction errors) of the tree(s) before it. By repeating this process the prediction error is slowly reduced and the model's predictive capacity increases. Because new trees focus on the weakness it is likely that they implicitly become diverse, focusing on different aspects of the data in order to best reduce the prediction error.

2.2. Distance norms

Distance norms represent different ways of measuring the magnitude of vectors. In other words, norms are used to measure the distance between two different points by measuring the vector between them. There are four commonly used norms: L_0, L_1, L_2 and L-infinity (L_{inf}). There is no universally best norm, and the best norm to be used depends on the specific problem at hand.

1. The L_0 norm counts the number of non-zero features in the vector; $\sum_{i=0}^{n} x_i \neq x_i'$
2. The L_1 norm measures the total magnitude of all features in the vector. Also known as Manhattan distance, it is calculated by summing up the absolute value of each feature. $\sum_{i=0}^{n} |x_i - x_i'|$
3. The L_2 norm measures the Euclidean distance is calculated as the square root of the squares of each of the features in the vector. $\sqrt{\sum_{i=0}^{n}(x_i - x_i')^2}$
4. The $L_{infinity}$ or L_{inf} norm measures the biggest individual feature in the vector. $max_i|x_i - x_i'|$

In the context of adversarial examples, the L_0 norm minimizes the amount of features that have to be changed to create the adversarial example; Papernot[12] uses this norm in their work on defensive distillation, one of the first defenses against adversarial examples. The L_1 and L_2 norms measure the total amount of change needed across all features, but can be difficult to compute based on the types of features. The L_2 norm was used in the first work that showed the existance of adversarial examples by Szegedy. The L_{inf} norm encourages many small changes over few big changes; Warde and Goodfellow[14] argue in their 2016 work on the robustness of neural networks that the L_{inf} norm is best. In general, there is no consensus on which norm to use and most works report both on the L_2 and L_{inf} norm.

2.3. Adversarial attacks

Initially machine learning algorithms were focused on improving the performance on the model, with little regard for security. In 2014 Szegedy et al. [13] showed that machine learning models are susceptible to so called ”adversarial examples”. By doing small imperceptible changes to an input image they found that they could change the model’s prediction to an arbitrary different class. This is an example of an evasion attack, where a malicious user changes their (malicious) input data slightly so that the machine learning model incorrectly predicts a benign class.

One of the most well known examples of an adversarial attack is by Goodfellow et al. [8] in their work on the Fast Gradient Sign Method. Seen in figure 2.2, they add a very small vector to the original image of a panda, roughly corresponding to only changing (some) of the smallest bits of the values of the image, meaning that for a human the two images are the same. However, the machine learning model they attack (GoogLeNet) is now 99.3% convinced that the panda has become a gibbon. While most examples of adversarial attack will use images because it makes the changes (or lack thereof) easy to show, the same principles apply to models with any other data type.

In the rest of this section I will give an overview of the development of attacks in general, and then focus on the attacks suitable for decision tree ensembles. Finally, we will discuss some interesting findings in the field related to the transferability of adversarial examples between models, and the universality of perturbations between both different images in the same model, as well as between different models.

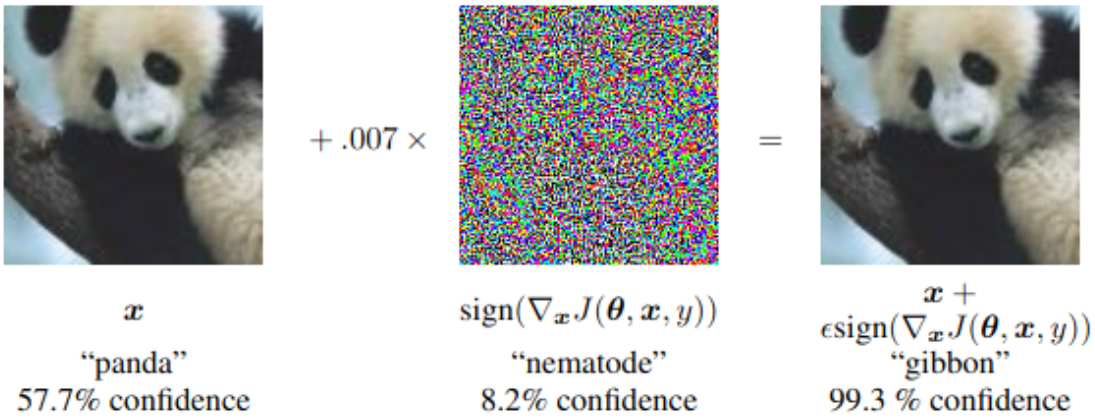


Figure 2.2: Famous example of an adverarial attack by Goodfellow et al. [8]. They show how adding a small amount of noise (middle image) to an image of a panda (left) can cause the model, GoogLeNet, to switch from predicting ”panda” to ”gibbon” (right), while the human eye cannot see a change in the picture.

2.3.1. Attacks

In this section I will give an overview of the development of attacks on machine learning models. While there are many more attacks, we focus on the key works in the field to give an overview.

Szegedy's attack
As mentioned earlier, Szegedy et al. [13] first showed the concept of adversarial examples in 2014. They showed that by doing very small perturbations to an original image, they could make a classifier model give completely different predictions. They calculated the perturbations by doing a computationally expensive nonlinear optimization function, and focused on the existence of adversarial examples and not on the performance of their algorithm.

Fast Gradient Sign Method
As the first work by Szegedy used a relatively computationally expensive optimization method and was not practical to use. In 2015, Goodfellow et al. [8] created the 'Fast Gradient Sign Method' attack that is much faster to compute. They generate adversarial examples by adding to the original input a small vector based on the gradient of the cost function. Essentially, they move every pixel a small amount in the direction of the gradient, which can be quickly calculated using back propagation, making this attack much more feasible to use. This allowed the method to be used for testing and/or training of more robust machine learning models.

Carlini & Wagner
In 2016, Carlini and Wagner[3] created a class of attacks for each of the L_{inf}, L_0 and L_2 norms that succeed in finding adversarial examples for all images against standard models as well as all defenses at the time. Still seen as the state-of-the-art approach today, although many defenses proposed since are effective against the basic implementation. The L_2 attack uses random starting point for its attack, selected from a ball around the best known adversarial example, in order to escape local minima.

2.3.2. Attacks on decision tree ensembles
In the rest of this thesis we focus on decision tree ensembles, so in this section we will explore some of the attacks suitable for attacking decision trees and decision tree ensembles. Many of the earlier works are black-box attacks, that do not rely on the inner workings of the model in any way, and are thus applicable to tree ensembles.

MILP/Kantchelian 2015
In 2016 Kantchelian et al.[9] give an optimal MILP-based solution for computing the exact minimal adversarial example in tree ensembles. They find that in general, finding the exact minimal adversarial perturbation for tree ensembles is NP-complete.

Boundary attack
In 2016 Brendel et al.[2] introduce the Boundary Attack, an effective black-box attack. The boundary attack was not the first black box attack, but it was the first attack suitable for larger and more complex models and datasets. The idea of the boundary attack is to start at an adversarial point, probably quite far away from the target victim point, and to iteratively greedily move closer towards the victim point. In each step they first make a random orthogonal step, and then a second step in the direction of the victim. They only accept steps that successfully move the adversarial point closer to the victim point while remaining adversarial. They only worked in the L_2 distance norm. The implementation starts either from a random starting point, randomly drawn from a uniform distribution in the entire search space, or from a given (adversarial) point.

Sign-OPT
In 2020 Cheng et al.[6] introduce the Sign-OPT attack, another black-box attack focused on being query efficient. This means that the are not only looking at the adversarial distances of the created examples, but also consider how many times they need to query the target model in order to obtain the results. It is an improvement on their earlier work in the field, and not the first work to focus on this type of attack, but this and their previous work are effective black-box attacks that are also suitable for discrete machine learning models, such as gradient boosted decision trees and random forests. The implementation starts from a random starting point, drawn from a normal distribution centered on the victim point.

Hop Skip Jump Attack
Another query-efficient black box attack made in 2020, the Hop Skip Jump Attack (HSJA) is an attack by Chen et al.[5] that is quite similar in idea to the boundary attack, but mainly differs in how they find the next step. While the boundary attack randomly tried orthogonal steps until they found an improvement, the HSJA attack instead uses the information gained by rejected perturbations to instead estimate the direction of the gradient at the current location, greatly reducing the number of model queries needed to find a new (better) point. The implementation starts from a random starting point, drawn from a uniform distribution around the victim point.

Leaf Tuple attack
In 2020 Zhang et al.[15] published the Leaf Tuple attack (LT-attack), the first effective white box attack on tree ensembles. They make use of the structure of trees to more efficiently traverse the search space, by transforming the continuous input space into a discrete 'leaf tuple' space. Essentially, they combine the decision thresholds of all trees in the ensemble and divide the search space according to these thresholds, meaning that every unique combination of tree leaves that an input could end up in is its own leaf tuple. Then, they devise an algorithm for moving from one leaf tuple to its direct neighbours, meaning they move from the current leaf tuple (and its corresponding value in the search space) to the best neighbouring leaf tuple. The implementation starts from a random starting point, drawn from a normal distribution centered on the victim point.

2.3.3. Transferability
In 2016 Papernot et al. [11] found that adversarial examples transfer. What this means is that (some) adversarial examples produced to mislead a certain machine learning model can also mislead other machine learning models, even when trained on different subsets of the same training set. They found that this works both between different models of the same type, 'intra-technique', as well as 'cross-technique', between different classes of machine learning models, for example from decision trees to neural networks.

2.3.4. Universality between models
Moosavi-Dezfooli et al.[10] also show the existence of 'universal perturbations'. These are universal perturbations for a given dataset, that cause (many) test images to be misclassified all using the same perturbation. In addition, they show that these universal perturbations not only transfer well between images in the same model but also to other models.

2.4. Research gap
Based on the works we have evaluated in the field of adversarial examples, specifically on decision tree ensembles, we identify the following research gaps:

- Many attacks use random starting points to initialize their attacks. While the exact way these are generated varies, to the best of our knowledge no work has been done in measuring the impact of the starting points used.
- It is not known how alternative methods of choosing the starting points compares to the randomly generated starting points.
- Points in the model are tested individually, and it is unknown to what extent we could re-use information learnt during the attack on one point when attacking a second point.

3

Significance of the starting points

In this chapter we will show that random starting points have a large influence on the performance of the attacks. We first discuss the datasets that we use for the experiments. Next, we will answer the question on a dataset-level by showing that the outcome of the attack is dependant on the starting points in section 3.2. Afterwards, we will look at the difference in performance on individual victim points in section 3.3.

3.1. Datasets

We describe the used datasets and their main characteristics in Table 3.1, along with the characteristics of the used GBDT models. We use the pre-trained models from Chen et al. [4], which have been used by recent works. These datasets are selected from datasets used by related works, to provide a variety of datasets with both binary and multi-class classification, and low and high feature counts.

Dataset	Features	Classes	Trees	Depth
2-6 mnist	784	2	1000	4
Breast cancer	10	2	4	6
covtype	54	7	160	8
diabetes	8	2	20	5
fashion mnist	784	10	400	8
higgs	28	2	300	8
ijcnn	22	2	60	8
mnist	784	10	400	8
webspam	254	2	100	8

Table 3.1: Overview of dataset and model characteristics.

3.2. Dataset-wide performance

First, we will show that the starting point has impact on the outcome of the attack, meaning that some starting points perform better than others. In order to show this we will attack the same victims with a varied number of randomly selected points. If the selected starting points have impact then we would expect the attack to perform better when attacking with more points, since the chance is higher to select a good starting point. Conversely, if the starting point does not matter then we would expect performance to be unchanged when using more starting points.

3.2.1. Experimental setup

We use Zhang's LT attack on all 9 datasets described in section 3.1, using the pre-trained GDBT models from Chen et al. We have modified the LT-attack to use random seeds on each run, so that we can run the attack multiple times and average out the variance of the randomness. For each dataset we attack

the first 100 points, and take the average adversarial distance in both the L_2 and L_{inf} norms across these 100 victims. We choose to attack 100 points for two reasons: firstly, not all datasets have a large test set and we choose this number so that we can apply the same attack on all datasets. We vary the vary the number of random starting points per victim point from 2 to 100. Due to their long runtime, we only go up to 50 starting points on the bmnist, fmnist, and mnist datasets. In order to average out the variance of the chosen random points, especially for the attacks with fewer random points, we repeat each attack on each dataset 50 times.

3.2.2. Results

The resulting scores are plotted in figure 3.1. We plot for each dataset a graph for both the l_{inf} and l_2 norms, with the X-axis representing the number of starting points used for each victim and the Y-axis the adversarial distance result of the attack. Each X-axis number in the bar charts shows the results of the 50 attacks done using that configuration.

We see that the average L_{inf} distance decreases as the number of random points increases, in all graphs. For most datasets we see that the rate of improvement decreases as more random points are used, but performance does continue to improve except for in the breast cancer dataset. As we would expect, we also see that the variance decreases as the attacks use more starting points. For many datasets the final attack has almost no variance in performance within the 50 runs, and while the attacks with 2 or 5 starting points generally have some outlier points, the later attacks usually do not.

3.2.3. Discussion

We see that we find lower adversarial distances, meaning better adversarial points, as we use more and more starting points. While the rate of improvement does slow down as the number of starting points increases, for all datasets other than breast cancer we do continue to improve all the way up to 100 random starting points. We conclude from this that the starting points do have a significant impact on the performance of the attack.

Investigating the results of the breast cancer dataset, which seems to reach a plateau in the adversarial distance found, with the L_2 norm already (rarely) at 10 starting points and almost always at 100, and with the L_{inf} norm from 50 points and reliably at 100. When we compare the found distances here to the optimal adversarial distance found by the MILP attack, we see that this is because we find optimal adversarial examples. Adding more starting points only makes it more consistent to find the optimum, but we can not perform any better.

We also see that the results in the L_2 and L_{inf} norms are very comparable. While we can not compare the adversarial distances directly, we can compare the shape of the graphs and we see that these all match. In the rest of this work, we will therefore focus on the L_{inf} norm in order to improve visual clarity. We choose L_{inf} over L_2 for two main reasons: firstly, we find the L_{inf} results slightly more consistent in the graphs in igure 3.1. While the L_{inf} graphs more consistently have some outliers, it happens less frequently to have many outliers than with the L_2 norm. Secondly, we find from the code implementation that some of the inner optimization loops are designed around the L_{inf}-norm and are afterwards checked to ensure the results are valid in the L_2 norm, reinforcing our decision to focus on L_{inf}.

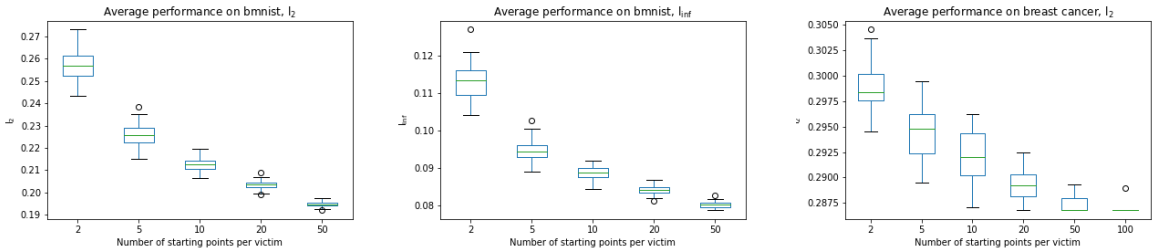
3.2.4. Conclusion

In this section we attacked 9 datasets with a varying number of (randomly generated) stating points, in order to evaluate whether the starting points have a significant impact on the performance of the attack. We find that the number of starting points has a significant impact on improving the performance. Additionally, we find that for as many starting points as we tested performance continues to improve until the optimum results are reached.

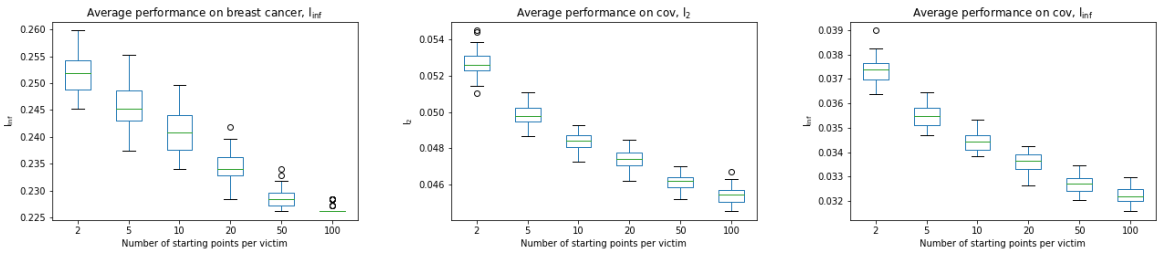
We also find that the results of the L_{inf} and L_2 norms are nearly identical and therefore will focus on only the L_{inf} norm in the rest of this thesis, for brevity.

3.3. Individual points

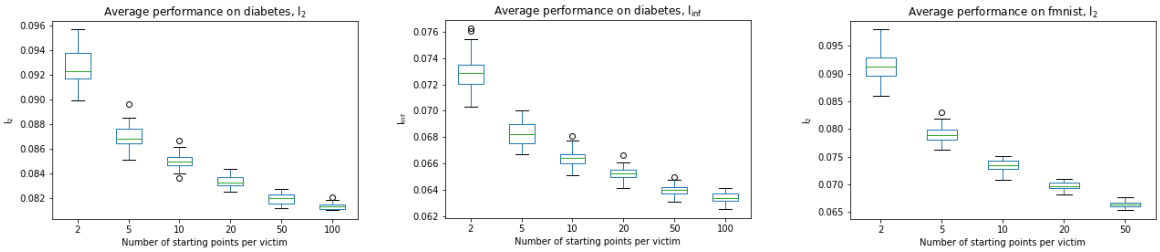
The previous section focused on the performance across the entire dataset, and showed that there is a significant difference in performance depending on the number of random points used. In this section we will look at the individual points that get attacked, and determine whether the seen performance



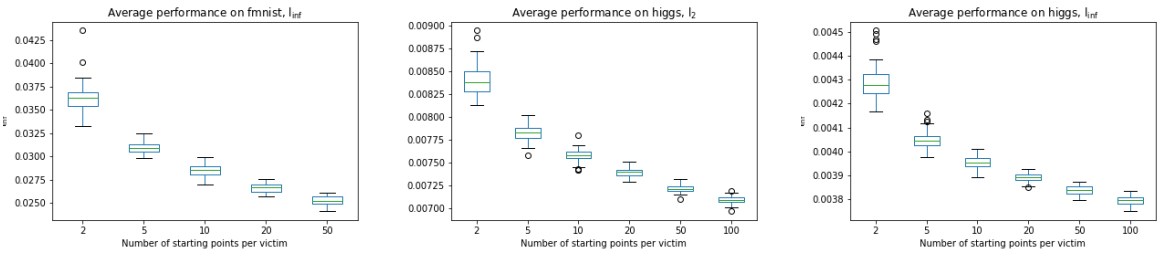
(a) bmnist dataset using the L_2 norm (b) bmnist dataset using the L_{inf} norm (c) Breast cancer dataset using the L_2 norm



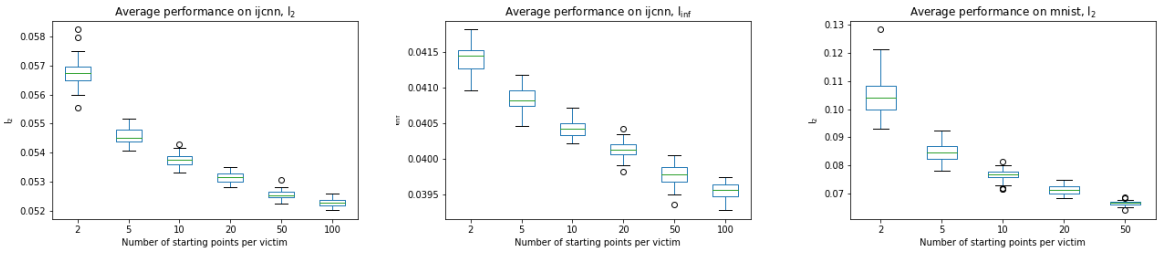
(d) Breast cancer dataset using the L_{inf} norm (e) covtype dataset using the L_2 norm (f) covtype dataset using the L_{inf} norm



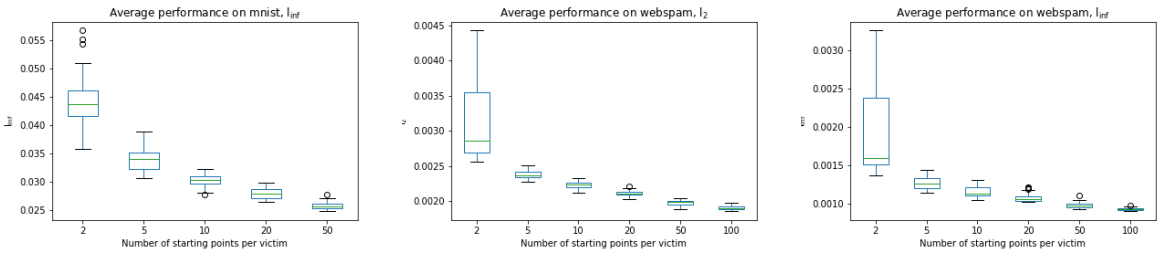
(g) diabetes dataset using the L_2 norm (h) diabetes dataset using the L_{inf} norm (i) fmnist dataset using the L_2 norm



(j) fmnist dataset using the L_{inf} norm (k) higgs dataset using the L_2 norm (l) higgs dataset using the L_{inf} norm



(m) ijcnn dataset using the L_2 norm (n) ijcnn dataset using the L_{inf} norm (o) mnist dataset using the L_2 norm



(p) mnist dataset using the L_{inf} norm (q) webspam dataset using the L_2 norm (r) webspam dataset using the L_{inf} norm

Figure 3.1: Plots showing the average adversarial distance across the entire dataset, as the number of random points used per victim increases. For each dataset the first 100 test set points are attacked, and each configuration is repeated 50 times to obtain the shown variance. Each dataset is attacked twice, once using the L_2 norm and once using the L_{inf} norm.

difference holds for all points or for some of the attacked points. We look at the spread of adversarial distance values in section 3.3.2. While this gives a good overview of how the adversarial distances are spread out, it can be difficult to see when victims are optimally solved compared to when they are almost-optimally solved. Therefore, we also look at whether the found adversarial distance is optimal or not, rather than its actual value in section 3.3.3.

3.3.1. Experimental setup

In order to get the performance of individual points we run the following experiment. We attack all 9 datasets described in section 3.1. Matching the previous section, for each dataset we attack the first 100 points, but this time using 1000 random points for each point. We attack each dataset once.

We store the L_{inf} adversarial distance found by the attack for each of the random points. For visualizing the spread of adversarial values we group the results by which victim is attacked and visualize the results in a boxplot.

For visualizing the difficulty of optimally attacking we compare the distances found by attacking each victim point with 1000 random starting points to the optimum results found by the MILP-attack described in section 2.3.2. For each starting point we compute whether it is within a small ϵ value (0.0001) of the optimum value. We then calculate the ratio of starting points that successfully manage to optimally attack the victim point and visualize this using bar charts.

We note that the models do not have perfect accuracy so some victim points are already mislabeled by the models, which is not supported by all attacks. Therefore, we exclude these points from the results here, meaning that the actual number of victims attacked per dataset can be slightly lower than 100. These points would not be interesting anyway, since the adversarial distance would trivially be 0 for them.

3.3.2. Individual point value spread

In this section we explore the effects of the starting point on the performance of the attack on a per-victim basis. We explore adversarial distances resulting of the attack in a series of boxplots shown in figure 3.2. We plot each dataset individually, since the values have no relation to each other. We plot the victim number on the X-axis, and the L_{inf} norm adversarial distances on the Y-axis.

We show the results on the breast cancer, MNIST and ijcnn datasets here, with the results of the other datasets found in the appendix A.1 due to the size of the graphs. We feel that these three datasets give a good overview of the results.

We see in figure 3.2a that for most victims in the breast cancer dataset all of the random points lead to the same adversarial distance, as the plot only has one median line for these victims. The remaining points we see that there are some where the median is better than the outliers, and some where the median is worse than the outliers.

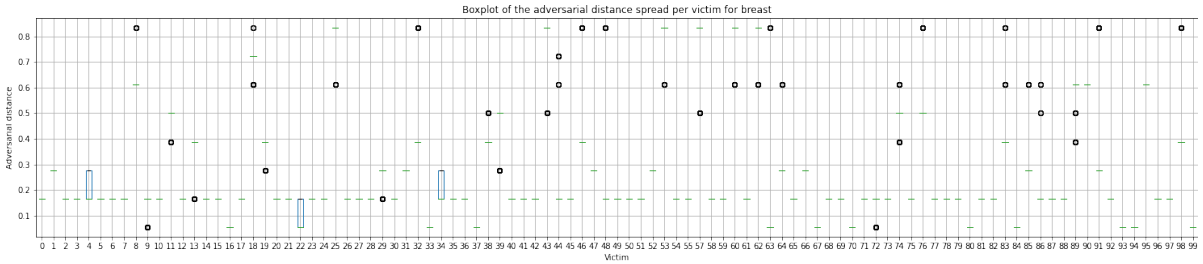
The ijcnn dataset in figure 3.2b is already substantially more varied, with a few points always successfully attacked like with the breast cancer dataset, but for most points there are larger spreads of outlier points around a relatively small median area. And for the mnist dataset in figure 3.2c we see even fewer points always successfully attacked, with most points having large spreads.

Discussion

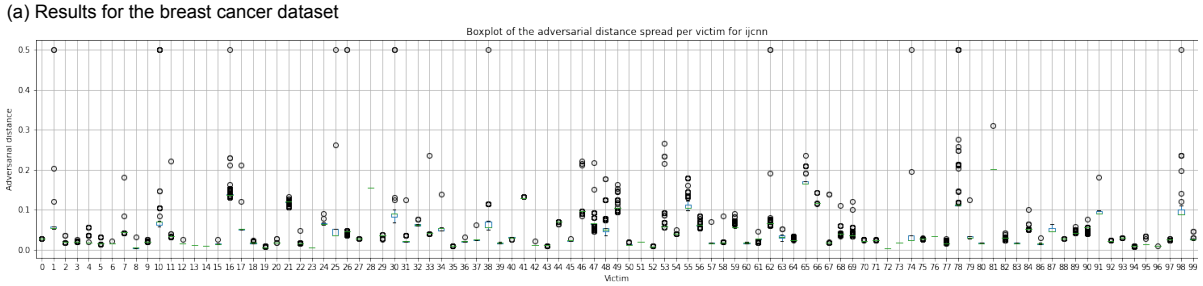
Breast cancer appeared to be the simplest dataset in the previous chapter because it already reached optimality occasionally with 50 starting points, and we see here that it is definitely the easiest dataset to attack. Most points are always optimally attacked, meaning only a small number of points lead to a difference in performance for the attack. The other datasets vary in how many points they have that are always successfully attacked, but we do see that all datasets have some points like this. If we were able to predict which points these are then we could skip attacking them multiple times, which could improve runtime significantly depending on the dataset.

For breast cancer we see very few different results for each point, which makes sense because it is the simplest model we are looking at as can be seen in table 3.1. For the other datasets we see that as the model increases in tree count/complexity, with the mnist-based datasets having the most most trees and also the most different results. This makes sense, since the more trees there are the more decision thresholds there are and thus the more possible results the attack has.

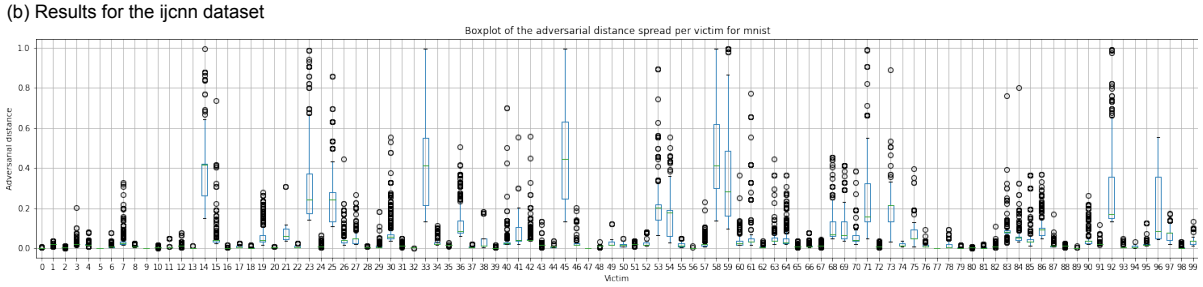
We see that the distance between the worst performing attack and the best performing attack also varies greatly from point to point and dataset to dataset. For mnist the difference for one victim can be



(a) Results for the breast cancer dataset



(b) Results for the ijcnn dataset



(c) Results for the mnist dataset

Figure 3.2: Boxplots showing the spread of L_{inf} adversarial distance scores for the first 100 victims of each dataset. Shown are the results for the mnist, ijcnn and breast cancer datasets.

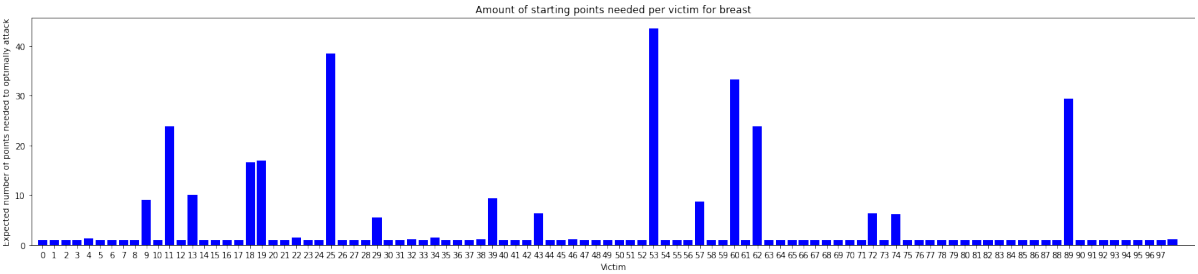
as large as 0.8, while for ijcnn the difference is only up to around 0.4, and for some of the other datasets it is even smaller.

Conclusion
In this section we looked at the difference in performance for each of the victim points we attack in a dataset. Even though the performance across the entire dataset improves significantly as the number of starting points increases, we find that all datasets have a subset of points that are always optimally attacked. The remaining points in the dataset vary in how large the diffence between well and badly performing starting points is.

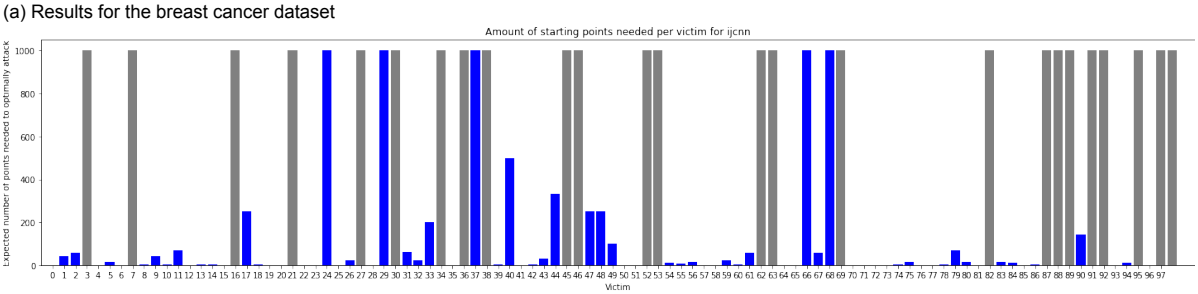
3.3.3. Individual point optimality

In this section we continue to explore the effects of the starting point on the performance of the attack on a per-victim basis. We explore how often the attack finds the optimal adversarial distance found by the MILP attack, getting an estimate for how many starting points a specific victim needs, on average, to be optimally attacked. We calculate this by diving the total number of starting points used (1000) by the number of points that led to the optimum. This gives us an expected number of starting points needed to find the optimum answer. For example, if we were to attack with 50 starting points we would expect to optimally attack all the victims with less than 50 in these graphs. We plot each dataset individually, since the y-axis values have no relation to each other. We plot the victim number on the X-axis, and the expected number of starting points needed on the Y-axis.

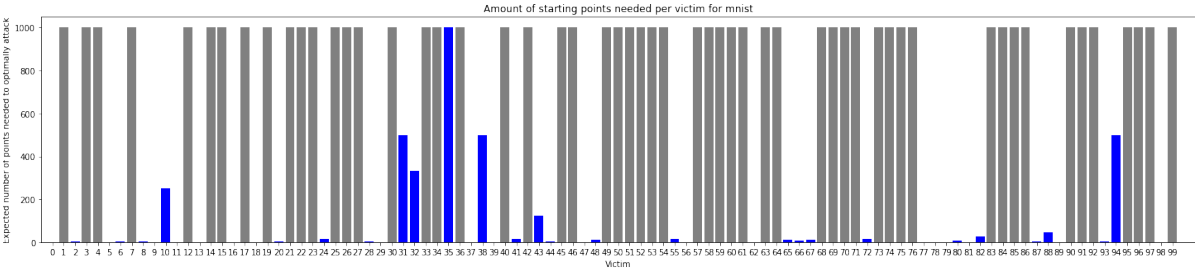
We will again only show the results on the breast cancer, MNIST and ijcnn datasets here, with the results of the other datasets found in the appendix A.2 due to the size of the graphs.



(a) Results for the breast cancer dataset



(b) Results for the ijcnn dataset



(c) Results for the mnist dataset

Figure 3.3: Plots showing the average number of random points needed to find an adversarial point as good as those found by the MILP attack. Shown are the results for the mnist, ijcnn and breast cancer datasets. Each of the first 100 test points in each dataset are attacked 1000 times, and from these results we calculate the average number of attacks needed to find the optimal answer once. We gray out the bars to indicate that no optimal scores were found by any of the 1000 starting points.

We plot our results in figure 3.3. Along the X-axis we see each individual victim and the Y-axis represents the average amount of random points needed to attack that specific victim. For cases where the victim is never successfully attacked we have colored the bar gray, to distinguish from cases where the victim was successfully attacked once. We clearly see the difference between the datasets here. The majority of the breast cancer dataset is easily attacked, always or almost always succeeding, with only around 16 points taking more than five average tries to optimally attack, and no points taking more than 50. MNIST, in contrast, has the majority of its points not optimally solved within 1000 attacks and are therefore colored gray.

Discussion
Here we see why the performance for breast cancer has been so much higher and why we actually hit the optimum performance back in section 3.2, as with only 16 points taking more than one attempt to get optimally it doesn't take many tries to get everything optimally attacked, hence we see this happen with 50 random attempts. ijcnn shows a relatively balanced division between points that are easy to attack and points that take more than 1000 attempts. For MNIST this graph does show that there are still quite a few points that are easy to attack, even in the most difficult dataset. However, we need to remember that even though most victims are never solved optimally with 1000 random starting points, this does not mean that all 1000 of those points are equally performant as we saw in figure 3.2.

For both ijcnn and MNIST we also see that there are points that take 50-200 points on average to solve. While this is definitely up to variance on exactly where they fall in this range, it does show that there is room for significant improvement in attack quality if we could increase the quality of our starting points.

If we could determine or predict the required amount of points for a given victim before running the attack, we could significantly improve the performance by focussing our time on the more difficult to attack victims instead.

Conclusion
In this section we looked at how many starting points it takes, on average, to optimally solve a given victim. We see that for breast cancer the number is surprisingly low, taking less than 50 starting points for the most difficult points in the test set. We see that the more difficult datasets start to mostly consist of points we never optimally attack even with 1000 random starting points, but that there is also a smaller subset of points for which a somewhat-larger amount of starting points than we have previously considered could give the optimal solution. In theory, it might be possible to balance our efforts between easy and difficult points better than using the same number of starting points for each victim.

3.4. Conclusions

In this chapter we answered looked at what the impact is of the starting points on the performance of the attack on both a dataset-wide level, and on an individual-victim level. Our results have shown that using more random starting points results in better attack performance, across all datasets and both L_{inf} and L_2 norms. We see that increasing the number of random starting points keeps improving the results of the attack, unless a plateau is reached as we reach optimal adversarial examples. We would expect this to continue after 100 as well, although the rate of improvement will likely flatten more and more. Therefore, we conclude that the impact of the starting points is significant.

On an individual-victim level, we see that except for breast-cancer all datasets have a mix of points that are easy to optimally attack, that require <10 random points, points that are reliably optimally attackable within 1000 random points as well as points that need more than 1000 random points to find an optimal adversarial example. Generally, we see that as the model complexity increases the ratio shifts towards more points that we never optimally solve. However, all models do still contain easy to attack points too. If there was a way to predict before running the attack whether a given point will be easy or difficult, we could focus our efforts on the more difficult points without sacrificing performance in the other points.

4

Alternate attack strategies

In this chapter we will be looking at how well other methods of generating starting points perform, when compared to the baseline random generation. First, we experiment with the distribution of the random generation in section 4.1. Next, we experiment with fixed directions of attack in section 4.2, and finally we experiment with reusing previously well performant points in section 4.3

4.1. Varying the mean and standard deviation

The default LT-attack uses random starting points generated around the target victim. A normal distribution with mean 0 and standard deviation 1 is used, and for each feature in the victim point a random number is drawn from the distribution and added (or subtracted) from the victim point. The models used are all scaled from 0 to 1, which we will look at in more detail in section 4.1.1.

In the rest of the section we will explore what happens when we vary this random distribution, seeing what happens when we reduce the standard deviation of the distribution in section 4.1.3 and next seeing what happens when we also move the mean of the distribution in section 4.1.4. Finally, in section 4.1.5 we look at whether the results obtained transfer to different test points of the same model.

4.1.1. Default attack range vs model range

First, however, we will go into more details on the models data range. All the models have been scaled to have their values between 0 and 1. The default attack uses a standard normal distribution, mean 0 and standard deviation 1, and adds this to the victim point to create the random point.

This means that the random value drawn from the distribution is already larger than 1 (or -1) approximately 31% of the time, causing the value to exceed the designed range of the model. And considering the victim point can be much closer to those bounds already, assuming the average victim feature has a value of 0.5 we would expect over 60% of points to fall outside of the 0-to-1 range the model expects the data to be scaled to. So why did we not see many random points performing the same in the previous chapter?

Before the generated value is actually used as a starting point, the point is first ran through a binary search between it and the victim point. This binary search tries to find a closer starting point for the actual attack, while remaining adversarial (as the starting point has to be adversarial). This causes the point to be closer to the victim point, and almost always causes the point to fall between 0 and 1. It doesn't guarantee them to be in the range of 0 to 1 that the model expects, but the model effectively clamps the generated random point to be between 0 and 1 afterwards. This means that the exact location, even outside of the 0 to 1 range of the model, impacts along which vector the binary search will work and therefore where the starting point will actually end up being.

This means that when changing the distribution we do not have to worry about points falling outside the 0 to 1 range, but it does mean that the larger the random distribution is the more likely we are to (effectively) have features values at the extremes. This will limit what values we experiment with in the next sections; we will focus on smaller ranges than the default, and not larger ranges.

4.1.2. Experimental setup

We set up the experiment as follows. For all following experiments we will attack the first 100 points in each dataset's test set, using 50 attacks per victim point. We take the average l_{inf} adversarial distance across all tested points as the resulting average adversarial distance. We repeat each attack 50 times to compensate for the randomness of the distribution/attack, showing the spread of values using box plots. This matches the setup of the previous chapter, allowing us to compare the results to those shown there.

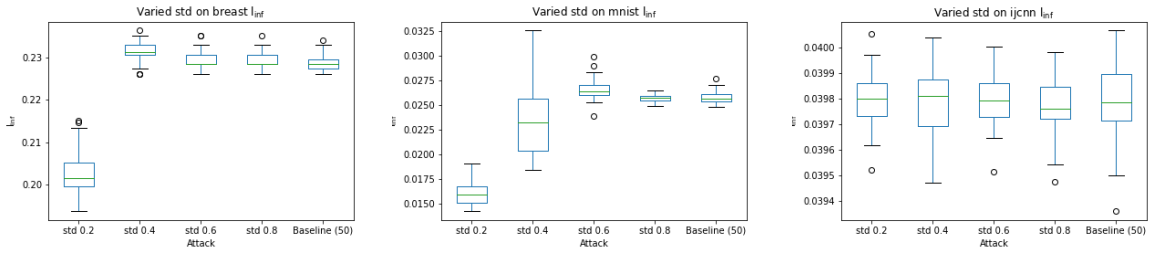
First, we will look at what happens when we reduce the standard deviation of the attack, giving us a narrower distribution. To achieve this we replace the default standard deviation of 1.0 with $0.2, 0.4, 0.6$ and 0.8. We compare to the baseline attack with standard deviation of 1.0. Each of these 5 configuration is ran 50 times. We choose 5 standard distributions spread out between 0 and 1 because we believe that if the standard deviation of the model does have a considerable effect on its performance, then having these 5 configurations should be enough to indicate it. The results of this experiment are shown and discussed in section 4.1.3.

Next, we look at what happens when we change the mean of the distribution instead. Because the model clamps value to between 0 and 1, we will use a much smaller standard deviation than the default 1.0 in order to focus on the effects of the mean of the distribution. For this reason, we will compare the baseline with mean 0 and standard deviation 1.0 to configurations with means $0.1, 0.3, 0.5, 0.7$ and 0.9 and standard deviation 0.1. We also switch to a bimodal distribution, meaning that when we use mean 0.1 we use a bimodal distribution around 0.1 and -0.1. Again, each of these 6 configurations is ran 50 times. The results of this experiment are shown and discussed in section 4.1.4.

We choose to limit ourselves to 7 datasets in this chapter, skipping bmnist and fmnist due to their long runtimes. In addition, we focus on the results of three datasets in this chapter, being breast cancer, MNIST and ijcnn, and show the results of the other datasets in the appendix. We choose these three datasets because breast cancer is the easiest and fastest of the 9 datasets we have used to attack. We choose MNIST because it is the dataset with the biggest performance gap between the LT attack and the MILP performance, only ahead of F-MNIST which takes substantially longer to run. We choose ijcnn because it is a middle of the pack dataset - it is a substantial jump up from breast cancer, but substantially smaller than the three MNIST datasets.

4.1.3. Varying the standard deviation

In this section we will look at changing the standard deviation of the distribution used, leaving the mean at 0. Figure 4.1 shows the resulting l_{inf} averages for each configuration type. The results for higgs, webspam, diabetes and covtype can be found in the appendix in figure B.1. When we compare the average best found adversarial distance we see that for the breast cancer and MNIST datasets using a standard deviation of 0.2 provides substantially better results than using larger standard deviations. While a small difference between configurations would be expected, the difference here is very large and only present for some of the configurations.



(a) Average adversarial distances for the breast cancer dataset (b) Average adversarial distances for the mnist dataset (c) Average adversarial distances for the ijcnn dataset

Figure 4.1: Average adversarial distances across all attacked points, per dataset. Each dataset is attacked with 4 different standard deviations and compared to the baseline with 50 random starting points. The X-axis indicates the attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

Investigating the points that lead to this large difference we find that 0.2 is not in fact performing better - it is performing much worse, and actually does not succeed in attacking all points in the dataset. In order to successfully attack a point, the attack needs to at some point randomly find an adversarial

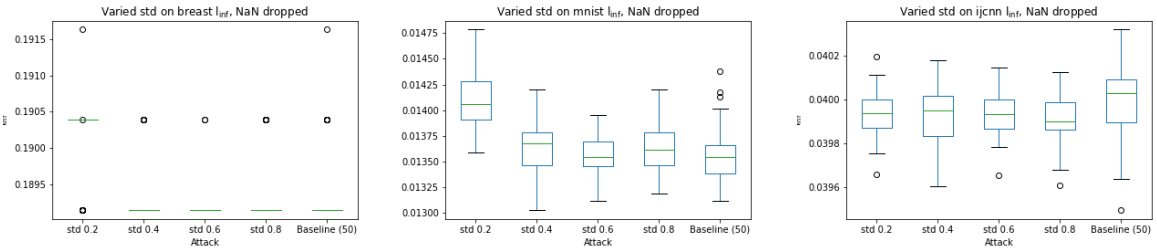
example to start from, and using the standard deviation of 0.2 causes it to fail to find a starting point for some of the points - these points might have the optimal adversarial example be significantly further away than 0.2, making it unlikely to stumble upon a starting point.

This means that not only is it failing to attack certain points, the points it is failing to attack tend to have a higher adversarial distance, leading to the very large disparity between the different attack configurations. For example, in the breast cancer dataset, only 10% of points fail to be attacked but this is enough to skew the average adversarial distance down by around 15%.

In order to get a better idea of the impact of points that not all configurations can attack, in the following section we will look to exclude points that are not successfully attacked from the results.

Only successfull points

In order to better compare the performance of the different configurations, we need to exclude points that are not attacked by all configurations. We still include points that were attacked by a subset of the configuration's attacks - for example, if 5 out of 50 attacks with standard deviation 0.2 successfully attack a point we include that point in the results. We do this to have as many points included as possible, even though the variance will be less accurately captured. Figure 4.2 plots the results that only include points that were successfully attacked at least once by each of the configurations.



(a) Average adversarial distances for the breast cancer dataset

(b) Average adversarial distances for the mnist dataset

(c) Average adversarial distances for the ijcnn dataset

Figure 4.2: Average adversarial distances per dataset for all successfully attacked points. Each dataset is attacked with 4 different standard deviations and compared to the baseline with 50 random starting points. Points are only included if each configuration succeeded in attacking it at least once. The X-axis indicates which attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

We can immediately see several differences from the previous set of plots. For the breast cancer dataset we see that there are very now few different outcomes left. Std 0.2 performs slightly worse than the other configurations, but the main result is that all other configurations perform the same. We removed 11 out of 100 points in the breast cancer dataset. and the remaining 88 points are so consistently attacked that the configurations larger than 0.2 show almost no difference at all. (the last point is one that is misclassified by the model, i.e., has adversarial distance 0. This point is excluded from the results.).

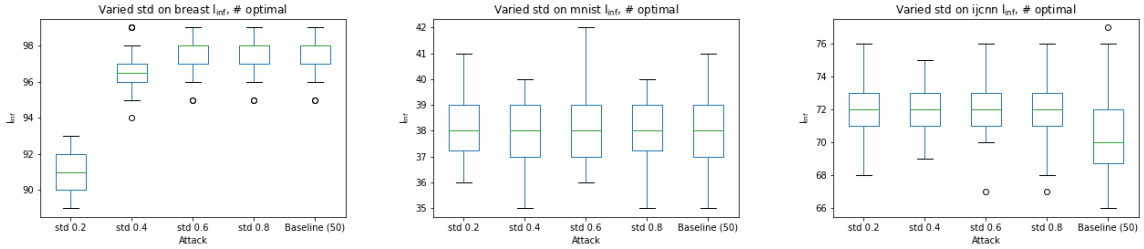
Similar results can be seen for the MNIST dataset, where 0.2 and 0.4 went from performing much better to performing much and somewhat worse, respectively. Here we had to remove 13 points from the dataset. For this dataset the remaining points in the dataset do offer some slight differences between the remaining configurations, but their performances are almost the same and we do not see a clear difference between configurations.

The results for the ijcnn dataset look identical to the previous ones. While there are a few points dropped, they appear to not make a significant difference in performance. There is a small difference between results - all the configurations seem to perform slightly better than the baseline, on average. The differences are small, however, and interestingly enough the best single run was using the baseline configuration.

However, these graphs have lost some information compared to the previous plots, as we are ignoring a subset of the points. In the next section we look at how many points are optimally attacked by each configuration.

Optimally attacked points

Instead of looking at the average adversarial distance, we only consider how many of the attacked points were optimally attacked. We use the MILP attack to find the optimal adversarial distance for each victim point, and classify a point as optimally attacked if it is within 0.001 of the found MILP optimal point to account for floating point calculation errors.



(a) Number of optimally attacked points for the breast cancer dataset

(b) Number of optimally attacked points for the mnist dataset

(c) Number of optimally attacked points for the ijcnn dataset

Figure 4.3: Number of optimally attacked points per dataset. Each dataset is attacked with 4 different standard deviations and compared to the baseline with 50 random starting points. Points are optimally attacked if the best found distance is equal (within a small ϵ distance) to the distance found by the MILP attack. The X-axis indicates which attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

For the breast cancer dataset, these graphs clearly show what we discovered in the previous section: the attack with standard deviation 0.2 is much worse at attacking a small subset of the points. It also explains why figure 4.2a has so few differences between the configurations - nearly all the points are optimally attacked by all configurations, so the differences are in only a select few points.

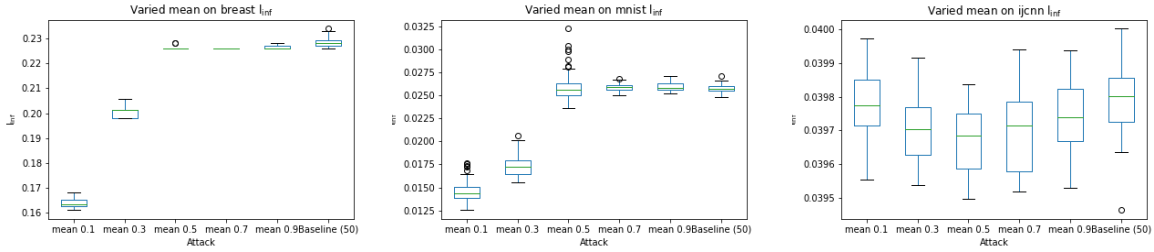
The MNIST dataset again shows few differences between the points, and the plot of the ijcnn dataset confirms that all the alternate configurations do slightly outperform the baseline on average, but again the single best run belongs to the baseline (and it is the same well-performant run).

Conclusion

The biggest takeaway from these results is that measuring the performance is difficult, as no single metric accurately captures the complete picture. We need to use a combination of the average adversarial distance, compensated for points that are unsuccessfully attacked, and the number of optimally attacked points to get a complete picture.

Changing the standard deviation can have big impacts on how likely the attack is to succeed at all, but for most datasets does not seem to offer substantial performance improvement.

4.1.4. Varying the mean



(a) Average adversarial distances for the breast cancer dataset

(b) Average adversarial distances for the mnist dataset

(c) Average adversarial distances for the ijcnn dataset

Figure 4.4: Average adversarial distances across all attacked points, per dataset. Each dataset is attacked with 5 different bimodal distributions and compared to the baseline with 50 random starting points. The X-axis indicate which attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

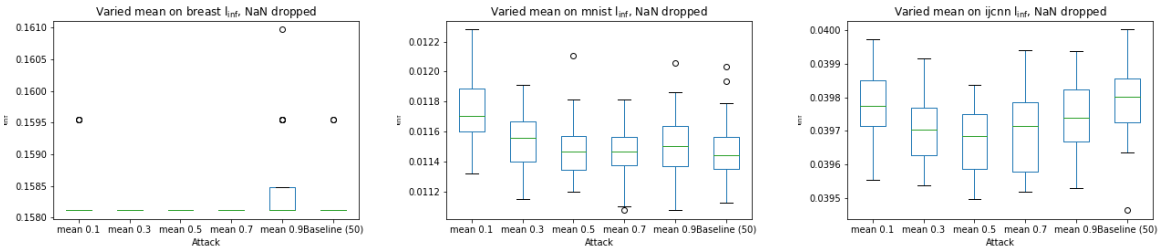
In the previous section we changed the standard deviation of the distribution, leaving the mean at 0. In this section we instead play with the mean, leaving the standard deviation at 0.1. We try means of 0 with std 1.0 as reference, and with std 0.1 we use means 0.1, 0.3, 0.5, 0.7, and 0.9. We look at

the results below using the same graphs as in the previous section. Figure 4.4 shows the resulting l_{inf} averages for each configuration type. The results for higgs, webspam, diabetes and covtype can be found in figure B.2.

When we compare the average best found adversarial distance as seen in figure 4.4 we see that for the breast cancer and mnist datasets the lower values, using a mean of 0.1 and 0.3, again performs much better than larger means, similarly to what we saw in the previous section. For the breast cancer dataset we also see that the baseline actually looks slightly less consistent than means 0.7 and 0.9. Additionally, we see that for the MNIST dataset mean 0.5 seems to have many more worse performing outliers, but also a better lower quartile; this is therefore also likely because of failed points. The results in the ijcnn dataset look more interesting, as we see a slight improvement in performance around mean 0.5, with an inverted bell shaped form in the graph.

For the MNIST and breast cancer datasets we expect to see the same as we did in the previous section, where once we exclude points that are never successfully attacked the configurations perform equal again. The shape of the graph of ijcnn suggests that this might not be related to failed attacks, but indicate a performance improvement. Similar graphs as for ijcnn are found for the diabetes and covtype datasets.

Only successful points



(a) Average adversarial distances for the breast cancer dataset

(b) Average adversarial distances for the mnist dataset

(c) Average adversarial distances for the ijcnn dataset

Figure 4.5: Average adversarial distances per dataset for all successfully attacked points. Each dataset is attacked with 5 different bimodal distributions and compared to the baseline with 50 random starting points. Points are only included if each configuration succeeded in attacking it at least once. The X-axis indicates which attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

Looking at figure 4.5 where we plot the same data but leave out points that were not successfully attacked, we do indeed see that for the breast cancer and mnist datasets the smaller means were due to failed attacks. For mnist the baseline does still look to perform best, although means 0.7 and 0.9 do occasionally outperform it, but at a cost of a larger variance.

However, the graph of ijcnn retains the same shape as before, suggesting that for this dataset using a mean of 0.5 with std 0.1 does outperform the baseline mean 0 with standard deviation 1.0.

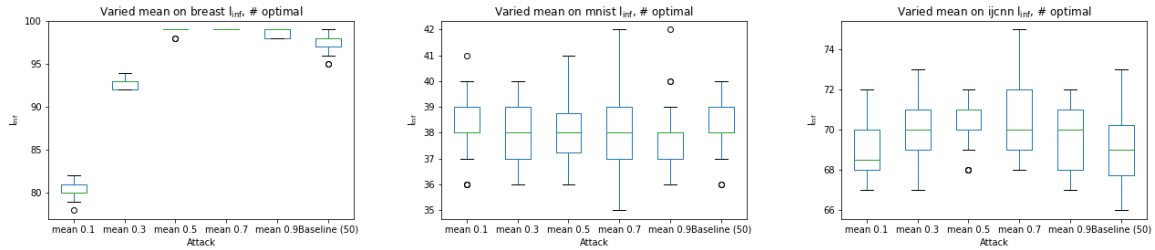
Optimally attacked points

In figure 4.6 we look at the number of optimally attacked points in order to confirm our conclusions above. In the breast cancer dataset we can confirm what we saw in figure 4.4a, that the higher mean are slightly more consistent than the baseline. In the mnist dataset we again see that 0.5 but especially 0.7 do sometimes perform better, getting more points optimally attacked, but with a higher variance as they also occasionally perform worse than the baseline. For ijcnn we again see a bell shape centered around 0.5-0.7 that performs better than the baseline. Around 0.5 seems most consistent, while 0.7 has the best upper quartile.

Conclusion

When comparing the performance between the baseline with mean 0 and standard deviation 1.0 and varying means between 0.1 and 0.9 with std 0.1 we see very large differences between the datasets. Generally, means below 0.5 risk failing at attacking all points. For the larger mean values it varies from dataset to dataset what the effects are.

For the breast cancer dataset the high means are slighlty more consistent than the baseline. For



(a) Number of optimally attacked points for the breast cancer dataset

(b) Number of optimally attacked points for the mnist dataset

(c) Number of optimally attacked points for the ijcnn dataset

Figure 4.6: Number of optimally attacked points per dataset. Each dataset is attacked with 5 different bimodal distributions and compared to the baseline with 50 random starting points. Points are optimally attacked if the best found distance is equal (within a small ϵ distance) to the distance found by the MILP attack. The X-axis indicates which attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

mnist the performance is not significantly impacted. For the ijcnn we see an improvement in performance, centered somewhere around 0.5-0.7.

A big drawback of this method is that we attack all points with the same distribution, and only look at results across the entire dataset. It is possible that a certain distribution does better for some points but worse for others.

4.1.5. Performance transferability

We saw in the previous section that sometimes specific ranges can work slightly better for a target model/dataset. In that section we compared the performance of different ranges on the same set of test points. We saw that for the ijcnn dataset, a mean of around 0.5-0.7 outperformed the baseline as seen in figure 4.4c. In this section we do an experiment to test whether the performance difference transfers to a different set of testpoints.

We will run the same configuration as in the previous section, however, we will now attack points 101-200 of the test set rather than points 1-100. If we again see that the means around 0.5-0.7 perform better than the baseline, then we have a strong indication that this performance increase might transfer to other parts of the testset and could be used to improve the attack. We plot the results in figure 4.7.

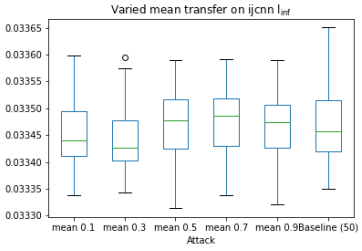


Figure 4.7: Boxplot showing the average adversarial distances for the ijcnn dataset on points 100-200, using the same configurations as in figure 4.4. The X-axis indicates which attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

We see that these results are quite different than the results in figure 4.4c. In general, you might notice that the l_{inf} values are much lower. This is because we are attacking a different set of test points which have different corresponding optimal values, so this is to be expected. Instead, we focus on comparing the shape of the graph. We see that we do not have the clear bell/cup shape we had previously. The attacks with mean 0.5-0.7 now no longer clearly outperform the baseline. Their average performance is worse. We actually see that the lower means score better than the baseline.

It might be possible to improve results by determining the configuration used on a per point level, if it is possible to predict beforehand which points will perform well with which configurations.

Conclusion
We compared the results on the ijcnn dataset on the first 100 test points with the performance on test points $101-200$, to see if the same configurations performed well in both cases. Unfortunately, it does not seem that the performance difference between configuration cleanly transfers to other parts of the test set, as the configurations went from better than the baseline to worse than the baseline.

4.2. Fixed attack direction

In this section we will explore what happens to the performance of the attack when we attack with starting points in a fixed direction from the victim, rather than randomly generated around the victim.

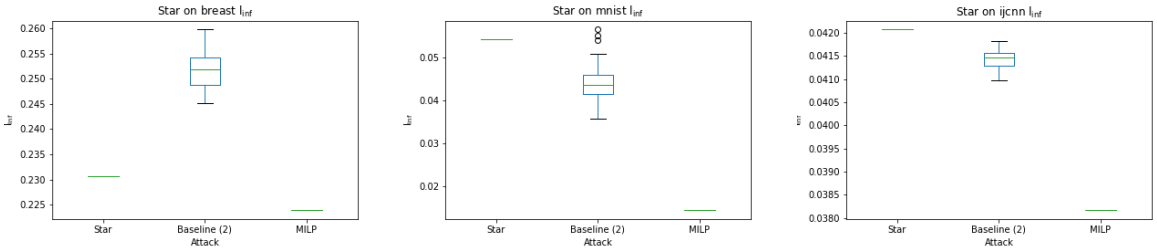
4.2.1. Experimental setup

We set up the experiment as follows. We attack each victim point twice, once from the positive direction and once from the negative direction. In order to create the positive direction starting point, we try to add $+1$ to each feature of the victim point. However, we then find that this will often fail to create a valid starting point as it has to be adversarial. In order to increase the chance we can attack a point we do the following. For each feature, we first try to add $+1$. If that fails, we instead try $+0.8$, and then $+0.6$, $+0.4$, and $+0.2$. We find that this is enough to always find an adversarial point, while still attacking from a fixed direction. For the negative direction starting point, we flip the sign on what we are adding to the victim

We will attack the first 100 points in each dataset's test set, and show the results in box plots comparing the performance to that of the baseline with 2 points and the optimal distance of the MILP attack. We will again focus on breast cancer, MNIST and ijcnn and show the other 6 datasets in the appendix. We will refer to this as the 'star' attack in the following sections for brevity.

4.2.2. Results

We look at the results below in figure 4.8 using the same graph setups as in the previous section. The other datasets can be found in figure C.1 We compare the performance of the star attack (with 2 starting points) to that of the baseline with 2 randomly selected starting points. We also include the optimum result obtained by the MILP attack as reference point, as the difference in performance shows the total improvement possible and so it can be treated as a 0-point.



(a) Average adversarial distances for the breast cancer dataset (b) Average adversarial distances for the mnist dataset (c) Average adversarial distances for the ijcnn dataset

Figure 4.8: Average adversarial distances across all 100 attacked points, per dataset. The X-axis shows the attack and the Y axis shows the adversarial distance in the L_{inf} norm. The performance of the star attack is compared to the baseline with 2 random points and the optimal answer of the MILP attack.

Since we only run the star attack once because it has (very nearly) no randomness, the results of the star attack show up as just a line in the bar chart, just like the results of the MILP attack.

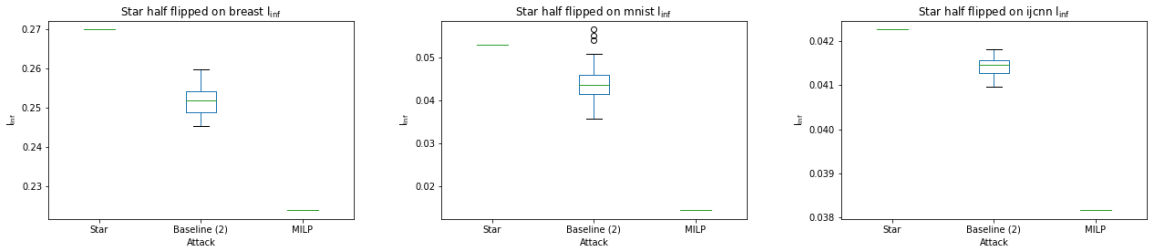
4.2.3. Discussion

We see that for the MNIST dataset the performance is similar to the worst performing runs of the baseline that used 2 random starting points. For ijcnn the performance is significantly worse than the baseline. We expected the performance to be similar to that of the baseline with 2 starting points. For breast cancer the results are much better than the baseline, which initially seems similar to what happened in the last section, where an attack performs better because it fails to attack all points. However, the star attack actually successfully attacks all points so this is not the case. Breast cancer

is a much more simple dataset to attack than the other two, but it is surprising to see the performance be so significantly different.

A possible explanation would be that this attack, generally, ends up with starting points much further away since we always (try to) have all features significantly changed, whereas for the random starting points there will usually be features that are quite close to the original value. This in turn means that with the star attack almost all features are on different sides of decision boundaries, while with the random starting points more features will fall in the same decision boundaries. This would then explain why the attack performs much worse, as it needs to find its way across many more terminal nodes in order to get to better results.

In order to examine whether this is just a coincidence or consistent for the breast cancer dataset we attack it again, but instead of +1 and -1 in all dimensions we flip the sign of the attack on the first three features compared to the previous run. We plot the results in figure 4.9a. We now see that the performance for breast cancer is much worse, now matching th performances of the other datasets. For comparison we also show the changed performances of mnist and ijcnn, which have not significantly changed, still performing worse than the baseline with 2 random starting points.



(a) Average adversarial distances for the breast cancer dataset

(b) Average adversarial distances for the mnist dataset

(c) Average adversarial distances for the ijcnn dataset

Figure 4.9: Average adversarial distances across all attacked points, per dataset. X labels indicate which attack type and the Y axis shows the adversarial distance in the L_{inf} norm. The performance of the half flipped star attack is compared to the baseline with 2 random points and the optimal answer of the MILP attack.

This is an interesting result, as it suggests that (at least for the breast cancer dataset) some configurations of the star attack perform much better than others. That +1 and -1 performs so well appears to just be a coincidence, but it would be interesting to see if there is any such configurations present for the other datasets.

In general, we believe it is likely that any given starting direction will only be perfectly suited for some of the test set, and a different starting direction would be more suited for the remainder of the test set. We investigate this idea further in the next section, where we attack all points from all directions.

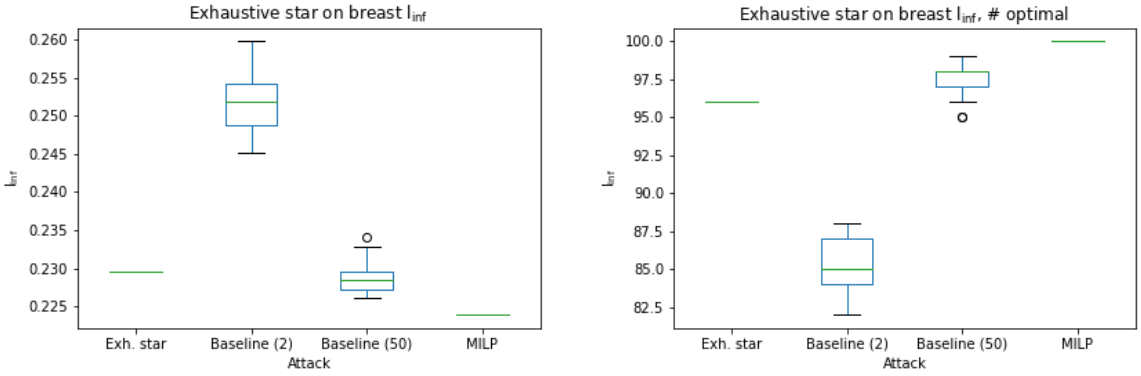
4.2.4. Exhaustive star

In the previous section we chose to attack from both +1 and -1, and saw that for the breast cancer dataset some directions actually perform very well. In this section we will investigate what happens when we attack with all permutations of starting directions on all test points, in order to check whether a combination of different directions for different points could lead to better performance. Since this is exponential in the number of features the dataset has, we unfortunately find ourselves computationally limited to breast cancer (10 features) and diabetes (8 features). Runtime for ijcnn (22 features) would already be several hours hour per attacked point, which we deem too long to experiment with.

We show results for the breast cancer dataset in figure 4.10 and for the diabetes dataset in figure 4.11, using similar graphs as in the previous section. We compare the results to the baseline with 2 random points just as we did with the star attack, as well as the largest baseline we ran at 50 points. For both datasets the attack successfully attacks all target points, so we omit the graphs relating to that here.

Unsurprisingly, we see that for both datasets we perform much better than the regular star attack, however, for the breast cancer dataset we actually perform worse than the baseline with 50 starting points, even though we effectively use 1024 starting points, and diabetes roughly matches the performance of the baseline with 50 starting points while using 256 starting points.

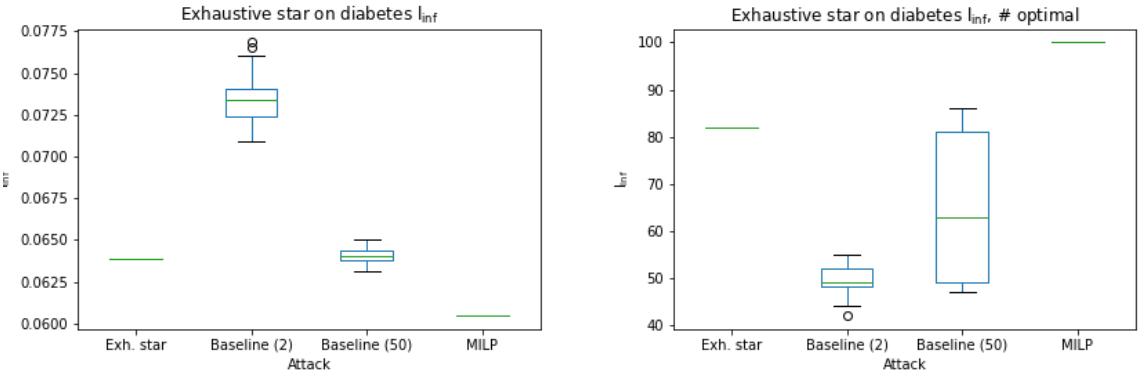
This is quite surprising, as with randomly generated starting points we saw that performance keeps



(a) Average adversarial distances. The X-axis shows the attack and the Y axis shows the adversarial distance in the L_{inf} norm.

(b) Number of optimally attacked points. The X-axis shows the attack and the Y axis shows the number of optimally attacked points.

Figure 4.10: Boxplot showing the results of the exhaustive star attack, compared to the baseline with 2 and 50 random points and the MILP attack, for the breast cancer dataset. The X-axis shows the attack and the Y axis shows the adversarial distance in the L_{inf} norm.



(a) Average adversarial distances. The X-axis shows the attack and the Y axis shows the adversarial distance in the L_{inf} norm.

(b) Number of optimally attacked points. The X-axis shows the attack and the Y axis shows the number of optimally attacked points.

Figure 4.11: Boxplot showing the results of the exhaustive star attack, compared to the baseline with 2 and 50 random points and the MILP attack, for the diabetes dataset.

increasing as we increase the number of starting points and this does not appear to be the case here. We also see that we do not optimally attack as many points as the baseline did, meaning that there are some points in both datasets for which the optimum value is not found with this experiment, but is reached with random points. This could be because this attack always changes all features, while the random starting points might leave some features unchanged.

4.2.5. Conclusions

We attacked using a fixed direction relative to the victim point and found the results significantly worse than that of the baseline for MNIST and ijcnn. For breast cancer performance is much better than that of the baseline with an equal number of points, performing closer to that of the baseline with 20 points. However, this appears to be a coincidence with the chosen direction as performance becomes similar to the other datasets with a different starting direction. In addition, we find that for the breast cancer and diabetes datasets attacking from all directions performs worse and 5-20 times slower than attacking with 50 random starting points.

4.3. Reusing previously well performing points

In the previous sections we explored whether we could improve on the random generation parameters to improve performance. In this section we will look at whether we can do better than random generation by instead re-using previously well-performing starting points when attacking new points. The idea behind this is that with completely random points there is some chance that the randomly generated point performs very poorly - perhaps it is very far away, perhaps it is stuck in a local minimum, or perhaps the path between the random point and the theoretical optimum point is long and convoluted or blocked by non-adversarial decision space. By using nearby random points that previously performed well, we might be able to filter out these worse points meaning that we perform better than with an equal number of random points.

While we can use random points that performed well for a different point, it offers no guarantee that they will also work well for a new point. Every point we attack has different coordinates and most likely will have a different optimal attack point, so there is no guarantee that this will perform better than the baseline. What we gain by filtering out bad points might be compensated for by a worse spread in random attack points, possibly making us perform worse. Overall, we expect it to perform slightly better than an equivalent number of randomly selected starting points. While there is a chance we will not find an easy to attack point because we are attacking from limited direction, we also expect to sometimes find those hard to find places near difficult victims.

Because we expect this attack become better the more points we attack we will also look at the performance of an extended attack, which attacks the same 100 points after attacking 100 different points. This is because as we attack more points, we get a larger selection of points to pick from that should on average get closer to the victim point. This means that we would expect the extended attack to perform better than the non-extended attack.

4.3.1. Experimental setup

We set up the experiment as follows. As in previous sections, we attack the first 100 points in each dataset's test set so that we can compare scores easily. For the larger attack, we first attack points 100-200 before attacking the first 100 points. This is to ensure that we can compare the results easily, since we can't directly compare results from points 100-200 with those of the first 100, while still giving the attack a larger number possible points to start from.

We group the victim points by their actual class label. The first victim point of each class gets attacked normally, using random points. We choose 20 random points because it provides a larger selection of starting options, where 5-10 random points were causing big variance between attack attempts for the first following points. With 20 the variance is substantially lower between runs of this experiment.

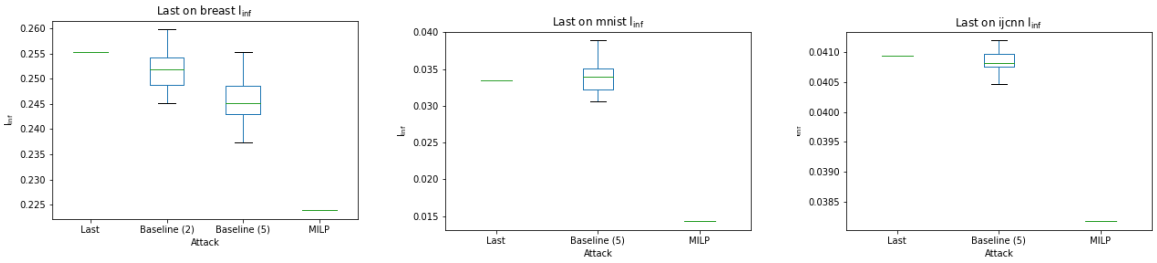
Afterwards, every point selects the 5 closest starting points from all previous victims from the same class, only taking into account the point that resulted in the lowest adversarial distance for that specific victim. We drop duplicate points in case there are multiple points with the same best-found optimum, so that we always have 5 distinct starting points. We found that using very few previous points, only 1 or 2, significantly increases the variance in performance of this attack, so we opt to attack with 5 points

and compare the results to 5 random starting points.

4.3.2. Results

We compare the performance of the 'last' attack with 5 previously used well-performing points to the performance of the baseline LT-attack with 5 random starting points. We also show the optimal result produced by the MILP attack, the distance to which can be used to more accurately describe how well an attack is performing. When the performance of the attack is substantially different from that of the baseline with 5 points we will also include other relevant baseline performances, to give a better perspective on performance.

Following the previous sections, we will compare on the breast cancer, mnist and ijcnn datasets. We show the results of the 'last' attack which attacked only the first 100 points, and so had no starting knowledge of previously well-performing points, in figure 4.12.



(a) Breast cancer dataset. We include the baseline with 2 points.

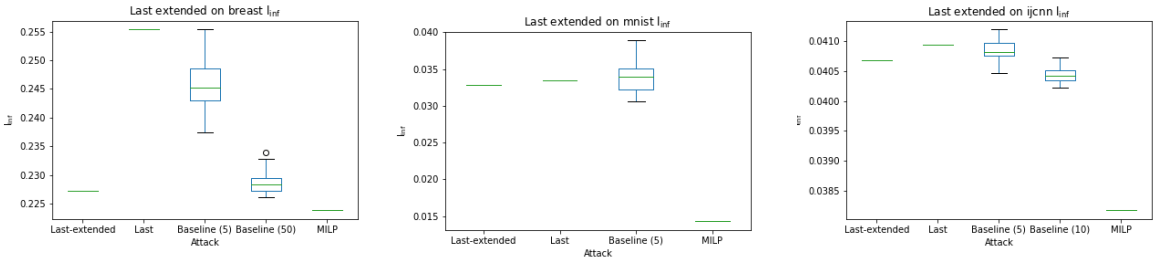
(b) MNIST dataset

(c) ijcnn dataset

Figure 4.12: Comparison of the last attack with 5 reused points to the baseline attack with 5 points and the MILP attack. The X-axis indicates the attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

We see in figure 4.12a that for the breast cancer dataset that the performance is much worse than the baseline with 5 points, and would even be in the worst 25% for the baseline attack with 2 points. In the MNIST and ijcnn datasets however, the performance is comparable to that of the baseline with 5 points.

Next, we will look at what happens to the results when we extend the last attack to first attack 100 different points, before attacking the same 100 points for comparison. The results are shown in figure 4.13. We clearly see a huge shift in performance for the breast cancer dataset from the last attack performing much worse than the baseline with 5 random points, to now the extended last attack performing approximately as well as the baseline with 50 random points. Meanwhile, for the mnist dataset the performance is almost unchanged, and for the ijcnn dataset the performance is slightly improved, from slightly worse on average than the baseline to slightly better.



(a) Breast cancer dataset. We include the baseline with 50 points as comparison.

(b) MNIST dataset

(c) ijcnn dataset. We include the baseline with 10 points as comparison.

Figure 4.13: Comparison of the extended last attack with 5 points to the baseline attack with 5 points and the MILP attack.The X-axis indicates the attack type and the Y-axis shows the adversarial distance in the L_{inf} norm.

4.3.3. Discussion

The performance of the last attack on MNIST and ijcnn is about what we expected, with ijcnn being slightly worse than anticipated as we expected both to outperform or match the baseline. The perfor-

mance on the breast cancer dataset, however, is surprisingly bad. We have seen before that the breast cancer dataset can be more volatile than the other datasets. A majority of the victims in the dataset can be optimally attacked from (nearly) any starting point, while a much smaller portion of points are more difficult to attack, as seen in 3.3. This means that it is possible the last attack is making use mostly of points from easy to attack points, from which it might not be likely to solve the difficult points.

When we investigate, we do indeed see that for the 16 points (between 0 and 100 in the test set) in the breast cancer dataset that takes more than 2 tries on average to find the optimal answer, 15 of those 16 points are not optimally solved by the last attack on the breast cancer dataset. All other points are indeed solved optimally, as we would expect. When compared to the baseline, between 0 and 4 of the 16 difficult points are solved worse than the baseline and either 0 or 1 point is solved better than the baseline. In the extended version of the attack, we see a massive improvement in the performance on the breast cancer dataset, and investigating the results show that now only one point is not optimally solved.

The extended last attack is unfortunately not a significant improvement on the mnist dataset. Investigating the points shows us that while around 20 points do see an improvement in the extended attack compared to the normal attack, almost the same amount of points also get worse in the extended attack. On the ijcnn dataset we see a similar effect - about 10 different points are both improved and worsened. It would be interesting to find out whether the points that worsen in performance do so because the starting points that led to better results were not found in the extended attack, or because the extended attack had different, closer starting points that ended up performing worse.

4.3.4. Conclusions

In this section we have created and evaluated the performance of the last attack in comparison to the baseline LT attack. We find that on attacking 100 points the last attack is generally outperformed by the baseline, or at best matches an average baseline attack. We find that the extended last attack, which simulates a larger attack by first attacking 100 different points and using the results gained during that attack on the original 100 points, does outperform the baseline. The degree to which it performs better varies from dataset to dataset, and does not uniformly improve all attacked points.

4.4. Conclusions

In this chapter we have investigated what the effect is of different random generation methods on the performance of random starting points. We found that generally using a different setup for the random generation does not result in significantly different performance. We see that for some datasets some configurations perform better, but that this does not necessarily hold for the entire test set.

In order we can improve on the performance of random starting points by using a different method to create starting points we first investigated using fixed starting directions relative to the victim point. We found that while it can sometimes give better performance than the baseline, this is likely only possible for simpler datasets and it is computationally expensive to find the effective directions. In addition, we found that even exhaustively trying all directions gives performance worse than using 50 random starting points, while using $5 - 20$ times the amount of time.

We also created an attack where we reuse previously well-performing points when attacking new victims. We found that on smaller test sets, where the attack does not have a large selection of starting points to choose from, it gets outperformed by the baseline. However, we find that when we increase the number of points attacked we outperform the baseline on some datasets, and match it on others.

5 Discussion

In this work we investigated the effects that (randomly generated) starting points have on the performance of adversarial attacks on decision trees. In this chapter we discuss the limitations we ran into in section 5.1, then the ethical concerns around improving attacks in section 5.2 and finally give recommendations for future work in section 5.3

5.1. Limitations

We found that actually evaluating the performance of attacks is difficult. We use a combination of different metrics to get an overview of performance at the dataset level, but in order to really understand the underlying behaviour we need to look at the performance at a point level, for which we have not found an effective method for larger datasets. This means that when answering the question of why we see a certain behavior, we can answer based on the general statistics and earlier behavior of the dataset in question, but do not form a better understanding of how this relates to other points and especially to what degree this generalizes to other datasets. While we have seen that between datasets there are large difference in performance, where some datasets do see markedly improved performance using certain methods while other datasets show no change or even worse performance, we have not found a clear explanation based on the underlying data/model as to why this is the case.

5.2. Ethics

As we are working on improving the effectiveness of attacks on machine learning models, we need to keep in the back of our mind the impact that these systems could have. We see that many defensive methods rely on adversarial training or at least adversarial examples at some point in their model creation pipeline to increase robustness, and it is with this in mind that we investigated this topic to begin with. Any improvements in speed or performance, while possibly used maliciously, can also be used to improve defenses. In general, we would argue that it is better to deal with the devil you know than the devil you don't - even though any development could be used by malicious parties, it is also important for everyone to be aware of what is possible.

5.3. Future work

The main follow up would be investigating to see whether reusing previously obtained knowledge can be used to more effectively attack points in an adversarial training setting. In our investigation we focused on the attack of new victims using previous information, while in adversarial training the same points might be attacked in each training epoch. Using the same starting points that previously worked well might be very effective in such a setting, as we can use the starting points that performed best in the previous training epoch.

It would be interesting to see what happens when we attack robustly trained models, rather than the non-robustly trained models we used. While robust models are designed to make attacks harder, they are generally not designed to protect against the idea that information can be learnt and reused for

new victim points so it would be interesting whether that make the performance we obtained (relatively) better than the baseline performance than it is now.

In addition, we think it would be interesting to investigate whether any of our findings, either in generating starting points or in using previously learnt information in attacking new points transfers between models trained on the same dataset. Since adversarial examples have been shown to transfer to some degree, it would be interesting to see whether these strategies also transfer between models or whether they are model specific.

6

Conclusion

This thesis aimed to investigate whether it is possible to improve the performance of adversarial attacks on decision tree ensembles by changing the generation method used for the starting points.

We investigated the impact of the starting points of attacks on their performance, and found that there is a significant impact on the performance. More randomly generated starting points greatly outperforms fewer randomly generated starting points, from which we conclude that there is a difference in how well starting points perform. When looking at the performance of individual victims, we see that there is a big difference between easy and hard to optimally attack victims. The dataset/model does have a big impact on the ratio of easy to difficult points, but all datasets contain both easier and harder points to attack.

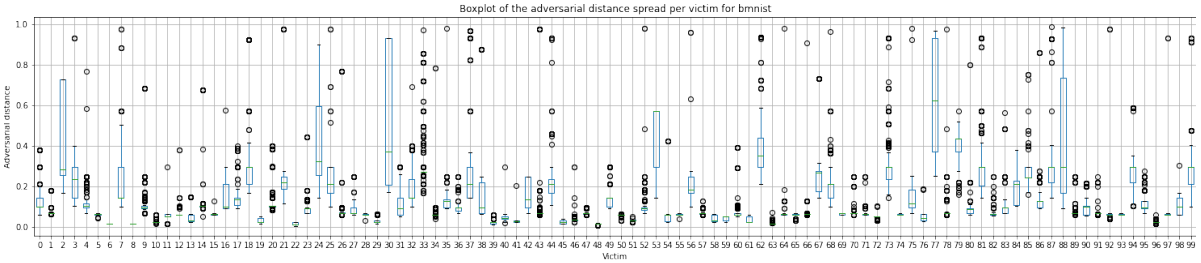
Next, we investigated what the effect is of different random generation methods on the performance of random starting points. We found that while for some datasets some specific random distributions do perform better than others, the difference is very small and does not generalize to the entire test set. As long as the distribution does not fail to create valid starting points, it does not have a real impact on performance. Analyzing the results is difficult. There is already high variance when looking at results across an entire dataset, but to investigate why certain behaviors show requires investigating individual points which have much higher variance still. We found a combination of metrics to look at to get a decent comparison of performance, but it still requires careful checking of the data to avoid drawing the wrong conclusions.

Finally, we investigate whether we can improve on randomly generated starting points by using a different method of generating starting points. We first investigated using fixed starting directions relative to the victim point, which we found performs significantly worse than randomly generated points except for a specific direction on the simplest dataset. Trying all possible directions does do a very successful attack, but is much slower and/or worse than simply using more random starting points instead. Next, we created an attack where we reuse previously well performing points when attacking new victims. We found that on small test sets this is worse than the baseline, but as we expand the testset, which allows the attack to use more information on previous points, we outperform the baseline on some datasets, and match it on others.

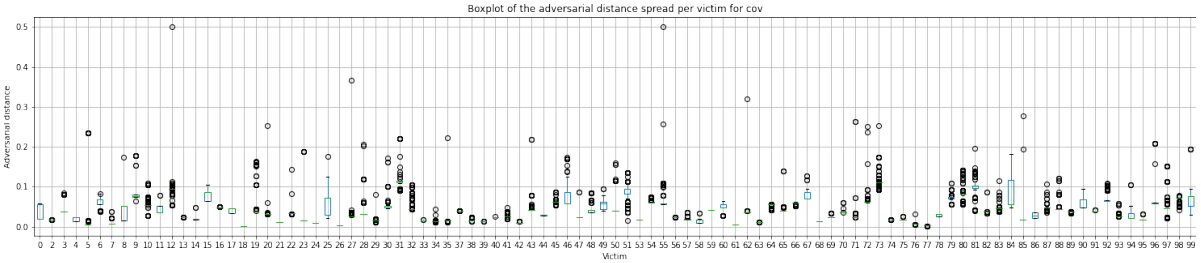
We conclude that for the purpose of improving adversarial attacks randomly generated starting points are very effective, and the exact distribution used to generate the points does not matter significantly. We do believe there is potential in improving the attacks, possibly through the starting points, by leveraging information gained by attacked points for attacking new victims.

A

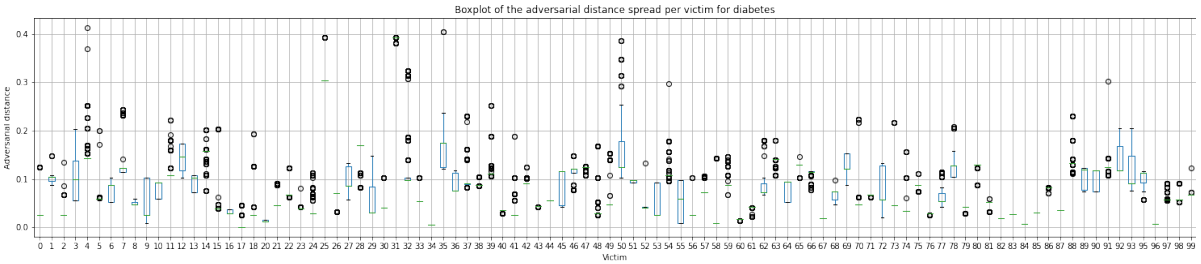
Individual point performance, extra plots



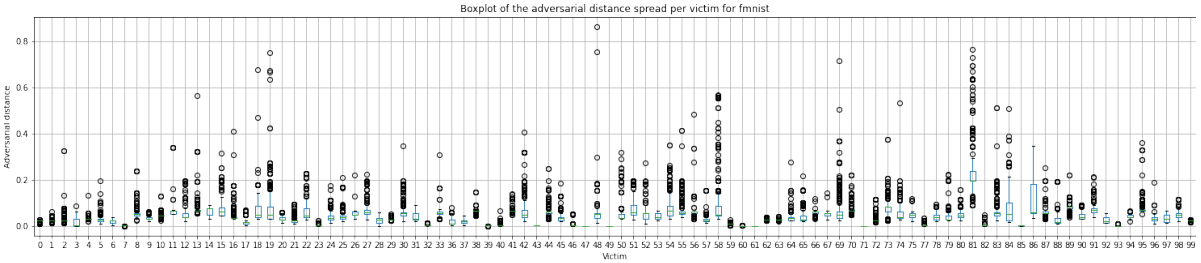
(a) Results for the bmnist dataset



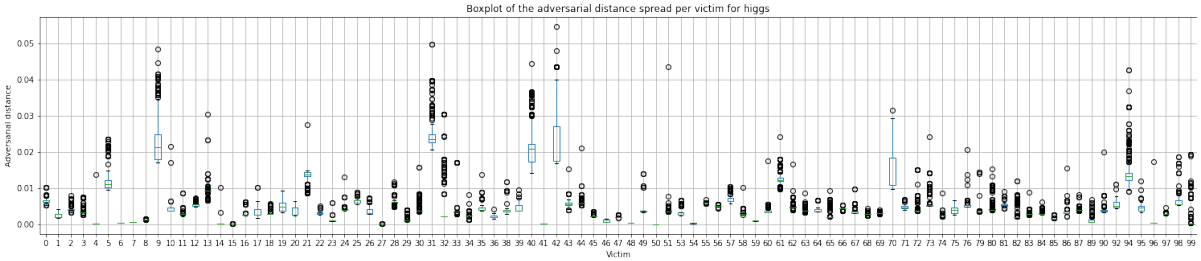
(b) Results for the covtype dataset



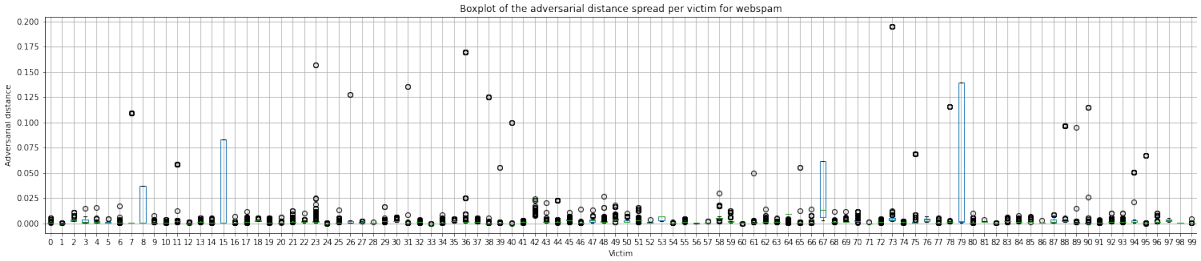
(c) Results for the diabetes dataset



(d) Results for the fmnist dataset

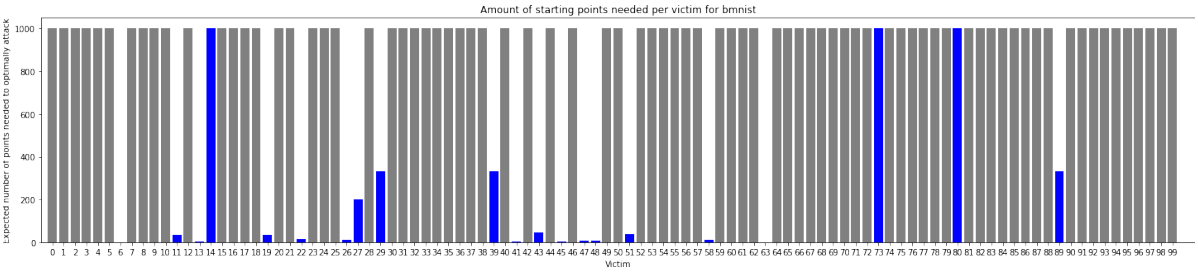


(e) Results for the higgs dataset

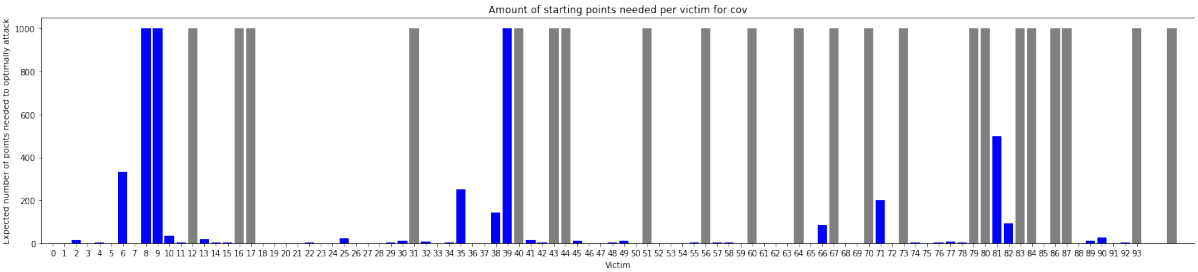


(f) Results for the webspam dataset

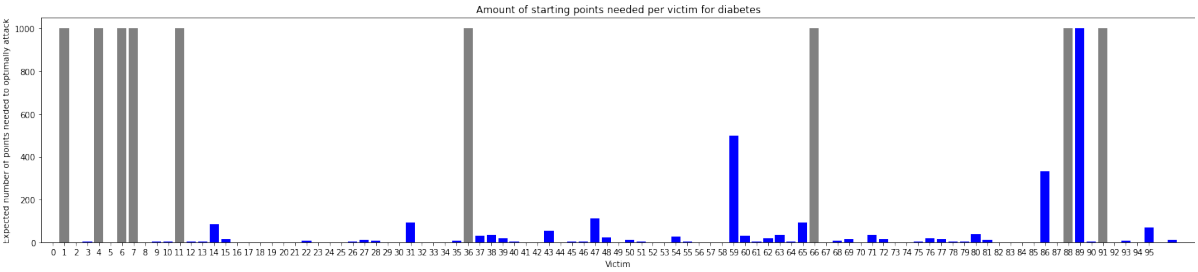
Figure A.1: Boxplots showing the spread of L_{inf} adversarial distance scores for the first 100 victims of each dataset.



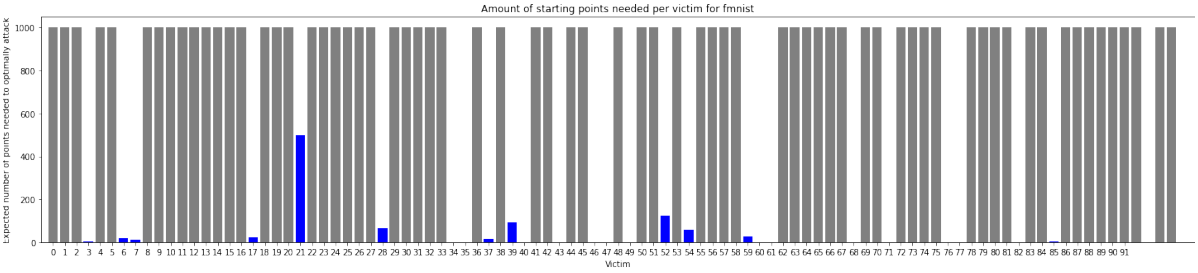
(a) Results for the bmnist dataset



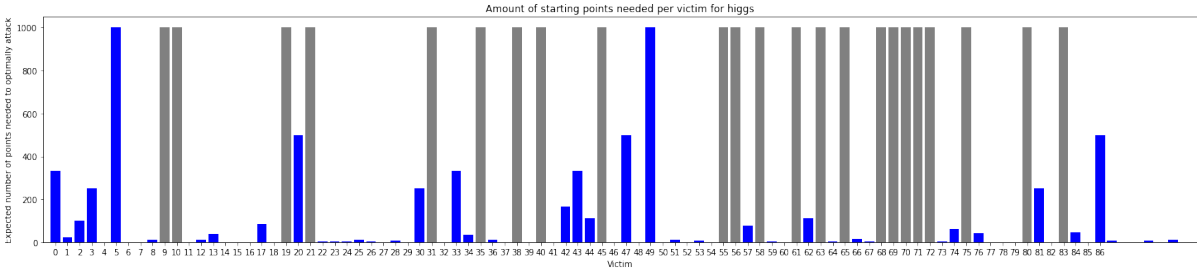
(b) Results for the covtype dataset



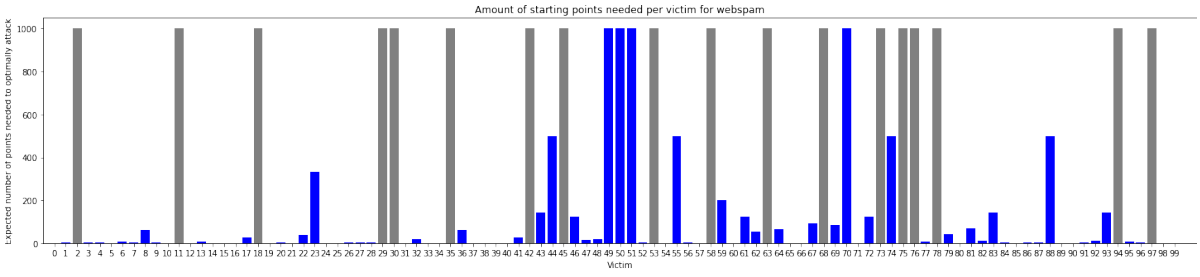
(c) Results for the diabetes dataset



(d) Results for the fmnist dataset



(e) Results for the higgs dataset

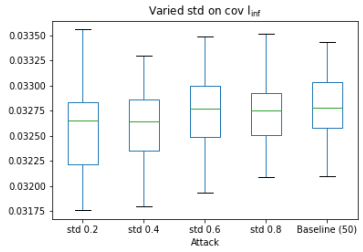


(f) Results for the webspam dataset

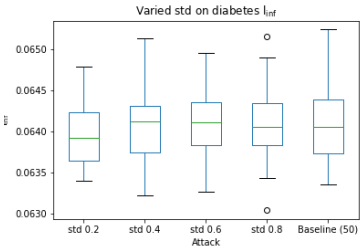
Figure A.2: Plots showing the average number of random points needed to find an adversarial point as good as those found by the MILP attack. Each of the first 100 test points in each dataset are attacked 1000 times, and from these results we calculate the average number of attacks needed to find the optimal answer once. We gray out the bars to indicate that no optimal scores were found by any of the 1000 starting points.

B

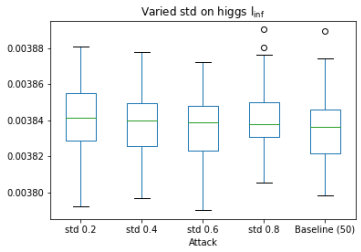
Varying mean and standard deviation, extra plots



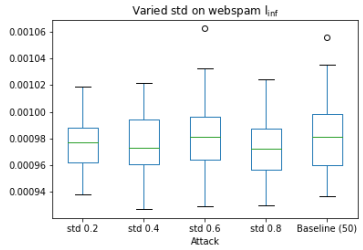
(a) covtype



(b) diabetes

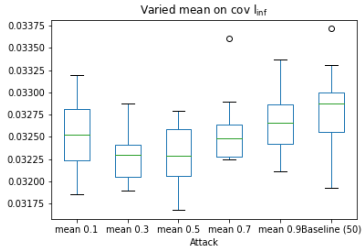


(c) higgs

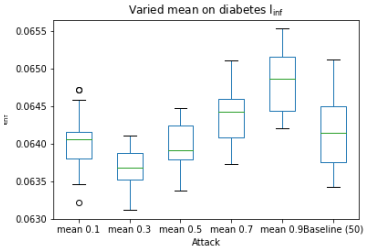


(d) webspam

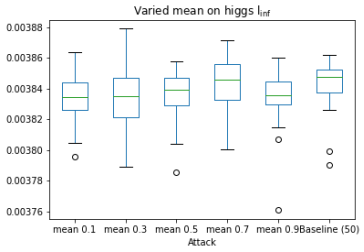
Figure B.1: Plots showing the average number of random points needed to find an adversarial point as good as those found by the MILP attack. Negative scores indicate that none of the random points scored as well as the MILP attack.



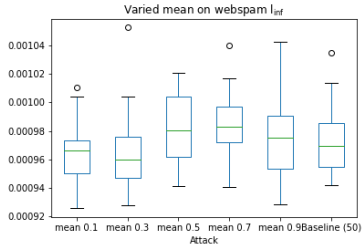
(a) covtype



(b) diabetes



(c) higgs

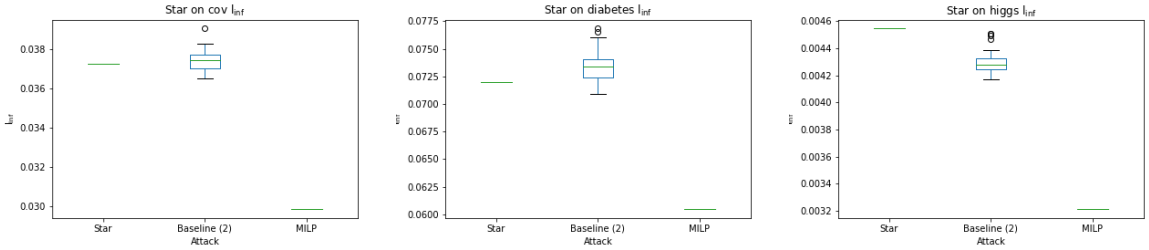


(d) webspam

Figure B.2: Plots showing the average number of random points needed to find an adversarial point as good as those found by the MILP attack. Negative scores indicate that none of the random points scored as well as the MILP attack.

C

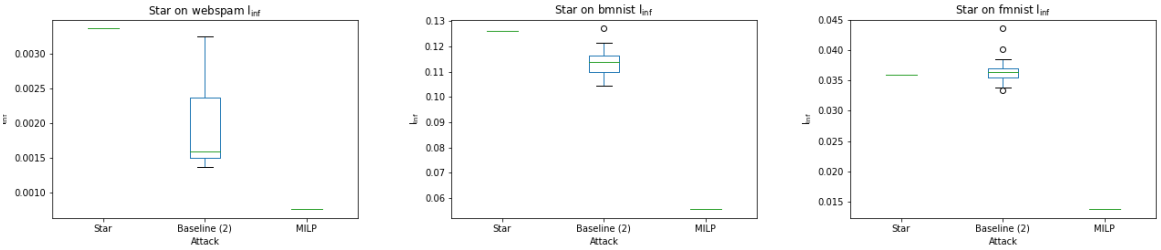
Fixed attack direction, extra plots



(a) Average adversarial distances for the cov-type dataset

(b) Average adversarial distances for the diabetes dataset

(c) Average adversarial distances for the higgs dataset



(d) Average adversarial distances for the webspam dataset

(e) Average adversarial distances for the bmnist dataset

(f) Average adversarial distances for the fmnist dataset

Figure C.1: Average adversarial distances across all 100 attacked points, per dataset. The X-axis shows the attack and the Y axis shows the adversarial distance in the L_{inf} norm. The performance of the star attack is compared to the baseline with 2 random points and the optimal answer of the MILP attack.

Bibliography

[1] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.

[2] Wieland Brendel, Jonas Rauber, and Matthias Bethge. “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models”. In: *arXiv preprint arXiv:1712.04248* (2017).

[3] Nicholas Carlini and David Wagner. “Towards evaluating the robustness of neural networks”. In: *2017 ieee symposium on security and privacy (sp)*. Ieee. 2017, pp. 39–57.

[4] Hongge Chen et al. “Robustness verification of tree-based models”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[5] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. “Hopskipjumpattack: A query-efficient decision-based attack”. In: *2020 ieee symposium on security and privacy (sp)*. IEEE. 2020, pp. 1277–1294.

[6] Minhao Cheng et al. “Sign-opt: A query-efficient hard-label adversarial attack”. In: *arXiv preprint arXiv:1909.10773* (2019).

[7] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[9] Alex Kantchelian, J Doug Tygar, and Anthony Joseph. “Evasion and hardening of tree ensemble classifiers”. In: *International conference on machine learning*. PMLR. 2016, pp. 2387–2396.

[10] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal adversarial perturbations”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1765–1773.

[11] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. “Transferability in machine learning: from phenomena to black-box attacks using adversarial samples”. In: *arXiv preprint arXiv:1605.07277* (2016).

[12] Nicolas Papernot et al. “Distillation as a defense to adversarial perturbations against deep neural networks”. In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 582–597.

[13] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

[14] David Warde-Farley and Ian Goodfellow. “11 adversarial perturbations of deep neural networks”. In: *Perturbations, Optimization, and Statistics* 311.5 (2016).

[15] Chong Zhang, Huan Zhang, and Cho-Jui Hsieh. “An efficient adversarial attack for tree ensembles”. In: *Advances in neural information processing systems* 33 (2020), pp. 16165–16176.