FINDING A HUMAN-LIKE CLASSIFIER

A PREPRINT

Anonymous author(s)

November 13, 2019

ABSTRACT

There were many attempts to explain the trade-off between accuracy[1] and adversarial robustness[2]. However, there was no clear understanding of the behaviors of a robust classifier which has human-like robustness.
We argue (1) why we need to consider adversarial robustness against varying magnitudes of perturbations not only focusing on a fixed perturbation threshold, (2) why we need to use different method to generate adversarially perturbed samples that can be used to train a robust classifier and measure the robustness of classifiers and (3) why we need to prioritize adversarial accuracies with different magnitudes.
We introduce Lexicographical Genuine Robustness (LGR) of classifiers that combines the above requirements. We also suggest a candidate oracle classifier[3] called "Optimal Lexicographically Genuinely Robust Classifier (OLGRC)" that prioritizes accuracy on meaningful adversarially perturbed examples generated by smaller magnitude perturbations. The training algorithm for estimating OLGRC requires lexicographical optimization [2] unlike existing adversarial training methods [3]. To apply lexicographical optimization to neural network, we utilize Gradient Episodic Memory (GEM) [4] which was originally developed for continual learning by preventing catastrophic forgetting [5].

Keywords adversarial examples · adversarial robustness · Lexicographical Genuine Robustness (LGR) · lexicographical optimization · Gradient Episodic Memory (GEM)

1 Introduction

Even though deep learning models have shown promising performances in image classification tasks [6], most deep learning classifiers mis-classify imperceptibly perturbed images, i.e. adversarial examples [7]. This vulnerability can occur even when the adversarial attacks were applied before they print the images, and the printed images were read through a camera [8]. That result shows real-world threats of classifiers can exist. In addition, adversarial examples for a classifier can be transferable to other models [3]. This transferability of adversarial examples [9] enables attackers to exploit a target model with limited access to the target classifier. This kinds of attacks is called black-box attacks.

[1] Accuracy: It refers to the accuracy of (unperturbed) original samples. It is also called 'natural accuracy' or 'standard accuracy' of a classifier.

[2] Adversarial robustness: It refers to the accuracy on the adversarially perturbed samples. It is also called 'robustness', 'robust accuracy' or 'adversarial accuracy' of a classifier. Mathematical definition is in definition 4.

[3] Oracle classifier: It refers to the hypothetically ultimate classifier. Often, human classification is considered as an oracle classifier even though there is some debate on this view [1].

1.0.1 Adversarially perturbed samples

An adversarially perturbed sample refers to the result of the perturbation (adversary generation) methods that has increased adversarial loss usually starting from an original sample. It is important to notice that an adversarially perturbed sample of a classifier may not be an adversarial example, which will be explained later in subsection 1.1. It can be just a non-adversarial perturbed sample (see Figure 4). Adversary generation methods try to effectively increase adversarial loss using the available information of the target classifier. Methods to generate adversarially perturbed samples include Fast Gradient Sign Method (FGSM) [3], Basic Iterative Method (BIM) [8], Projected Gradient Descent (PGD) [10], Distributionally Adversarial Attack (DAA) [11] and Interval Attack [12].

We will use the following terminology for the following paragraphs unless we specify otherwise. x is an original sample, y is corresponding label for x, ϵ is the perturbation norm, sign indicates the element-wise sign function and $L(\theta, x, y)$ is the loss of the classifier parameterized by θ.

Fast Gradient Sign Method (FGSM) [3] generates the adversarial result x'_{FGSM} with the following formula. $x'_{FGSM} = x + \epsilon \text{sign}\left[\nabla_x L(\theta, x, y)\right]$. FGSM was suggested from the hypothesis that linear behaviors of classifiers are enough to cause adversarial susceptibility of models. The formula was obtained by applying local linearization of the cost function and finding the optimal perturbation. Note that we only show formula for l_∞ norm (max norm) attacks, but we can easily get the formula for other attacks when we replace the sign function with identity function or others.

In order to get the strongest attacks using first order information of the models, Projected gradient descent (PGD) generates the adversarial result x'_{PGD} by applying iterative steps like the following. $x^{(t+1)} = \prod_{x+S}\left[x^{(t)} + \alpha \text{sign}\left[\nabla_{x^{(t)}} L(\theta, x^{(t)}, y)\right]\right]$ where $x + S$ is the set of allowed perturbation region for sample x that is limited by ϵ, $x^{(0)}$ is the random starting points in $x + S$, $\prod_{x+S}[]$ means the projection result on $x + S$ and α is the step size [10]. Note that we also only show formula for l_∞ norm attacks. Basic Iterative Method (BIM) [8] use the same iterative steps with PGD method except that it start from a fixed starting point, i.e. $x^{(0)} = x$ for BIM.

1.0.2 Adversarial training

Adversarial training [3] was developed to avoid adversarial vulnerability of a classifier. It tries to reduce the weighted summation of standard loss (empirical risk) $\mathbb{E}\left[L(\theta, x, y)\right]$ and adversarial loss $\mathbb{E}\left[L(\theta, x', y)\right]$, i.e. $\alpha\mathbb{E}\left[L(\theta, x, y)\right] + (1-\alpha)\mathbb{E}\left[L(\theta, x', y)\right]$ where α is a hyperparameter for adversarial training, and x' is an adversarially perturbed sample from x with $\|x' - x\| \leq \epsilon$. (Usually, $\alpha = 0.5$ is used for adversarial training.) By considering both standard and adversarially perturbed samples, adversarial training try to increase accuracies on both clean[3] and adversarially perturbed samples. In the literatures on adversarial training, inner maximization of a classifier refers to generating adversarial attacks, i.e. generating adversarially perturbed samples x^* that maximally increase the loss. And outer minimization refers to minimizing the adversarial loss of the model. Madry et al. [10] explained that inner maximization and outer minimization of the loss can train models that are robust against adversarial attacks.

However, adversarial training [3] has shown some issues. As some researches on adversarial robustness explained the trade-offs between accuracy on clean data and adversarial robustness [13–16], when we used adversarial training, we can get a classifier whose accuracy is lower than using standard (non-adversarial) training method [13, 17]. Also, a research studied samples whose perceptual classes are changed due to perturbation, but not in the model's prediction, what they called "invariance-based adversarial examples" [18]. They found that classifiers trained with adversarial training can be more susceptible to invariance-based adversarial examples.

1.0.3 Toward a human-like classification

We define three properties of human-like classification: (1) human-like classification is robust against varying magnitudes of adversarially perturbed samples and not just on a fixed maximum norm perturbations, (2) when we think about adversarially perturbed samples with increasing magnitudes, a human-like classifier does not consider already considered samples multiple times and (3) human-like classification prioritizes adversarial accuracies with smaller

[3] clean: In most of the literature about adversarial examples and robustness, the term 'clean' means 'unperturbed', 'unchanged' and 'original'. Whether a sample is clean or not (perturbed) will be determined by its history. A clean sample x_1 and a perturbed sample x'_2 originated from a clean sample x_2 can have exactly the same elements even if their histories are different. In general, if two samples have exactly same elements, we say the two samples are currently equivalent regardless of their histories.

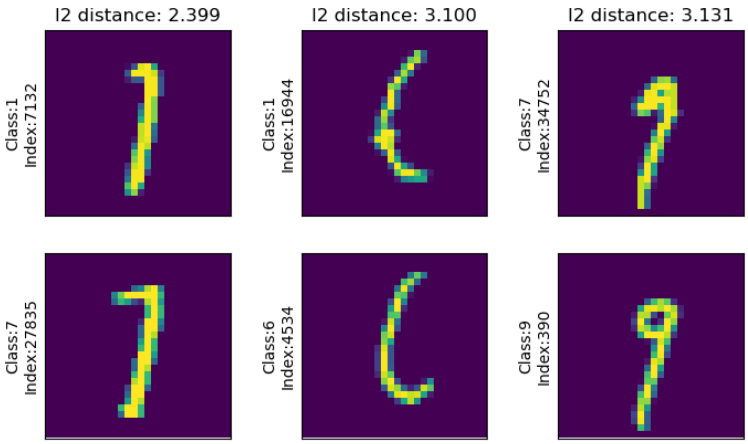


Figure 1: Examples of confusing near image pairs with different classes of MNIST training dataset [20]. The l_2 norms of the pairs are 2.399, 3.100 and 3.131 from left to right. From these examples, we can say the exclusive belongingness assumption may not be realistic.

perturbation norm.

The objective of this paper is to design and train a classifier whose robustness is more resemblance to robustness of human than a model trained by standard adversarial training [3]. We introduce Lexicographical Genuine Robustness (LGR) of classifiers to combine three properties. Using LGR can prevent the problem of training classifiers with lower accuracy on clean data by considering the third property. LGR also enable to avoid what we named "pseudo adversarial examples" of models which are conceptually similar to invariance-based adversarial examples [18]. From LGR, we introduce a candidate oracle classifier called "Optimal Lexicographically Genuinely Robust Classifier (OLGRC)".

1.1 Definition of adversarial examples

We know move on to more precise definition of adversarial examples and the detailed explanation of our ideas. The definition of adversarial example by Biggio et al. [19] was used in many theoretical analysis of adversarial robustness [13–16]. These analyses showed adversarial examples are inevitable and there is a trade-off between accuracy on clean data and adversarial robustness, i.e. accuracy on adversarially perturbed samples. However, we argue why simply increasing adversarial robustness can get a classifier whose behavior is different from humans.

Problem setting 1. *In a clean input set $\mathcal{X} \subseteq \mathbb{R}^d$, let every sample x exclusively belong to one of the classes $\mathcal{Y}$, and their classes will be denoted as c_x. A classifier f assigns a class label from $\mathcal{Y}$ for each sample $x \in \mathbb{R}^d$. Assume f is parameterized by θ and $L(\theta, x, y)$ is the loss of the classifier provided the input x and the label $y \subseteq \mathcal{Y}$.*

Note that this exclusive belonging assumption is introduced to simplify the analysis and it can be unrealistic. In a real situation, 1) input information might not be enough to perfectly predict the class, 2) input samples might contain noises which erase class information, 3) some input samples might be better to give non-exclusive class (see Figure 1), or 4) sometimes labels might also contain some noises due to mistakes.

Definition 1. *Given a clean sample $x \in \mathcal{X}$ and a maximum permutation norm (threshold) ϵ, a perturbed sample x' is an adversarial example by the definition of Biggio et al. [19] if $\| x - x' \| \leq \epsilon$ and $f(x') \neq c_x$.*

Note that some perturbed samples x' and some adversarial examples may also belong to $\mathcal{X}$. Although generated by adversary generation methods mentioned in subsection 1.0.1, perturbed samples are not necessarily adversarial examples (see Figure 4). For example, when allowed perturbation norm ϵ is too small, predicted class of adversarially perturbed samples x' can be c_x. We are only focus on the analysis using l_p norm for measuring the distance, but the concept of adversarial examples and our analysis are not confined to these metrics. Many ideas in our analysis can

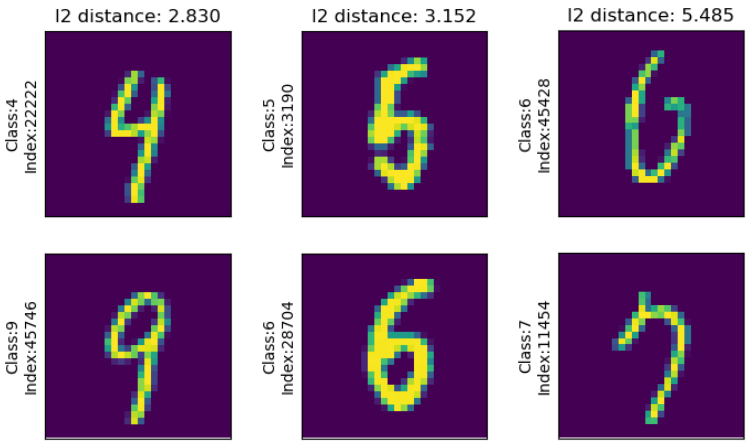


Figure 2: Examples of near image pairs with different classes of MNIST training dataset [20]. The l_2 norms of the pairs are 2.830, 3.152 and 5.485 from left to right.

might be applied to adversarial examples based on relaxed versions of metrics.

1.2 Classification of MNIST dataset and definitions

Let's consider the classification task on MNIST dataset [20]. We will use the norms calculated by viewing an image input as a (flatten) 784 dimensional vector. The smallest l_2 norm[5] of the image pair for different class on training data is 2.399. However, as the l_2 norm of the nearest image pair for class 6 and 7 on training data is 5.485, when we train a classifier using adversarial training with $\epsilon = 2.399$ (one can use half of the minimum distance $\epsilon = \frac{2.399}{2} = 1.200$ to consider the distance between decision boundary and the samples in different classes, but our explanation doesn't consider that approach), the trained classifier might mis-classifies a perturbed image of digit 6 as an image of digit 7 when perturbation norm is 2.5 (> 2.399) even if such perturbation norm is smaller than the half (2.742) of the expected minimum norm 5.485. Hence, we might want a classifier who is also robust when ϵ is larger than 2.399.

If we want a classifier which has no adversarial example when $\epsilon = 5.4 < 5.485$, we need to have a classifier that outputs original class for every training image and perturbed images when norms of the perturbations are at most 5.4. However, the l_2 norm of the nearest image pair for class 4 and 9 on training data is 2.830 (see Figure 2). What does it mean? As the image on the bottom left can also be considered as an adversarial example perturbed from top left, the classifier needs to classify its class is 9 when it was an original image and its class is 4 when it was an adversarial example. Can we have a classifier with such psychometric power that knows the previous history of images? Do we have such psychometric power? Or, are we not robustness enough [1] even for classifying MNIST data [20]? The more important question we need to ask is "Do we really want such kinds of ability?". And we answer the answer is "No!". This confusion arises from the gap between our intuitive understanding [7] and Biggio et al. [19]'s definition of adversarial examples. (Note that the reason we encounter these kinds of problems is not because we are doing multiclass classification. A similar problem can occur in binary classification problems as shown in Figure 3.)

Even though intuitive definition of adversarial examples are samples which are generated by applying imperceptible perturbations to clean samples [7], by relying on the Biggio et al. [19]'s definition of adversarial examples, some theoretical analyses tried to analyze the adversarial robustness even when the norms of the adversarial perturbations are big enough so that they can change the perceptual classes of samples.

Definition 2. *Let us distinguish two kinds of class for a given clean sample $x \in \mathcal{X}$ and its corresponding perturbed sample x'.*

[5] l_∞ norm was more commonly used in the literature as l_∞ norm of the perturbation should be large in order to change the perceptual class [21]. However, a classifier trained by adversarial training with l_∞ adversary is susceptible to adversarial attacks with l_0 or l_2 adversary [22] which suggests that we also need to consider l_0 and l_2 norm robustness.

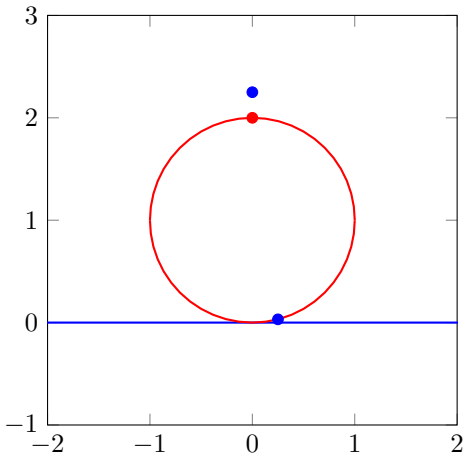


Figure 3: Binary classification example for class A: red and class B blue. We are considering l_2 norm robustness. We can think of a classifier that classifies points inside the circle classified as class A and class B otherwise. The accuracy of the classifier is 1. However, if the maximum perturbation norm $\epsilon > 0$, the classifier will always have adversarial examples. For example, point $(0, 2 + \epsilon)$ is an adversarial example as it is classified as class B according to the classifier and $(0, 2)$ has class A. If we try to use adversarial training, we encounter the problem where point $(\frac{\epsilon}{2}, 1 - \sqrt{1 - \frac{\epsilon^2}{4}})$ (or (1,1) when $\epsilon > 2$) can also be considered as an adversarial example originated from $(\frac{\epsilon}{2}, 0)$. From this example, we can see that only considering one maximum perturbation threshold is not enough for robustness of a classifier even for binary classification problems. (Visualized points are based on $\epsilon = 0.5$. The decision boundary of oracle classifier for this example will be a parabola which satisfies $x_2 = \frac{1}{4}x_1^2$ where x_1 and x_2 represent the first and second axes. Every point in this parabola has same distance to the nearest sample in class A and B.)

- *De facto class: c_x of the clean sample x. $c_{x'}$ of the perturbed sample x' if $x' \in \mathcal{X}$. De facto class is undefined for some perturbed sample $x' \in \mathcal{X}^C = \mathbb{R}^d - \mathcal{X}$.*

- *De jure class: c_x of the clean sample x. c_x of the perturbed sample x', i.e. original class of the perturbed example x'.*

Intuitively speaking, de facto class of a sample is current perceptual class of a sample. The name "De jure" can be debatable and confusing, but it follows the tradition of the researchers who consider the original class of a perturbed sample as a legitimate class and try to increase robustness based on that even if the perturbation can change the de facto class.

One thing to notice is that we can change the de facto class by perturbating a clean sample x when large perturbation is allowed, but we can't change the de jure class of it. De facto class and de jure class are not dependent on the classifier f.

Definition 3. *Furthermore, we distinguish two kinds of adversarial example x' when x was the original sample of x' before the adversarial perturbation.*

- *Pseudo adversarial example: an adversarial example x' by definition 1 whose de facto class is different from its de jure class, i.e. $c_{x'} \neq c_x$.*

- *Genuine adversarial example: an adversarial example x' by definition 1 whose de facto class is undefined, i.e. $x' \in \mathcal{X}^C$.*

Note that even though the classifier f determines whether a given perturbed sample x' is an adversarial example or not, it doesn't affect whether an adversarial example x' is a pseudo adversarial example or genuine adversarial example.

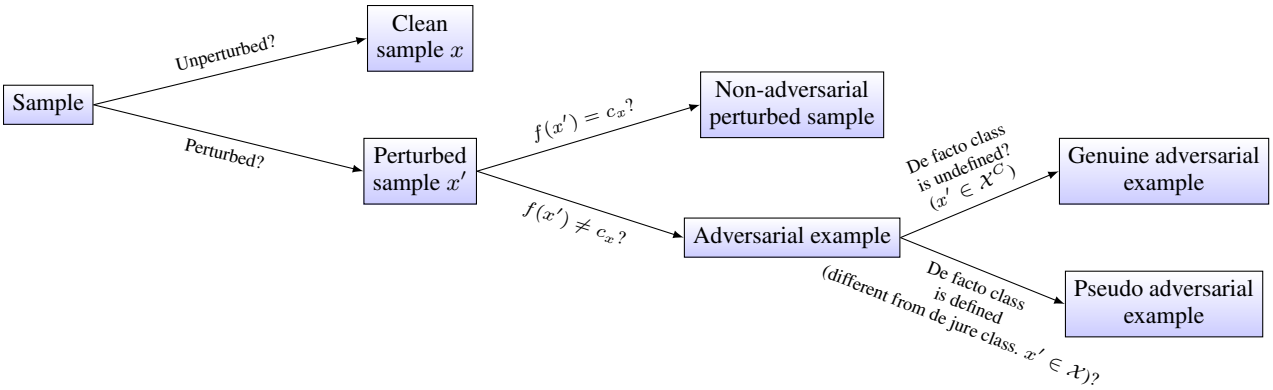


Figure 4: Classification tree of a sample.
For a given sample, the history (whether it was perturbed or not) of a sample will determine whether the sample is a clean sample x or a perturbed sample x'. For a perturbed sample x', its de jure class c_x and the classifier f will determine whether it is a non-adversarial perturbed sample or an adversarial example. Finally, the existence of de facto class will determine whether an adversarial example is a genuine or pseudo adversarial example.

1.3 Understanding our definitions on MNIST dataset [20]

With our definitions, let's consider again the classification task on MNIST dataset [20] (see Figure 2). When we think about adversarial examples for $\epsilon = 5.4$, again, the image on the bottom left can be considered as an adversarial example perturbed from the top left image. As its de facto class is 9 no matter it's a clean or adversarial example, it is a pseudo adversarial example of top left image if it was an adversarial example.

Do we want our classifier to be robust against pseudo adversarial examples? Short answer to this question is "No, we don't need to.". When we consider the classification process of humans, we do not care about whether a given sample was a clean or perturbed sample, i.e. the previous history of the sample. We only care about the most likely class of the current sample and such class is close to the concept of de facto class. And this principle was commonly used in many visual assessment of adversarial robustness [10, 13, 23–25] even if some of them follow the definition of adversarial example of Biggio et al. [19].

Let's consider a general situation where a classifier f tries to increase the adversarial robustness for a perturbation norm ϵ which is large enough so that the perturbation results can change the de facto classes of some samples. In other words, the classifier tries to assign de jure class even for pseudo adversarial examples. This implies that the classifier tries to assign perceptually wrong classes for pseudo adversarial examples who are currently equivalent to clean examples, and this will decrease accuracy on clean data without increasing human-like robustness on these samples. Hence, not only we don't need to increase robustness against pseudo adversarial examples, but also we should avoid increasing robustness against them in order to get a model with human-like robustness (Note that robustness will be calculated by de jure classes of pseudo adversarial examples).

Let's compare the training tasks when we only have clean samples and when we only have perturbed samples. Perturbed samples can be derived from clean samples and theoretically they can take any values in their allowed perturbation regions. Because of that perturbed samples have more uncertainty than clean samples. In order words, clean samples have more information than perturbed samples. This observation can lead to a preference to prefer using clean samples when we train a model. When we think about a training task with both clean and perturbed samples, the preference will be correspond to increasing natural accuracy before we consider the accuracy on perturbed samples. This preference can be generalized to a principle that we prioritize the adversarial accuracy on smaller perturbation norm.

From the above explanations, we can summarize the properties of human classification or human-like robustness.

1. Human classification is robust against adversarially perturbed samples generated from varying magnitudes of perturbations and not just fixed maximum norm perturbations.
2. The previous history of a sample has no effect in classification. Only the current sample will determine the classification result. From this, a human-like classifier avoids assigning de jure class for pseudo adversarial

examples. More generally, a human-like classifier avoids considering already considered samples several times.

3. Human classification prioritizes the robustness for smaller perturbation norm than the robustness for larger perturbation norm.

The question arising from the second property is "How do we know a given adversarial example is a pseudo adversarial example or genuine adversarial example?". It would be trivial when we know the data distribution and predefined classes for all data like the toy example in section 2. However, in practice, we only have limited training data and hard to know the data distribution. We introduce a method to estimate whether a perturbed sample x' has de facto class or not, and thus try to avoid using pseudo adversarial examples for adversarial training and measure the robustness of classifiers. We then combine this with a lexicographical optimization method.

1.4 Mathematical definitions of accuricies

Before further diving into the adversarial robustness of classifiers, we give the mathematical definitions of the accuracies.

Definition 4. *We define natural accuracy and adversarial accuracies for given maximum perturbation norm and exact perturbation norm. Note that* $\mathbb{1}\left(\right)$ *is an indicator function which has value* 1 *if the condition in the bracket holds and value* 0 *if the condition in the bracket doesn't hold.*

- *Natural accuracy:* $\mathbb{E}_{x\in\mathcal{X}}\left[\mathbb{1}\left(f(x)=c_x\right)\right]$.
- *(Standard) Adversarial accuracy (by maximum perturbation norm):*
  $\mathbb{E}_{x\in\mathcal{X}}\left[\mathbb{1}\left(f(x^*)=c_x\right)\right]$ *where adversarially perturbed sample* $x^* = \underset{x':\left\|x'-x\right\|\leq\epsilon}{\operatorname{argmax}} L(\theta, x', c_x)$.
- *(Standard) Adversarial accuracy (by exact perturbation norm):*
  $\mathbb{E}_{x\in\mathcal{X}}\left[\mathbb{1}\left(f(x^*)=c_x\right)\right]$ *where adversarially perturbed sample* $x^* = \underset{x':\left\|x'-x\right\|=\epsilon}{\operatorname{argmax}} L(\theta, x', c_x)$.
- *Genuine adversarial accuracy (by maximum perturbation norm):*
  $\mathbb{E}_{x\in S_{max}(\epsilon)}\left[\mathbb{1}\left(f(x^*)=c_x\right)\right]$ *where* $S_{max}(\epsilon) = \left\{x\in\mathcal{X}|\exists x'\in\mathcal{X}^C : \|x'-x\|\leq\epsilon\right\}$ *and adversarially perturbed sample* $x^* = \underset{x'\in\mathcal{X}^C:\left\|x'-x\right\|\leq\epsilon}{\operatorname{argmax}} L(\theta, x', c_x)$.
- *Genuine adversarial accuracy (by exact perturbation norm):*
  $\mathbb{E}_{x\in S_{exact}(\epsilon)}\left[\mathbb{1}\left(f(x^*)=c_x\right)\right]$ *where* $S_{exact}(\epsilon) = \left\{x\in\mathcal{X}|\exists x'\in\mathcal{X}^C : \|x'-x\|=\epsilon\right\}$ *and adversarially perturbed sample* $x^* = \underset{x'\in\mathcal{X}^C:\left\|x'-x\right\|=\epsilon}{\operatorname{argmax}} L(\theta, x', c_x)$.

Note that the only difference of adversarial accuracies by maximum perturbation norm and exact perturbation norm is that their allowed regions of adversarially perturbed sample x^*, i.e. $x' : \left\|x^{'} - x\right\| \leq \epsilon$ vs. $x' : \left\|x^{'} - x\right\| = \epsilon$. The reason why we are separating them will be explained later. Due to the additional requirement $x'\in\mathcal{X}^C$ in adversarially perturbed sample x', pseudo adversarial examples will not be considered in genuine adversarial accuracy and thus give more meaningful adversarial accuracy. Depending on $\mathcal{X}$, genuine adversarial accuracies can be undefined. In other word, genuine adversarial accuracies will be undefined when $S_{max} = \varnothing$ or $S_{exact} = \varnothing$.

Definition 5. *We define adversarial accuracy functions* $a : [0,\infty) \to [0,1]$ *for a classifier* f*. These functions are defined by measuring adversarial accuracies with varying perturbation norms, but genuine adversarial accuracy function uses slightly modified formula.*

- *(Standard) Adversarial accuracy function (by maximum perturbation norm):*
  $a_{std;\,max}(\epsilon) = \mathbb{E}_{x\in\mathcal{X}}\left[\mathbb{1}\left(f(x^*)=c_x\right)\right]$ *where* $x^* = \underset{x':\left\|x'-x\right\|\leq\epsilon}{\operatorname{argmax}} L(\theta, x', c_x)$.

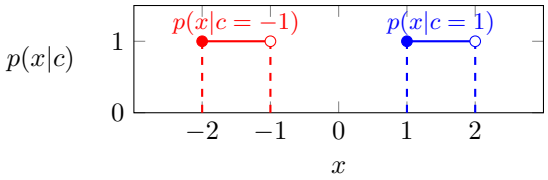


Figure 5: Plots of $p(x|c=-1)$: red and $p(x|c=1)$: blue for first toy example.

- *(Standard) Adversarial accuracy function (by exact perturbation norm):*
  $a_{std;\,exact}(\epsilon) = \mathbb{E}_{x\in\mathcal{X}}\left[\mathbb{1}\left(f(x^*) = c_x\right)\right]$ *where* $x^* = \underset{x':\,\|x'-x\|=\epsilon}{\operatorname{argmax}} L(\theta, x', c_x)$.

- *Genuine adversarial accuracy function (by exact perturbation norm):*
  $a_{gen;\,exact}(\epsilon) = \mathbb{E}_{x\in S_{exact}(\epsilon)}\left[\mathbb{1}\left(f(x^*) = c_x\right)\right]$ *where* $S_{exact}(\epsilon) = \left\{x \in \bar{\mathcal{X}} | \exists x' \in {\mathcal{X}_\epsilon}^C : \|x'-x\| = \epsilon\right\}$, *previously allowed perturbation region* $\mathcal{X}_\epsilon = \left\{x' \in \mathbb{R}^d : \|x'-x\| < \epsilon \text{ where } x \in \mathcal{X}\right\}$ *and* $x^* = \underset{x'\in\mathcal{X}_\epsilon^C:\,\|x'-x\|=\epsilon}{\operatorname{argmax}} L(\theta, x', c_x)$ *when* $\epsilon > 0$.
  $a_{gen;\,exact}(0) = \mathbb{E}_{x\in\mathcal{X}}\left[\mathbb{1}\left(f(x) = c_x\right)\right]$ *when* $\epsilon = 0$.

Likewise, the only difference of adversarial accuracy functions by maximum perturbation norm and exact perturbation norm is that their allowed regions of adversarially perturbed sample x^*. Adversarial accuracy function will be also called the change of adversarial accuracy. Genuine adversarial accuracy function will be conventionally also called the change of genuine adversarial accuracy even if it is not strictly correct. We don't define genuine adversarial accuracy function by maximum perturbation norm. One thing to notice in the $S_{exact}(\epsilon)$ in the definition of genuine adversarial accrucacy function is that it use $\bar{\mathcal{X}}$, i.e. the closure of $\mathcal{X}$. The reason we are using $\bar{\mathcal{X}}$ instead of $\mathcal{X}$ will be explained in subsection 2.2. The additional requirement used in genuine adversarial accuracy function was $x' \in \mathcal{X}_\epsilon^C = \mathbb{R}^d - \mathcal{X}_\epsilon$ rather than $x' \in \mathcal{X}^C$. It is because we consider the situation where we continuously increase the exact perturbation norm ϵ and we want to ignore already considered points for calculation of adversarial accuracy with smaller perturbation norm. This can also be considered as using samples in previously allowed perturbation region $\mathcal{X}_\epsilon$ as a new clean input set $\mathcal{X}' = \mathcal{X}_\epsilon$.

2 Toy example

Let's think about a toy example (see Figure 5) with predefined (pre-known) classes in order to simplify the analysis. There are only two classes -1 and 1, i.e. $\mathcal{Y} = \{-1, 1\}$, and 1-dimensional clean input set $\mathcal{X} = [-2, -1) \cup [1, 2) \subseteq \mathbb{R}$. $c_x = -1$ when $x \in [-2, -1)$ and $c_x = 1$ when $x \in [1, 2)$. $p(c=-1) = p(c=1) = \frac{1}{2}$, i.e. we assume uniform prior probability.

Let's define three classifiers f_1, f_2 and f_3 for this toy example (see Figure 6). When step function $\text{step}(x)$ is defined as $\text{step}(x) = \begin{cases} 1, & \text{if } x \geq 0, \\ -1, & \text{if } x < 0 \end{cases}$, let $f_1(x) = \text{step}(x-1)$, $f_2(x) = 1 - \text{step}(x+4) + \text{step}(x)$, and $f_3(x) = \text{step}(x)$. Notice that natural accuracy for all three classifiers is 1.

2.1 Changes of adversarial accuracy on the toy example

We now explain the change of adversarial accuracy for $f_1(x)$ by exact perturbation norm ϵ (see top right of Figure 7). When $0 < \epsilon \leq 1$, we can change the predicted class for $x \in [1, 1+\epsilon)$ by subtracting ϵ, and we can't change the predicted class for $x \notin [1, 1+\epsilon)$, thus standard adversarial accuracy will be $1 - \frac{1}{2}\epsilon$. When $1 < \epsilon \leq 2$, there will be same amount of adversarial examples with $\epsilon = 1$, thus (standard) adversarial accuracy will be $1 - \frac{1}{2} = \frac{1}{2}$. When $2 < \epsilon \leq 3$, we can still change the predicted class for $x \in [1, 2)$ by subtracting ϵ. Addition to that we can also change the predicted class for $x \in [1-\epsilon, -1)$ by adding ϵ and (standard) adversarial accuracy will be $-\frac{1}{2}\epsilon + \frac{3}{2}$. Adversarial accuracy is 0 for $\epsilon = 3$ and this holds also for $\epsilon > 3$.

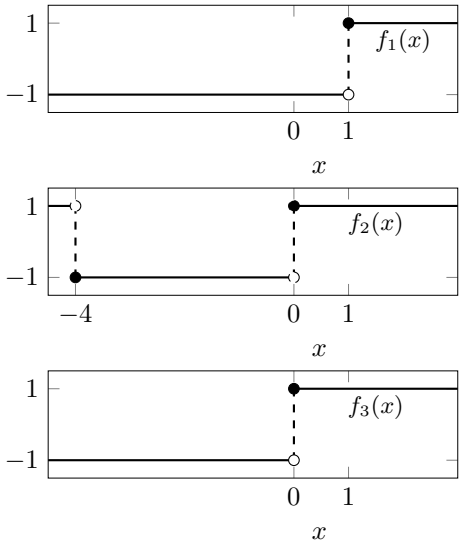


Figure 6: Plots of three classifiers. Top: $f_1(x) = \text{step}(x-1)$, Middle: $f_2(x) = 1 - \text{step}(x+4) + \text{step}(x)$, Bottom: $f_3(x) = \text{step}(x)$ where $\text{step}(x) = 1$ for $x \geq 0$ and $\text{step}(x) = -1$ for $x < 0$.

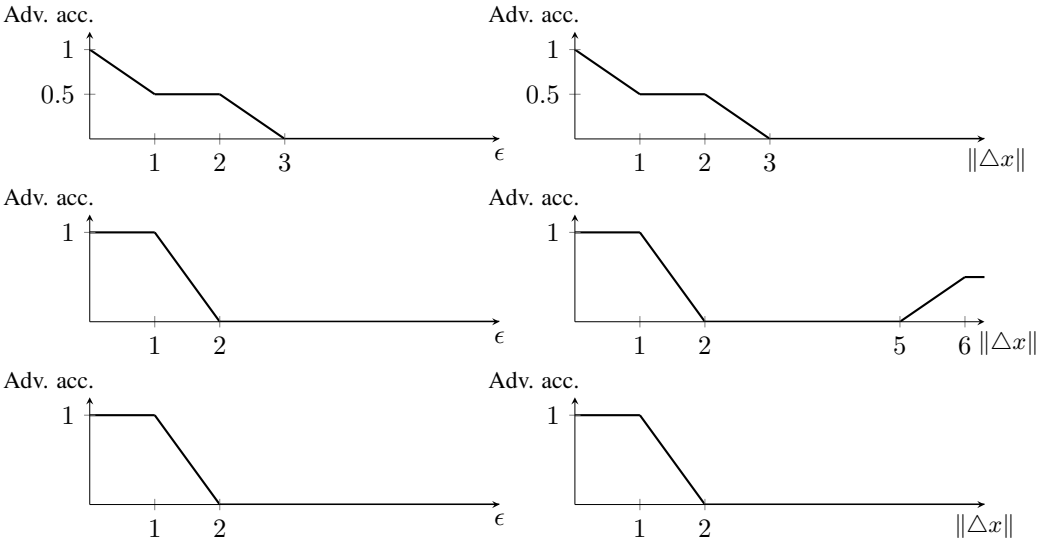


Figure 7: Top: Change of (standard) adversarial accuracy for $f_1(x)$ by maximum perturbation norm ϵ (left) and exact perturbation norm $\|\triangle x\|$ (right) where $\triangle x = x' - x$, Middle: Change of adversarial accuracy for $f_2(x)$ by maximum perturbation norm ϵ (left) and exact perturbation norm $\|\triangle x\|$ (right), Bottom: Change of adversarial accuracy for $f_3(x)$ by maximum perturbation norm ϵ (left) and exact perturbation norm $\|\triangle x\|$ (right).
Observed behaviors of f_2 and f_3 will be same when we compare the adversarial accuracy by maximum perturbation norm ϵ, however, observed behaviors of f_2 and f_3 are different when we compare the adversarial accuracy by exact perturbation norm $\|\triangle x\|$.

When we think about the change of adversarial accuracy for $f_2(x)$ by exact perturbation norm ϵ, by similar analysis, we can check it will be look like middle right graph in Figure 7 when $\epsilon \leq 5$. However, intriguing phenomenon occurs when $\epsilon > 5$. When $5 < \epsilon \leq 6$, $x \in [1, \epsilon - 4)$ cannot change the predicted class as subtracting or adding ϵ will result in the same class 1, thus adversarial accuracy will be $\frac{\epsilon-5}{2}$. If $\epsilon \geq 6$, adversarial accuracy will be $\frac{1}{2}$.

The change of adversarial accuracy for $f_3(x)$ by exact perturbation norm ϵ can be understand similarly with $f_2(x)$.

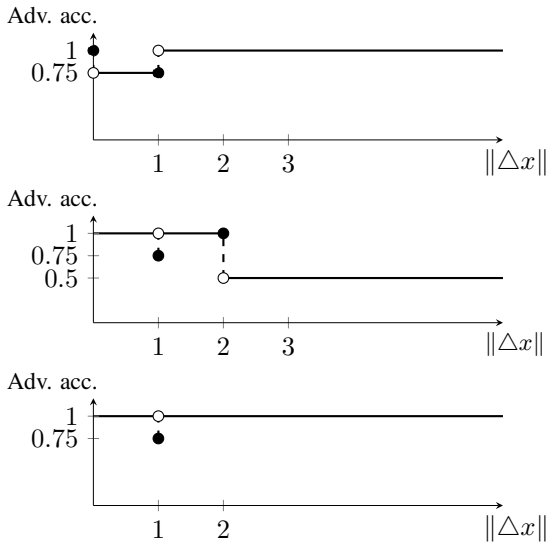


Figure 8: Top: Change of genuine adversarial accuracy for $f_1(x)$ by exact perturbation norm $\|\triangle x\|$, Middle: Change of genuine adversarial accuracy for $f_2(x)$ by exact perturbation norm $\|\triangle x\|$, Bottom: Change of genuine adversarial accuracy for $f_3(x)$ by exact perturbation norm $\|\triangle x\|$.

2.2 Changes of genuine adversarial accuracy on the toy example

Now, we move on to the explanation for the changes of genuine adversarial accuracy for $f_1(x)$, $f_2(x)$ and $f_3(x)$ (see Figure 8). When $0 < \epsilon \leq 1$, previously allowed perturbation region $\mathcal{X}_\epsilon = (-2-\epsilon, -1+\epsilon) \cup (1-\epsilon, 2+\epsilon)$. When $\epsilon > 1$, previously allowed perturbation region $\mathcal{X}_\epsilon = (-2-\epsilon, 2+\epsilon)$. For calculation of genuine adversarial accuracies, we will consider four points, i.e. $S_{exact}(\epsilon) = \{-2-\epsilon, -1+\epsilon, 1-\epsilon, 2+\epsilon\}$, when $0 < \epsilon \leq 1$ (point 0 will be counted twice when $\epsilon = 1$) and two points, i.e. $S_{exact}(\epsilon) = \{-2-\epsilon, 2+\epsilon\}$, when $\epsilon > 1$. Note that if we did not use closure in the definition of $S_{exact}(\epsilon)$, $S_{exact}(\epsilon) = \{-2-\epsilon, 1-\epsilon\}$, when $0 < \epsilon \leq 1$ and $S_{exact}(\epsilon) = \{-2-\epsilon\}$, when $\epsilon > 1$. This will ignore many points and can not measure proper robustness of classifiers.

In the change of genuine adversarial accuracy for $f_1(x)$, when $0 < \epsilon \leq 1$, $-2-\epsilon$, $-1+\epsilon$ and $2+\epsilon$ will be non-adversarial perturbed samples and $1-\epsilon$ will be adversarial example, and thus $a_{gen;exact}(\epsilon) = \frac{3}{4} = 0.75$. When $\epsilon > 1$, $-2-\epsilon$ and $2+\epsilon$ will be non-adversarial perturbed samples, and thus its genuine adversarial accuracy is 1.

When considering the change of genuine adversarial accuracy for $f_2(x)$, for $0 < \epsilon < 1$, $-2-\epsilon$, $-1+\epsilon$, $1-\epsilon$ and $2+\epsilon$ will be non-adversarial perturbed samples, and thus $a_{gen;exact}(\epsilon) = 1$. When $\epsilon = 1$, $-2-\epsilon$, $1-\epsilon$ and $2+\epsilon$ will be non-adversarial perturbed samples and $-1+\epsilon$ will be adversarial example, and thus $a_{gen;exact}(1) = \frac{3}{4} = 0.75$ (Actually, $1-\epsilon = 0 = -1+\epsilon$, but they counted twice.). When $1 < \epsilon \leq 2$, $-2-\epsilon$ and $2+\epsilon$ will be non-adversarial perturbed samples, and thus $a_{gen;exact}(\epsilon) = 1$. However, when $\epsilon > 2$, only $2+\epsilon$ will be non-adversarial perturbed samples and $-2-\epsilon$ will be adversarial example, and thus $a_{gen;exact}(\epsilon) = \frac{1}{2} = 0.5$.

Through similar process, one can understand the change of genuine adversarial accuracy for $f_3(x)$.

3 A candidate oracle classifier: Optimal Lexicographically Genuinely Robust Classifier (OLGRC)

We introduce Lexicographical (Standard) Robustness (LSR or LR) which is a total preorder based on adversarial accuracy functions by the exact perturbation norm ϵ. Furthermore, we explain why LSR is not enough to specify a human-like classifier and why we need Lexicographical Genuine Robustness (LGR). From this, we suggest a candidate oracle classifier what we called "Optimal Lexicographically Genuinely Robust Classifier (OLGRC)".

3.1 Lexicographical Standard Robustness (LSR)

Let's say we have two classifiers f_1 and f_2 for given data $D \subseteq \mathcal{X} \times \mathcal{Y}$ (Here, we are considering general classifiers and not f_1 and f_2 for our toy example.). Let $a_1, a_2 : [0, \infty) \to [0, 1]$ be the corresponding standard adversarial accuracy functions by exact perturbation norm ϵ for f_1 and f_2, respectively.

Definition 6. *We define a total preorder of classifiers called Lexicographical Standard Robustness (LSR).*

- *We say "f_2 is lexicographically more robust (LR) than f_1" or denote "$f_2 >_{LR} f_1$" or "$f_2 >_{LSR} f_1$" iff* $\exists d \geq 0 : a_1(\epsilon) = a_2(\epsilon), \forall \epsilon < d$ *and* $a_1(d) > a_1(d)$.

- *"f_2 is lexicographically equivalently robust (LR) with f_1" or denote "$f_2 =_{LR} f_1$" or "$f_2 =_{LSR} f_1$" iff* $a_1(\epsilon) = a_2(\epsilon), \forall \epsilon \in [0, \infty)$.

The reason why we consider adversarial robustness against varying magnitudes of perturbations and not a fixed maximum perturbation norm ϵ is that increasing robustness on a fixed maximum perturbation norm ϵ will not give a classifier that has human-like robustness as explained in 1.2 (The first property in 1.3.). The defined (total) preorder prioritizes the robustness for smaller perturbation norm because more information in the samples can be lost when larger perturbation is allowed, and thus adversarial accuracy for larger perturbation norm is less important (The third property in 1.3.). This prioritization is also related to the observation that we need to avoid increasing robustness against pseudo adversarial examples who are more likely to occur when the magnitude of the perturbation is large (It is also connected to the second property in 1.3, but in an incomplete way as samples used for adversarial accuracy with small perturbation magnitude can be repeatedly used for larger perturbation magnitudes.). Furthermore, there is also a reason for using adversarial accuracy by exact perturbation norm not by maximum perturbation norm. That was because using adversarial accuracy by exact perturbation norm enables further discretibility as shown in Figure 7.

Let's go back to the toy example 2 and three classifiers f_1, f_2 and f_3 for that toy example. According to the Lexicographical Standard Robustness (LSR), we have $f_1 <_{LR} f_3 <_{LR} f_2$. Then, can we say f_3 is better than f_1, and f_2 is better than f_3? Well, it is true in terms of Standard Robustness only. However, in the following subsection 3.2, we argue why f_2 can be better than f_3 in other aspects. One thing to note here is that if we define $f_4(x) = \text{step}(x) - \text{step}(x - 4)$, we can check $f_2 =_{LR} f_4$ while $f_2 \neq f_4$. Hence, LSR doesn't have an antisymmetric property, thus it is not a partial order.

3.2 Lexicographical Genuine Robustness (LGR)

In the previous subsection, we explained that the total preorder based on Lexicographical Standard Robustness (LSR) can handle the first and third properties in subsection 1.3, but only incompletely for the second property. To also handle the second property, we use genuine adversarial accuracy function which ignores already considered points for calculation of adversarial accuracy.

Let's say we have two classifiers f_1 and f_2 for given data $D \subseteq \mathcal{X} \times \mathcal{Y}$ (Again, we are referring general classifiers and not f_1 and f_2 for our toy example.). Let $a_1, a_2 : [0, \infty) \to [0, 1]$ be the corresponding genuine adversarial accuracy functions by exact perturbation norm ϵ for f_1 and f_2, respectively.

Definition 7. *We define a total preorder of classifiers called Lexicographical Genuine Robustness (LGR).*

- *We say "f_2 is lexicographically genuinely more robust (LGR) than f_1" or denote "$f_2 >_{LGR} f_1$" iff* $\exists d \geq 0 : a_1(\epsilon) = a_2(\epsilon), \forall \epsilon < d$ *and* $a_2(d) > a_1(d)$.

- *"f_2 is lexicographically genuinely equivalently robust (LGR) with f_1" or denote "$f_2 =_{LGR} f_1$" iff* $a_1(\epsilon) = a_2(\epsilon), \forall \epsilon \in [0, \infty)$.

Let's go back again to the toy example 2 and classifiers f_1, f_2 and f_3 for that toy example.

According to the Lexicographical Genuine Robustness (LGR), we have $f_1 <_{GLR} f_2 <_{GLR} f_3$.

Let's consider the perturbations needed to change the predicted classification results. Similar to the gradients of differentiable function, the perturbations can be considered as interpretations of classifier as they can change the

predicted class. When we think about changing de facto classes, we need positive perturbation $\triangle x = x' - x \in (2, 4)$ in order to change class -1 to class 1, and we need negative perturbation $\triangle x \in (-4, -2)$ in order to change class 1 to class -1. From this we can see that direction of the perturbations can explain the change of de facto classes.

Considering the perturbations needed to change the predicted classes for classifier f_3, we need positive perturbation $\triangle x \in (1, \infty)$ in order to change class -1 to class 1, and we need negative perturbation $\triangle x \in (-\infty, -1)$ in order to change class 1 to class -1. Note that direction of the perturbations can explain the change of de facto classes.

Considering the perturbations needed to change the predicted classes for classifier f_2, we need negative perturbation $\triangle x \in (-6, -1)$ in order to change class 1 to class -1. However, not only positive perturbation $\triangle x \in (1, \infty)$ can change class -1 to class 1, but also negative perturbation $\triangle x \in (-\infty, -2)$ can change class -1 to class 1. Hence, the direction of the perturbations no longer explain the change of de facto classes for f_2. We saw that the directions of the perturbations of classifier f_3 explain more the change of de facto classes than classifier f_2.

Also, when the Occam's razor principle was considered, we would prefer classifier f_3 over f_2 as they have same standard adversarial robustness for $\|\triangle x\| \leq 4$ and f_2 has one more decision boundary point than f_3, i.e. more complex than f_3.

3.3 Optimal Lexicographically Genuinely Robust Classifier (OLGRC)

Optimal Lexicographically Genuinely Robust Classifier (OLGRC) is defined as the maximal classifier based on Lexicographical Genuine Robustness (LGR), i.e. this classifier o satisfies either $o =_{LGR} g$ or $o >_{LGR} g$ for any classifier g. OLGRC is determined by expanding explored regions. If each expansion step is (almost everywhere) uniquely determined and expansion can fill the whole space $\mathbb{R}^d$, there will be unique OLGRC (in almost everywhere sense). Whether there is unique OLGRC (in almost everywhere sense) or not will be determined by the definition of the metric. We do not cover the detailed conditions for uniqueness.

The behavior of OLGRC is similar to the behavior of the support vector machine (SVM) [26] in that its boundary tries to maximize its distance (margin) to the data points. However, linear SVM can only be trained for linearly separable problems even if we assume exclusive belonging settings. On the other hand, Kernel SVM tries to maximize its distance based on the norms of the feature space. Thus, it is probably vulnerable to adversarial attacks in the input set while OLGRC tries to maximize its distance based on the norms of the input set in order to increase adversarial robustness.

When we think about the problem setting in the toy example 2, the classifier f_3 is the OLGRC as it's impossible to have a classifier whose change of genuine adversarial accuracy is higher than f_3.

4 Training method to find Optimal Lexicographically Genuinely Robust Classifier (OLGRC)

We are going to use $l_1, l_2, \cdots$ to denote loss functions in this section unlike section 1.1 which were used to represent l_p norms.

4.1 Generating adversarially perturbed sample x' avoiding already explored regions.

As mentioned in the second properties of the human classification, we need a method that estimates whether a perturbed sample x' has de facto class or not to avoid using pseudo adversarial examples in adversarial training. To do that, we train a discriminator that is trained to distinguish clean samples and adversarially perturbed samples. Even if its classification is incomplete because of the overlapping samples, this discriminator allows us to avoid using pseudo adversarial examples for adversarial training. Note that this discriminator has a similar role with the discriminator in Generative Adversarial Nets [27] in that its gradients will be used to generate adversarial examples.

In our training method, we will use different magnitudes of perturbations $d_1 = 0 < d_2 < \cdots < d_T$. Then, the discriminator will assign corresponding classes for each magnitude. As we need to estimate previously allowed perturbation region $\mathcal{X}_\epsilon$, we provides two different inputs for each class: adversarially perturbed samples $x^* = \underset{x' \colon \|\triangle x\| = \epsilon}{\operatorname{argmax}} L(\theta, x', c_x)$ and their opponents $x^{**} = \underset{x' \colon \|\triangle x\| = \epsilon}{\operatorname{argmin}} L(\theta, x', c_x)$. When we have a discriminator, we will use lexicographical optimization [2] (that will be mentioned in 4.2) to prioritize by avoid generating samples in the previously allowed perturbation region using the discriminator, i.e. to make $x^* \in \mathcal{X}_\epsilon^C$, and to make the perturbed samples adversarial.

4.2 Using Gradient Episodic Memory (GEM) [4] for lexicographical optimization [2] of neural networks

Gradient Episodic Memory (GEM) [4] was originally developed to prevent catastrophic forgetting [5] which indicates the situation when a network was trained on some tasks, and trained on a new task after finishing the train on the previous tasks, then the network performs poorly (forget to perform well) on the previous tasks. Gradient Episodic Memory (GEM) is a method that enables to minimize the loss for task t without increasing losses for all previous task $k < t$ locally. It is based on first-order approximation of the loss and angles between different loss gradients.

To our best knowledge, the lexicographical optimization [2] of neural networks was only used to avoid catastrophic forgetting in continual learning[6] [4]. However, we argue that lexicographical optimization of neural networks is not only needed for traditional multi-task learning (MTL), but also for single task learning (STL). Single task learning problems can be described as learning tasks that have only one target loss. However, we often add regularization terms in the training loss in order to prevent over-fitting. As reducing the main loss (target loss) is more important than reducing the regularization terms, we can use lexicographical optimization by prioritizing the main loss. Progressively growing generative adversarial networks [28] uses images with different complexity. We can also think about using lexicographical optimization in their model so that the discriminator and the generator make sure to correctly learn simple structures first. As Lexicographical Genuine Robustness (LGR) also considers multiple accuracies with preference, it can also be considered as a problem that requires lexicographical optimization.

To better understand GEM [4], let us assume that there are losses $l_1(\theta), \cdots, l_T(\theta)$ with lexicographical preference, in other words, we want to reduce $l_t(\theta)$ without increasing $l_1(\theta), \cdots, l_{t-1}(\theta)$ for $t \in \{1, \cdots, T\}$. We also have (pre-projection) parameter updates $g_1, \cdots, g_T$ where $g_t = -\epsilon \nabla l_t(\theta)$ and ϵ is a learning rate.

We locally satisfy lexicographical improvement when $\langle g_t, g_k \rangle \geq 0, \forall k < t$ [4]. If it is not satisfied, we can project g_t to a nearest $\tilde{g}_t$ such that it satisfies $\langle \tilde{g}_t, g_k \rangle \geq 0, \forall k < t$, i.e. $\tilde{g}_t = \underset{\langle \tilde{g}, g_k \rangle \geq 0, \forall k < t}{\operatorname{argmin}} \|g_t - \tilde{g}\|_2$. As this problem is a quadratic program (QP) problem, they suggested solving this by its dual problem and recovering $\tilde{g}_t$ when $t \ll p$ (where p is the number of parameters in the neural network).

4.3 Combing gradient updates with lexicographical preferences: Onestep method

Unlike continual learning that only reduces loss for current task without forgetting previous tasks [4], we will reduce multiple losses simultaneously with lexicographical preferences. For each lexicographical training step, we can apply weights update for task 1 to task T. But, it requires to calculate $\tilde{g}_t$ for each task t and require much computational complexity. In stead of applying several small steps for different tasks, we suggest to apply only one combined weights update for each lexicographical training step. We will call this approach as "Onestep method".

When we have suggested parameter updates $\tilde{g_1}, \cdots, \tilde{g_T}$, let's consider their weighted mean $\tilde{g}_{\text{Onestep}} = \sum_{t=1}^{T} \alpha_t \tilde{g}_t$ where $\alpha_1, \cdots, \alpha_T \geq 0$ and $\sum_{t=1}^{T} \alpha_t = 1$. As $\langle \tilde{g}_t, g_1 \rangle \geq 0, \forall t$, we have $\langle \tilde{g}_{\text{Onestep}}, g_1 \rangle = \left\langle \sum_{t=1}^{T} \alpha_t \tilde{g}_t, g_1 \right\rangle = \sum_{t=1}^{T} \alpha_t \langle \tilde{g}_t, g_1 \rangle \geq 0$. Similarly, as $\langle \tilde{g}_t, g_s \rangle \geq 0, \forall t \geq s$, we have $\left\langle \sum_{t=s}^{T} \alpha_t \tilde{g}_t, g_s \right\rangle = \sum_{t=s}^{T} \alpha_t \langle \tilde{g}_t, g_s \rangle \geq 0$. It means that we can have same lexicographical training effect by simply applying the combined weights update $\tilde{g}_{\text{Onestep}}$.

[6] continual learning: It is a learning process to train a model for a sequence of tasks.

5 Related works

Considering adversarial robustness for different perturbation itself is not new. As standard accuracy can be considered as adversarial accuracy with zero perturbation, measuring standard accuracy and adversarial accuracy can be regarded as an example. Recently, a research [29] considered the model's robustness against multiple perturbation types and suggested adversarial training schemes for multiple perturbation types. However, their adversarial training methods ("Max" and "Avg" strategies) did not consider the different importance of adversarial accuracy with different magnitudes.

Similar concepts with pseudo adversarial examples and problems of using them for adversarial training have been studied. The concept of pseudo adversarial example is similar to the concept called "invariance-based adversarial example" [18] whose predicted class by f is the same with the original class c_x even if the predicted class by an oracle classifier o is changed. However, their definition requires an oracle classifier o which is hard to be defined while our definition requires predefined class for samples in clean data set $\mathcal{X}$, and thus it is easier to do theoretical analysis. Invalid adversarial example [30] is also similar to pseudo adversarial example. Their definition assumes data distribution of each class should be a manifold which limits the behavior of data distribution while we don't set manifold requirement in order to consider every possible situation.

There were some attempts to balance the accuracy of clean data and accuracy on perturbed samples of classifiers. MixTrain [12] uses adversarial training by dynamically adjusting the hyperparameter α for adversarial training. TRADES (TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization) method [31] tries to minimize the difference between predicted results of adversarially perturbed samples and predicted results clean samples instead of minimizing the difference between predicted results of adversarially perturbed samples and clean labels. To our best knowledge, there were no attempts to understand the different importance of adversarial accuracies of different magnitudes and prioritized training methods for adversarial robustness. We also handle the problem of simply increasing standard adversarial robustness, i.e. simply finding classifiers who are lexicographically more robust (LSR) than others.

6 Experiment

In order to compare the different training methods, we experimented with 5 different training methods: standard (non-adversarial) training, standard adversarial training [3], TRADES [31], OLSRC and OLGRC. OLSRC refers to the model that trained by applying Onestep method in subsection 4.3 without applying the adversary generation method that avoids generating samples in the previously allowed perturbation region in subsection 4.1. OLGRC refers to the model that trained by applying Onestep method in subsection 4.3 with adversary generation method that avoid generating samples in the previously allowed perturbation region in subsection 4.1.

We used PGD method [10] (using exact perturbation norms) to generate adversarially perturbed samples. We used ADAM algorithm [32] to train the discriminator for OLGRC.

We found that using mini-batch training for lexicographical optimization might not work well and lexicographical optimization [2] would require full batch training. As lexicographical optimization uses different weights update from a standard method (which is using more than one objective function), simply using mini-batch gradients update can result in catastrophic forgetting in other mini-batches. In other words, even if a weights update with lexicographical optimization can improve losses for current mini-batch satisfying the lexicographical improvement, as losses functions on different mini-batch will be different from current mini-batch, the current weight update can increase losses in different mini-batch. In order to avoid this problem, we applied full batch training in all experiments.

We did not plot for the changes of genuine adversarial accuracy in our experiments. We think it is unnecessary to plot them for toy example 2. It is impossible to plot the changes of genuine adversarial accuracy for the MNIST experiment as we don't know the actual data distribution and don't have predefined classes for all data. (Note that even if we can use discriminators to estimate them, the discriminators depends on the trained classifiers and estimated changes of genuine adversarial accuracy may not be comparable.)

6.1 Toy example in section 2

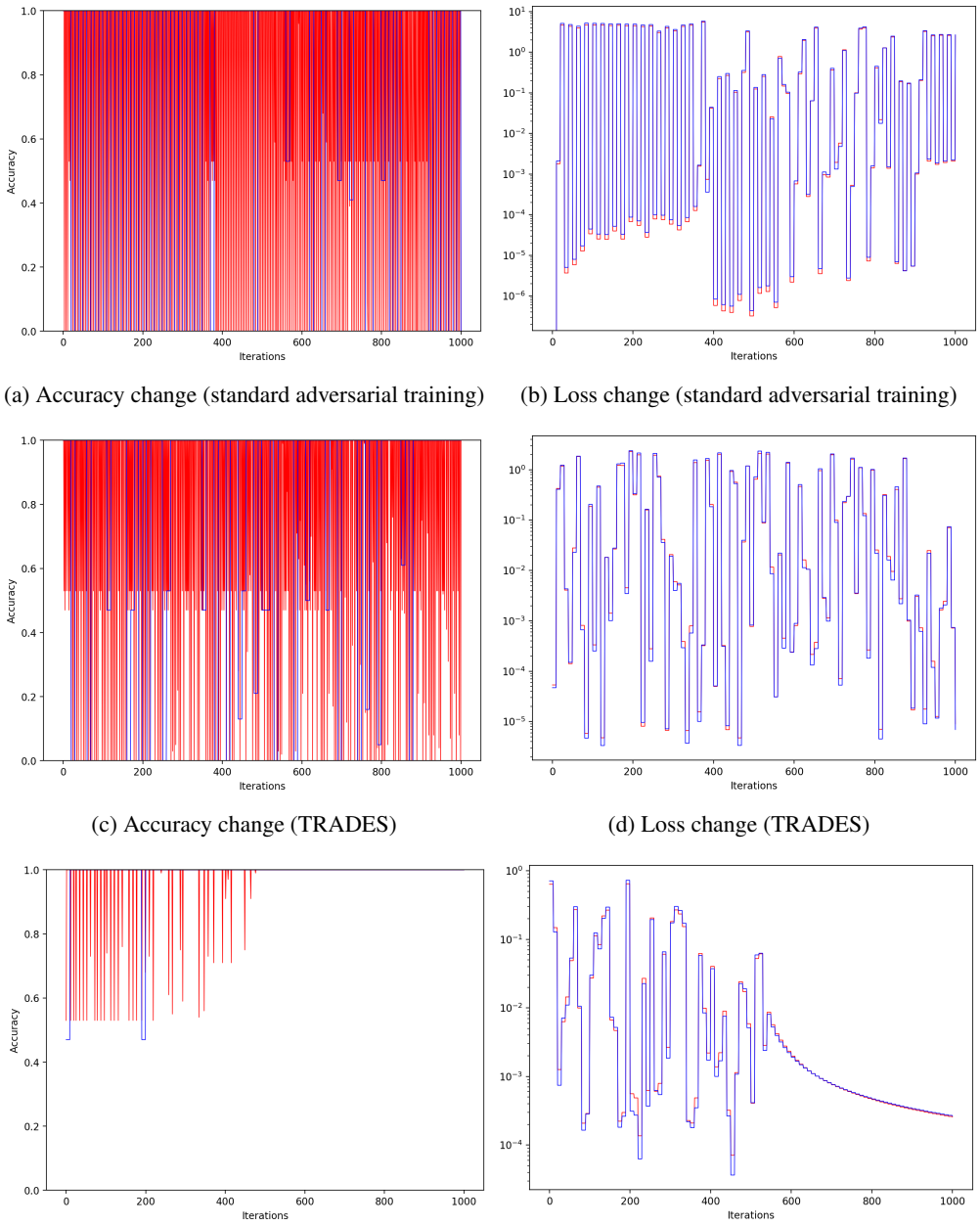
We randomly generated 100 training and 100 test samples from the toy example.
Fully connected neural network with one hidden layer (with 256 hidden neurons and leaky-ReLU non-linearity with parameter 0.2) was used for experiments. Full batch training was used with learning rate of 0.015 for 1000 epochs (iterations). Gradient descent algorithm was used for weights update.

6.1.1 The first experiment: examining the effect of lexicographical optimization [2]

In order to see the effect of lexicographical optimization [2] in adversarial training, in this experiment, we only used perturbation norm 4 for adversarial attacks. We used $\alpha = 0.5$ for standard adversarial training [3]. In order to apply comparable effects on the training of OLSRC, we used $\alpha_1, \alpha_2 = 0.5$ for weights of Onestep method and 10^{-10} was used for numerical stability in GEM algorithm [4]. We used $\frac{1}{\lambda} = 1.0$ for TRADES [31] training. Standard (non-adversarial) training and OLGRC were not experimented.

Comparing the change of accuracies and losses by iterations in Figure 9, we can observe that training processes of standard adversarial training [3] and TRADES [31] are not stable and classifiers can not be trained properly as both methods don't have prioritization of losses. (It seems TRADES method is less fluctuating than standard adversarial training, but it could be because of different effect of loss function.) On the other hand, training of OLSRC is much more stable as it prioritizes natural cross-entropy loss.

Comparing the plots for change of adversarial accuracy in Figure 10, the final classifier obtained by standard adversarial training [3] achieved 0 for both natural accuracy and adversarial accuracy (perturbation norm: 4). The final classifier obtained by TRADES [31] training achieved 1.0 natural accuracy and almost 0 adversarial accuracy (perturbation norm: 4). However, it might achieve 1.0 natural accuracy by chance considering the fluctuating training accuracy. Final OLSRC achieved 1.0 natural accuracy and about 0.5 adversarial accuracy (perturbation norm: 4).

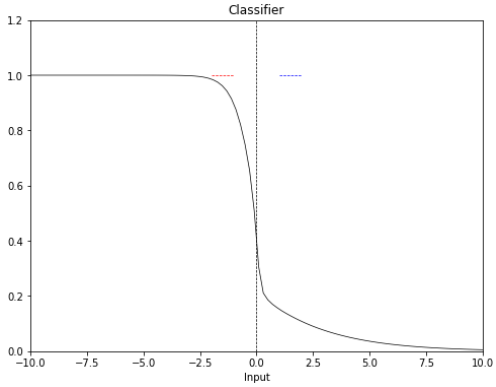


(a) Accuracy change (standard adversarial training) (b) Loss change (standard adversarial training)

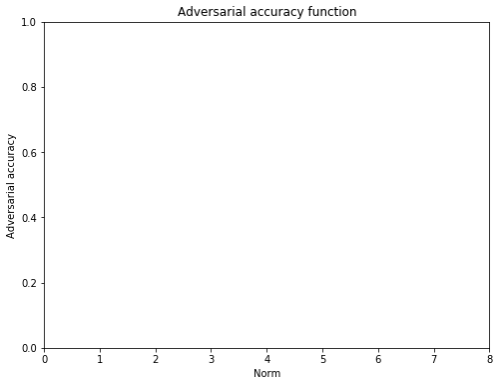
(c) Accuracy change (TRADES) (d) Loss change (TRADES)

(e) Accuracy change (OLSRC) (f) Loss change (OLSRC)

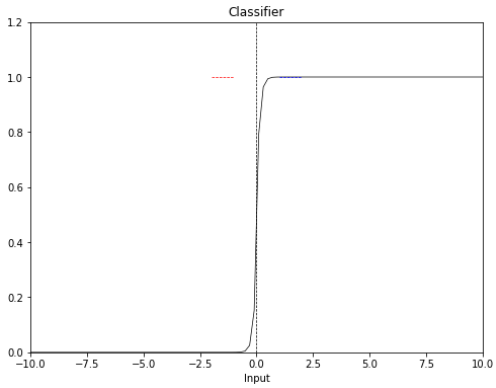
Figure 9: Change of accuracies and losses by iterations for the first experiment. Red: training data, Blue: test data.



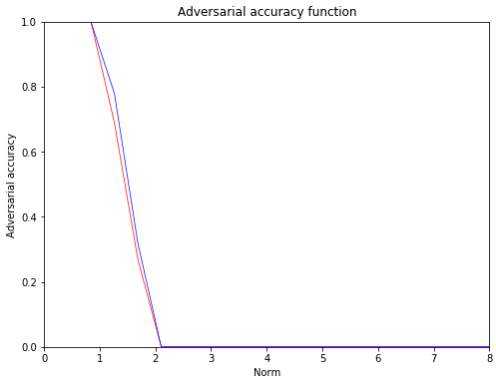
(a) Final classifier (standard adversarial training)



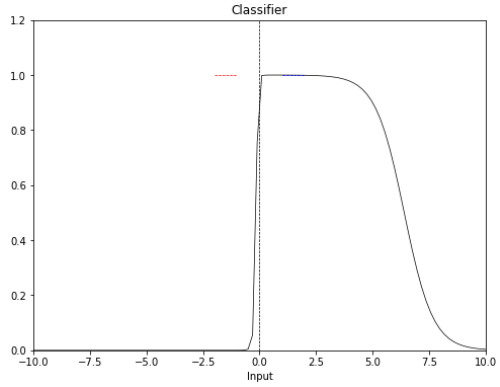
(b) Change of adversarial accuracy (standard adversarial training)



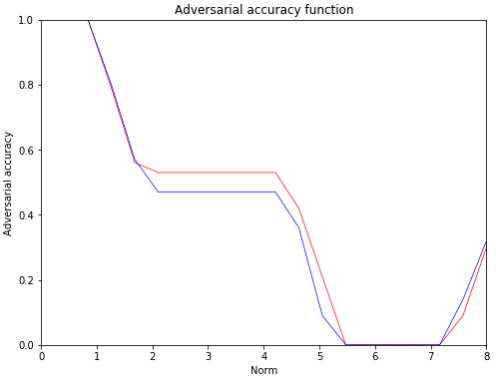
(c) Final classifier (TRADES)



(d) Change of adversarial accuracy (TRADES)



(e) Final classifier (OLSRC)



(f) Change of adversarial accuracy (OLSRC)

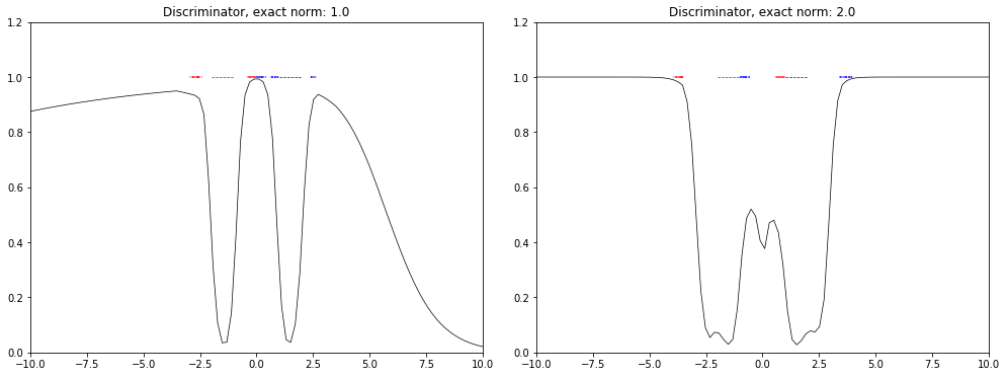
Figure 10: Final classifiers (estimated $p(c = 1|x')$) trained by different training methods for the first experiment. Red: $[-2, -1)$ and blue: $[1, 2)$ dashed lines in the classifier plots represent the regions for class -1 and class 1. Red: training data, Blue: test data (only for change of adversarial accuracy plots).

6.1.2 The second experiment: examining the effect of avoiding already explored regions

In order to see the effect of avoiding already explored regions, in the second experiment, we used exact perturbation norms 1, 2, 3, 4, 5, 6 for adversarial attacks. Only OLSRC and OLGRC were experimented. We used $\alpha_1 = 0.5, \alpha_2 = \cdots = \alpha_7 = \frac{1}{12}$ for weights of Onestep method and 10^{-3} was used for numerical stability in GEM algorithm [4].

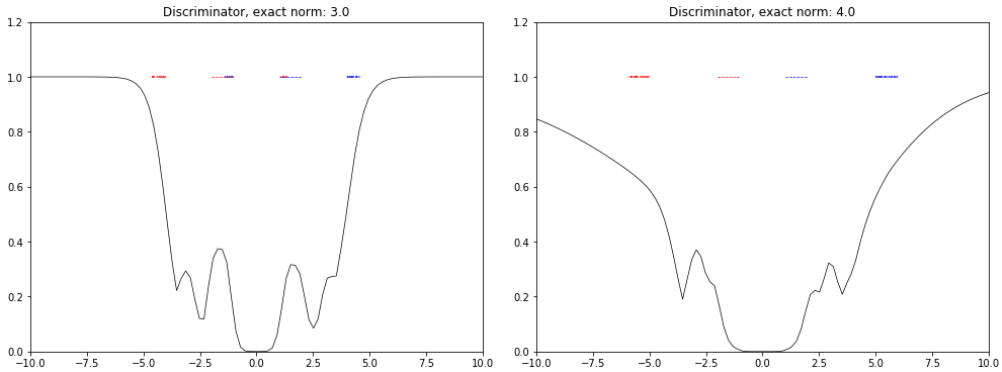
When the generated adversarially perturbed samples using discriminator were observed, we can check that the perturbation process avoids already explored regions even though it is incomplete. For example, when $\epsilon = 4, 5$, perturbed samples went to the right direction without making any mistake (Recall that previously allowed perturbation region $\mathcal{X}_\epsilon$ is $(-2-\epsilon, 2+\epsilon)$ when $\epsilon > 1$, and in order to avoid already explored points, perturbed samples need to move outward.). Estimated $p(x' \in \mathcal{X}_\epsilon^C | x')$ also roughly capture the regions that need to be explored.

Comparing the final classifiers and changes of adversarial accuracy in Figure 12, we can observe the shape of the trained OLGRC and its changes of adversarial accuracy are quite similar to f_3 in section 2 which is the theoretical OLGRC. Notice that it was not achievable when we only used training method for OLSRC as shown in the figure.



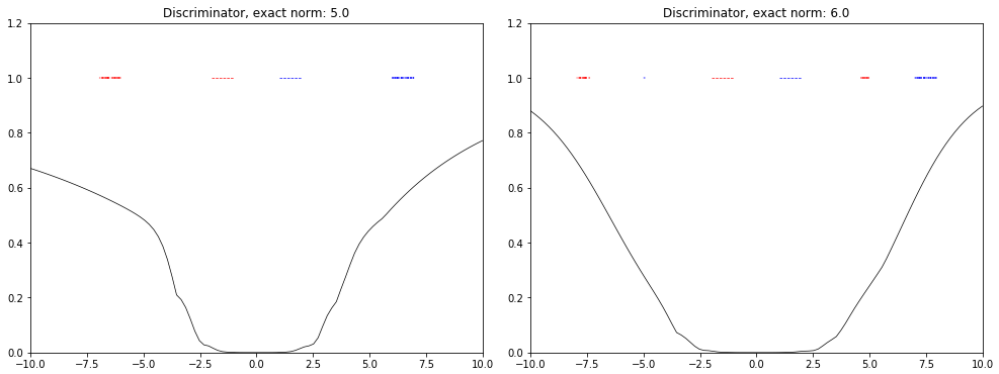
(a) $\epsilon = 1$

(b) $\epsilon = 2$



(c) $\epsilon = 3$

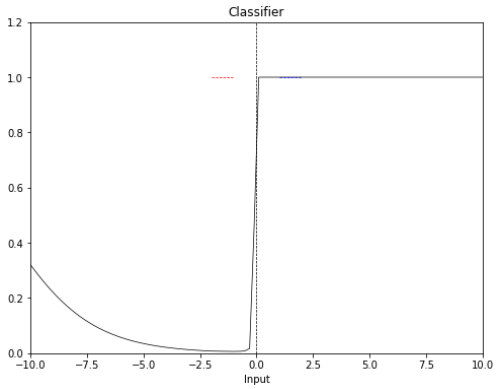
(d) $\epsilon = 4$



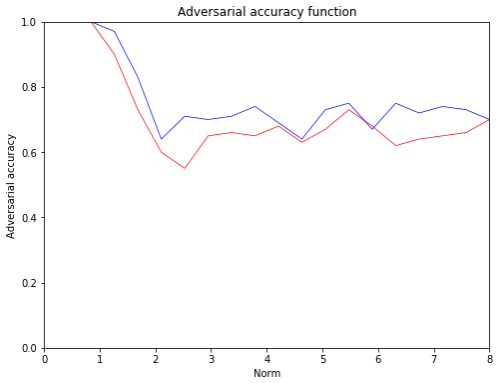
(e) $\epsilon = 5$

(f) $\epsilon = 6$

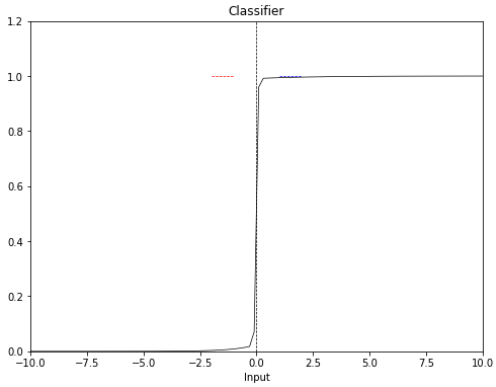
Figure 11: Plotted graphs show estimated probabilities that the input is not in the previously allowed perturbation region, i.e. estimated $p(x' \in \mathcal{X}_\epsilon^C | x')$. Red: $[-2, -1)$ and blue: $[1, 2)$ dashed lines represent the regions for class -1 and class 1. Generated adversarially perturbed samples using discriminator were color plotted class -1: red and class 1: blue.



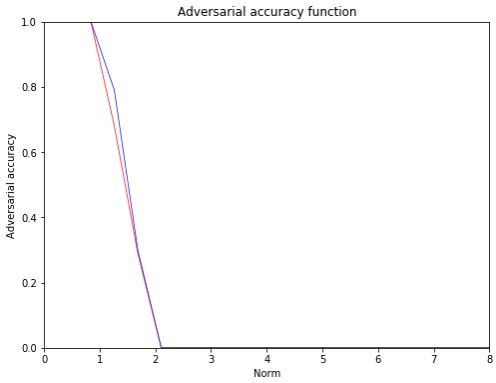
(a) Final classifier (OLSRC)



(b) Change of adversarial accuracy (OLSRC)



(c) Final classifier (OLGRC)



(d) Change of adversarial accuracy (OLGRC)

Figure 12: Final classifiers (estimated $p(c = 1|x')$) trained by different training methods for the second experiment. Red: $[-2, -1)$ and blue: $[1, 2)$ dashed lines in the classifier plots represent the regions for class -1 and class 1. Red: training data, Blue: test data (only for change of adversarial accuracy plots).

6.2 Experiment on MNIST data [20]

In order to prevent catastrophic forgetting in different mini-batches in mini-batch training, we only used randomly sampled 2000 samples as training data and full batch training was used with learning rate 0.001 for 2000 epochs (iterations). Note that our results will not be comparable with other previous analysis on MNIST data because we are using smaller training data.

For this experiment, we used common architecture with two convolution layers and two fully connected layers which can be found at `https://github.com/MadryLab/mnist_challenge`. In order to speed up the training, we applied the ADAM algorithm [32] after projections were applied because of the easiness of implementation. However, as applying adaptive gradient optimization after projections might violate lexicographical improvements, we speculate that it would be better to apply projection after adaptive gradient optimization method was applied.

We used the Projected Gradient Descent method [10] with 40 iterations to generates adversarial attacks. Only l_2 norm 4 adversarial attack is used for adversarial training for l_2 norm robust model and only l_∞ norm 0.3 adversarial attack is used for adversarial training for l_∞ norm robust model.

We used $\alpha = 0.5$ for standard adversarial training [3]. In order to apply comparable effects on the training of OLSRC and OLGRC, we used $\alpha_1, \alpha_2 = 0.5$ for weights of Onestep method. We used $\frac{1}{\lambda} = 1.0$ for TRADES [31] training. Note that due to different formulation of losses training results of TRADES will not be directly comparable with other training methods.

When we compare the results of different training methods (shown in Table 1), we can notice that using OLSRC and OLGRC are better than standard adversarial training [3] in terms of natural and adversarial accuracy. Unlike theoretical

Training methods	Standard (non-adversarial) training	Standard adversarial training	TRADES	Our (OLSRC)	Our (OLGRC)
Natural accuracy	96.930	98.120	**98.240**	98.160	98.200
Adversarial accuracy (l_∞ norm: 0.3)	0.0000	41.350	36.250	**43.260**	42.850
Natural cross entropy	8.7390×10^{-5}	3.3587×10^{-5}	3.2148×10^{-5}	3.2317×10^{-5}	$\mathbf{3.0954 \times 10^{-5}}$
Adversarial cross entropy (l_∞ norm: 0.3)	3.2721×10^{-2}	7.9387×10^{-4}	1.0360×10^{-3}	$\mathbf{7.8460 \times 10^{-4}}$	8.0536×10^{-4}

Table 1: Results on test data when l_∞ norm attacks were used for training and test

Training methods	Standard (non-adversarial) training	Standard adversarial training	TRADES	Our (OLSRC)	Our (OLGRC)
Natural accuracy	96.930	97.550	**98.020**	97.800	97.930
Adversarial accuracy (l_2 norm: 4.0)	5.2300	25.420	25.750	26.540	**26.710**
Natural cross entropy	8.7390×10^{-5}	3.8494×10^{-5}	$\mathbf{3.6156 \times 10^{-5}}$	3.8684×10^{-5}	3.7624×10^{-5}
Adversarial cross entropy (l_2 norm: 4.0)	2.5849×10^{-2}	9.9746×10^{-4}	1.0993×10^{-3}	$\mathbf{9.7528 \times 10^{-4}}$	1.0579×10^{-3}

Table 2: Results on test data when l_2 norm attacks were used for training and test

expectation, trained OLSRC was not lexicographically more robust than trained OLGRC (even on the trained data). It could be the result of simultaneously reducing more than one loss and applying the ADAM [32] after projections were applied. (When it comes to natural accuracy for both experiments, TRADES [31] achieved the best result. It could be because of different formulation of losses. It also achieved the smallest training loss in both experiments among adversarially trained models. Results on training data were not shown.)

7 Conclusion

In this work, we explained why existing adversarial training methods cannot train a classifier that has human-like robustness. We identified three properties of human-like classification: (1) human-like classification should be robust against varying magnitudes of adversarially perturbed samples and not just on a fixed maximum norm perturbations, (2) when we consider robustness on increasing magnitudes of adversarial perturbations, a human-like classifier should avoid considering already considered points multiple times, and (3) human-like classification need to prioritize the robustness against adversarially perturbed samples with smaller perturbation norm.

The suggested properties explain why previous methods for adversarial training and evaluation can be incomplete. For example, the second property explains why commonly used evaluation of adversarial robustness may not fully reveal our intuitive understanding of human-like robustness as standard adversarial accuracies don't avoid pseudo adversarial examples.

We defined a candidate oracle classifier called Optimal Lexicographically Genuinely Robust Classifier (OLGRC). OLGRC is (almost everywhere) uniquely determined when dataset and norm were given.

In order to train a OLGRC, we suggested a method to generate adversarially perturbed samples using a discriminator. We proposed to use Gradient Episodic Memory (GEM) [4] for lexicographical optimization [2] and an

approach to applying GEM when simultaneously reducing multiple losses with lexicographical preferences.

From the first experiment on the toy example from section 2, we showed that lexicographical optimization enables stable training even when other adversarial training methods failed to do so. The second experiment on the same toy example showed that we can use discriminator to roughly generate adversarially perturbed samples by avoiding already explored regions. Because of that, we could train a classifier that is similar to the theoretical OLGRC.

From the experiment on the MNIST data, we showed that our methods (OLSRC and OLGRC) achieved better performances on natural accuracy and adversarial accuracy than using standard adversarial training method [3].

8 Future work

In our work, we applied GEM [4] method to adversarial training which is not traditionally a multi-task learning (MTL) problem. This perspective also leads us to use multiobjective optimization [33] (without lexicographical preference) to the problems those were not considered as such. For example, one can use multiobjective optimization to train a single ensemble model that reduces losses in different datasets instead of training different models separately and averaging them. Multiobjective optimization can be used to find an efficient black-box attack by finding adversarial examples that can fool a list of models. By replacing the calculation of an average, it can also be used to smoothen the interpretations of a model [34].

Gradient episodic memory (GEM) [4] with standard gradient descent optimization method is slow and it needs to be combined with adaptive gradient update algorithms. One needs to try applying adaptive gradient update algorithms before the projection was applied. Also, GEM with mini-batch training cannot prevent not increasing losses in the other mini-batches. It is a serious limitation in deep learning applications. Future work needs to find a way to handle this problem.

To simplify the problem of finding a human-like classifier, we assumed the exclusive belonging which is unrealistic in many problems. We need analysis when this assumption is violated. We might need to consider easing the lexicographical preference as we expect to get the accuracy that is less than 1 when the exclusive belonging assumption is violated. Another approach would be estimating the hypothetical original data which satisfies the exclusive belonging assumption. In that approach, we consider current data are obtained by adding some input or label noises to the unknown original data.

Our training method will find a classifier that is robust against one form (l_1, l_2 or l_∞) of adversarial attacks with different magnitudes. However, we need to find a classifier that is robust against many forms of adversarial attacks (including shift, rotation [35], spatial transformation [36], etc.) with different magnitudes as attackers can try different kinds of attacks to exploit the classifier. Our model suggest a (almost everywhere) unique classifier that is robust against one form of adversarial robustness with some conditions. Because of this, in order to find a classifier that is robust against many forms of adversaries, we need to define a combined metric (or its generalization).

References

[1] Gamaleldin Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In *Advances in Neural Information Processing Systems*, pages 3910–3920, 2018.

[2] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[3] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[4] David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[5] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[8] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv:1607.02533 [cs, stat]*, July 2016. arXiv: 1607.02533.

[9] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks Against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 506–519, New York, NY, USA, 2017. ACM.

[10] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv:1706.06083 [cs, stat]*, June 2017. arXiv: 1706.06083.

[11] Tianhang Zheng, Changyou Chen, and Kui Ren. Distributionally Adversarial Attack. *arXiv:1808.05537 [cs, stat]*, August 2018. arXiv: 1808.05537.

[12] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. MixTrain: Scalable Training of Verifiably Robust Neural Networks. *arXiv:1811.02625 [cs, stat]*, November 2018. arXiv: 1811.02625.

[13] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There Is No Free Lunch In Adversarial Robustness (But There Are Unexpected Benefits). *arXiv:1805.12152 [cs, stat]*, May 2018. arXiv: 1805.12152.

[14] Elvis Dohmatob. Limitations of adversarial robustness: strong No Free Lunch Theorem. *arXiv:1810.04065 [cs, stat]*, October 2018. arXiv: 1810.04065.

[15] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial Spheres. *arXiv:1801.02774 [cs]*, January 2018. arXiv: 1801.02774.

[16] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically Principled Trade-off between Robustness and Accuracy. *arXiv:1901.08573 [cs, stat]*, January 2019. arXiv: 1901.08573.

[17] Shibani Santurkar, Dimitris Tsipras, Brandon Tran, Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Computer vision with a single (robust) classifier. *arXiv preprint arXiv:1906.09453*, 2019.

[18] Jörn-Henrik Jacobsen, Jens Behrmann, Nicholas Carlini, Florian Tramèr, and Nicolas Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness. *ArXiv*, abs/1903.10484, 2019.

[19] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Camille Salinesi, Moira C. Norrie, and Óscar Pastor, editors, *Advanced Information Systems Engineering*, volume 7908, pages 387–402. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[20] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[21] Ian Goodfellow. Defense against the dark arts: An overview of adversarial example security research and future research directions. *arXiv preprint arXiv:1806.04169*, 2018.

[22] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on mnist. *arXiv preprint arXiv:1805.09190*, 2018.

[23] Andrew Slavin Ross and Finale Doshi-Velez. Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients. *arXiv:1711.09404 [cs]*, November 2017. arXiv: 1711.09404.

[24] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. *arXiv:1805.09190 [cs]*, May 2018. arXiv: 1805.09190.

[25] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S. Davis, and Tom Goldstein. Universal Adversarial Training. *arXiv:1811.11304 [cs]*, November 2018. arXiv: 1811.11304.

[26] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[28] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[29] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. *arXiv preprint arXiv:1904.13000*, 2019.

[30] David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

[31] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018.

[34] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[35] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811, 2019.

[36] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

[37] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[38] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv preprint arXiv:1511.04599*, 2015.

[39] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. *arXiv preprint arXiv:1907.02044*, 2019.

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

[41] Jianbo Chen and Michael I Jordan. Boundary attack++: Query-efficient decision-based adversarial attack. *arXiv preprint arXiv:1904.02144*, 2019.

9 Supplementary

9.1 About non-increasing Lexicographical Standard Robustness

One can think it is more make sense to have a classifier whose Lexicographical Standard Robustness doesn't increase as the magnitude increase (in addition to the condition that the natural accuracy of the classifier is 1). However, depending on the dataset, it can be impossible to have such property for all $\|\triangle x\| \in [0, \infty)$. Let's think about another toy example (see Figure 13). There are only two classes $\{-1, 1\}$, i.e. $\mathcal{Y} = \{-1, 1\}$, and $\mathcal{X} = [-2, 2)$. $c_x = -1$ when $x \in [-2, -1) \cup [0, 1)$ and $c_x = 1$ when $x \in [-1, 0) \cup [1, 2)$. $p(c = -1) = p(c = 1) = \frac{1}{2}$.

If a classifier $f(x) = \begin{cases} 1, & \text{if } x \in [-1, 0) \cup [1, 2), \\ -1, & \text{if } x \in [-2, -1) \cup [0, 1) \end{cases}$ for $x \in [-2, 2)$ (necessary condition to have natural accuracy 1) regardless of behavior for $x \notin [-2, 2)$, it's Lexicographical Standard Robustness will not increase for $\|\triangle x\| \leq 1$ and its adversarial accuracy for $\|\triangle x\| = 1$ is 0. If we want $f(x)$ to satisfy non-increasing Lexicographical Standard Robustness property also for $1 < \|\triangle x\| \leq 2$, adversarial accuracy for $1 < \|\triangle x\| \leq 2$ should be equal to 0 and it requires $f(x) = \begin{cases} 1, & \text{if } x \in [-4, -3) \cup [2, 3), \\ -1, & \text{if } x \in [-3, -2) \cup [3, 4) \end{cases}$ for $x \in [-4, -2) \cup [2, 4)$. However, as adversarial accuracy for $\|\triangle x\| = 3$ is $\frac{1}{2} > 0$, f no longer satisfy non-increasing Lexicographical Standard Robustness property.

Hence, we need to recognize that even if it may seems counter-intuitive, it could be impossible to have a classifier that satisfy non-increasing Lexicographical Standard Robustness property when the natural accuracy of the classifier is 1.

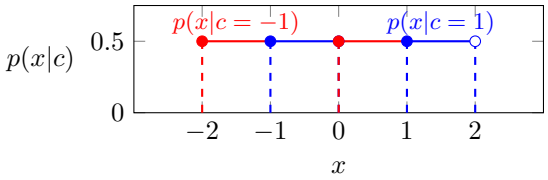


Figure 13: Plots of $p(x|c=-1)$: red and $p(x|c=1)$: blue for toy example.

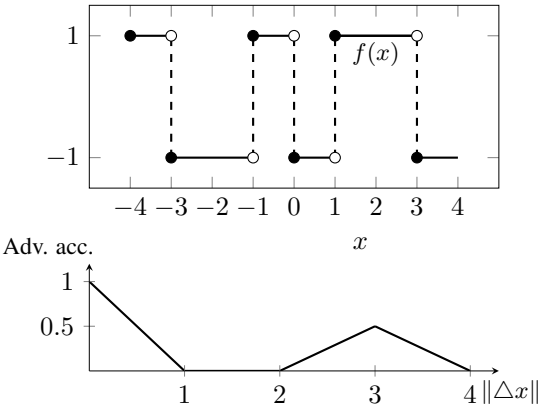


Figure 14: Above: Plot of a classifier $f(x)$ for the toy exmple in Figure 13, Below: Change of adversarial accuracy for $f(x)$ by exact perturbation norm $\|\triangle x\|$. Notice that f does not satisfy non-increasing Lexicographical Standard Robustness property for $2 < \|\triangle x\| < 3$.

9.2 Non-uniform prior

We can think of the same toy example in section 2 except for the prior probability. For example, $p(c=-1)=\frac{1}{5}, p(c=1)=\frac{4}{5}$. Then, it might be more reasonable to depending on a weighted version of ϵ when we define the change of adversarial accuracy functions, i.e. it will be depending on $\frac{\epsilon}{p(c=c_x)}$ rather than depending on ϵ.

9.3 Interpretation of a classifier: Negative Adversarial Remover (NAR)

Definition 8. *We define decision boundary (DB) for a classifier f and a class $c \in \mathcal{Y}$.*

- *Decision boundary :* $DB_c = \left\{x \in \mathbb{R}^d : \forall N(x), \exists x_1', x_2' \in N(x) \text{ such that } f(x_1') = c, f(x_2') \neq c\right\}$ *where $N(x)$ is a neighborhood of x.*

Note that when f is calculated from an accessible differentiable function g, i.e. $f(x) = \underset{c\in\mathcal{Y}}{\operatorname{argmax}}\, g(x)_c$, DB_c is not equivalent to $NB_c = \left\{x \in \mathbb{R}^d : \forall N(x), \exists x_1', x_2' \in N(x) \text{ such that } g(x_1')_c \geq p(C=c), g(x_2')_c < p(C=c)\right\} \bigcup \left\{x \in \mathbb{R}^d : g(x)_c = p(C=c)\right\}$ when prior is not uniform. NB_c will be called neutral boundary (NB).

Definition 9. *We define negative adversarial remover (NAR) and nearest decision boundary point (NDBP) for a classifier f, a sample $x \in \mathbb{R}^d$ and a class $c \in \mathcal{Y}$.*

- *Negative adversarial remover :* $NAR_c(x) = -\underset{\triangle x:\, x+\triangle x \in DB_c}{\operatorname{argmin}} \|\triangle x\|$
- *Nearest decision boundary point:* $NDBP_c(x) = \underset{x+\triangle x:\, x+\triangle x \in DB_c}{\operatorname{argmin}} \|\triangle x\|$

Note that NAR and NDBP for a sample x can be more than one points. One can check that $x = \text{NAR}_c(x) + \text{NDBP}_c(x)$. This indicates that when $f(x) = c$, $\text{NAR}_c(x)$ can be an interpretation of the sample x as it is the perturbation that change a point in the decision boundary, i.e. $\text{NDBP}_c(x)$, to sample x. $\text{NDBP}_c(x)$ is also similar to the concept called baseline in Integrated Gradients interpretation method [37] while $\text{NDBP}_c(x)$ is dependent on sample x unlike baseline which will be predefined by users. If f is calculated from an accessible differentiable function g, i.e.

$f(x) = \underset{c \in \mathcal{Y}}{\operatorname{argmax}}\, g(x)_c$, we can use DeepFool algorithm [38] or Fast Adaptive Boundary (FAB)-attack [39] to estimate $\text{NAR}_c(x)$ when $c = f(x)$. If we only have f, we can use Boundary attack [40] or HopSkipJumpAttack [41] to estimate $\text{NAR}_c(x)$ when $c = f(x)$.