University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master's Degree in Computer Science

Robustness Verification of k-Nearest Neighbor Classifiers by Abstract Interpretation

Master Thesis

Supervisor
Prof. Francesco Ranzato
University of Padova

Co-supervisor
Marco Zanella
University of Padova

Master Candidate
Nicolò Fassina

Student ID
2020579

Academic Year
2021-2022

Nicolò Fassina: *Robustness Verification of k-Nearest Neighbor Classifiers by Abstract Interpretation*, Master Thesis in Computer Science, ©December 2022.

"Computer science is no more about computers than astronomy is about telescopes."
— Edsger Dijkstra

Acknowledgments

This Master's thesis has been a great experience. It has given me a lot of insight into a topic that I am passionate about, and has taught me a lot about writing and researching. This last project made me realize that if I am willing to put in the effort, any objective is achievable.

I would like to express my gratitude to all the people who have helped me make this work possible. First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Francesco Ranzato, for his guidance throughout the entire process and to always been supportive. My sincere thanks also go to my co-supervisor, Marco Zanella, for his advice and continuous feedback that allowed me to carry out this project in the best possible way. It was an honor and privilege to have contributed to the writing of the paper "Robustness Verification of k-Nearest Neighbors by Abstract Interpretation". This incredible opportunity has helped me grow, both personally and professionally.

Last but not least, I would like to thank everyone who has been close to me along the way. A special thanks to my family and friends for their encouragement and invaluable patience, but mostly to have been there for me through thick and thin, especially during the final stages.

Words cannot fully express what I feel, but I know for sure that without the love and the emotional support of my family, friends and professors during my last years, I would not be where I am today.

Padova, December 2022 Nicolò Fassina

Abstract

Abstract interpretation is an established mathematical framework introduced by Cousot and Cousot in 1977 and ubiquitously used in static program analysis. In recent years, many noteworthy works have shown how abstract interpretation can be successfully applied to formally verify robustness properties of some major machine learning techniques like (deep) neural networks, decision trees and support vector machines.

This research work aims to pursue this line of research by proposing a novel abstract interpretation-based framework for designing a sound abstract version of the k-Nearest Neighbors (kNN) algorithm, a well-known non-parametric supervised learning method widely used for classification and regression tasks, which is then instantiated to the standard interval domain approximating the range of numerical features, to verify its robustness and stability properties. This verification approach has been fully implemented and evaluated on several datasets, including standard benchmark datasets for individual fairness verification, and then compared with some related works finding adversarial examples on kNNs. The experimental results turned out to be very promising and showed high percentages of provable robustness and stability in most of the reference datasets, thus making a step forward in the current state-of-the-art of formal verification of machine learning models.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AI	Artificial Intelligence
***k*NN**	*k*-Nearest Neighbors
***k*NAVe**	*k*NN Abstract Verifier
GeoAdEx	Geometric Adversarial Example
inf	Infimum
ML	Machine Learning
poset	Partially Ordered Set
QP	Quadratic Programming
RAF	Reduced Affine Form
sup	Supremum

1 Introduction

Adversarial machine learning [16, 43, 19] is an emerging research field that combines machine learning (ML) and computer security to design methodologies for making ML techniques safer and more robust to *adversarial attacks*. ML algorithms have become increasingly popular, and are recently being used in a variety of real-world applications including image classification, face recognition, autonomous driving, and many more. Despite their huge and unexpected success in performing various complex tasks, security has never been in the spotlight, at least until now.

Motivated by the fact that real-world applications need to be resilient to arbitrary input data, *adversarial examples*, inputs obtained by applying small perturbations to examples from the input space in order to change the decision of a classifier with high confidence, are now more than ever study and research topic. A ML classifier is in fact defined to be *robust* for a (typically very small) perturbation of its input samples, representing a possible adversarial attack, when it assigns the same correct class to all the samples within that perturbation, thus avoiding misclassifying examples that are only slightly different from correctly classified examples drawn from the data distribution. In many works, *robustness* is often mistakenly called *stability*, as if they were synonyms, but these two properties are actually quite different: while stability requires the classifier to maintain its (unique) decision within the perturbation, robustness goes further and imposes also a correctness constraint, that is, the decreed class must coincide with the one associated with the tested sample before any perturbation was applied.

Robustness, both to accident and malevolent agents, is clearly a crucial determinant of the desired quality of machine learning systems. Currently, there are several defense techniques to increase the robustness of ML models, ranging from adversarial training and input validation to testing and automatic verification. There is a large spectrum of possible negative impacts from poorly secured systems, particularly issues around sensitive applications such as medical and transportation systems, where a failure can lead to potentially irreversible and life-threatening consequences [2]. For example, an attacker may wish to cause a self-driving car to incorrectly recognize road signs, most likely causing a car crash, or cause a medical device to falsely reconstruct a body scan leading the doctor to overlook a particular disease or to think of a less serious or completely different one. Although machine learning models have become part of everyday life, many of them, even the most advanced, are still easily susceptible to the almost imperceptible perturbations of their inputs and therefore it becomes necessary to have efficient strategies that allow us to analyze and verify their robustness property in a precise and provable way.

In this research work, we focused on the formal verification problem for robustness and stability properties of the *k-nearest neighbors* algorithm [1, 14], also known in literature as kNN or k-NN. The idea behind this ML algorithm is very simple and consist in predicting the outcome for a test sample by inferring its k closest neighbors ranging in a given dataset, according to some distance metric, hence the name k-nearest neighbors. It is mainly used for *classification tasks*, namely, when we want to attribute a discrete label representing a certain category to an input sample, but can also be instantiated for *regression tasks*, which are more concerned with predicting a continuous value. Despite being one of the most straightforward supervised machine learning techniques, kNN is a well-liked and reliable predictive model with a wide range of applications [18]. In particular, kNNs are successfully applied in different fields where adversarial attacks must be taken into account, notably in chemistry since the early 1970s. For instance, 1NN was used to classify molecular structures using its nuclear magnetic resonance spectra [17], while the general version was exploited to classify sensor array data for two types of chemical warfare agents [34], substances whose toxic properties are meant to kill, injure or incapacitate human beings.

We propose a novel approach to automatically infer when a k-nearest neighbors classifier is *provably robust* and/or *stable* for an input sample in a test dataset with respect to

a given perturbation. Our methodology is based on the long-established framework of *abstract interpretation* [7, 8, 32] for designing correct and precise over-approximations of dynamic computations, which has already been successfully exploited for the formal verification of several different machine learning models [42]. Specifically, we leverage a *numerical abstract domain* to define computer-representable abstract versions of both the kNN classifier and the adversarial perturbation. The resulting over-approximate classifier turns out to be a *sound* version of the concrete one, i.e., its outcome is always a superset of the outcome of the latter, and has been designed to be fast and scalable in the size of the training set as well as in the value of k, to which no upper bound has been set. This abstract interpretation-based technique allows to obtain provable robustness and stability percentages of any dataset consisting of training and test sets. Our formal verification methodology has been instantiated to the well-known numerical abstract domain of intervals [8] and has been implemented in Python, leading programming language when it comes to machine learning, in a tool called kNAVe (kNN *A*bstract *Ve*rifier). We performed an exhaustive experimental evaluation of the final verifier on 7 datasets commonly used in formal robustness verification and on 4 standard datasets for *individual fairness* verification, achieving very promising results in most of these. Moreover, robustness results were compared with the output data of the GeoAdEx tool by Sitawarin et al. [38], which computes minimum-norm adversarial examples on kNNs classifiers by solving an optimization problem.

The rest of the thesis is organized as follows:

Chapter 2 describes the background needed to better understand the concepts covered in this research work, providing some basic notions about used notation, and on what concerns machine learning and abstract interpretation;

Chapter 3 reviews state-of-the-art formal methods currently applied to verify major machine learning models and highlights some recent works finding adversarial examples on kNNs;

Chapter 4 describes our formal verification method based on abstract interpretation, explaining in detail how it works and why it is sound;

Chapter 5 shows the experimental results obtained by executing our abstract verifier on the reference datasets, and compares the latter with those of GeoAdEx;

Chapter 6 sums up the contribution of this research work and proposes future improvements to possibly obtain even better results.

2
Background

This chapter will introduce and describe all the knowledge needed to fully understand the thesis topic. It begins by reporting some of the notations that will often recur in the various chapters, and then moves on to discuss the topics in two macro-sections: the first provides the reader with an overview of what machine learning is, what supervised classification means and how kNN classifiers work, ending up analyzing their safety and fairness issues, and the properties that must be consequently evaluated; while the second introduces basic concepts of the abstract interpretation framework, specially numerical abstract domains and the interval domain.

2.1 Notation

An input space of feature vectors $X \subseteq \mathbb{R}^n$ and a set of classification labels L are assumed. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $z \in \mathbb{R}$ and $i \in [1, n] \subset \mathbb{N}$ be, respectively, two vectors of n real numbers, a real number and a natural number between 1 and n. We denote with $\mathbf{x}_i \in \mathbb{R}$, $\mathbf{x} \cdot \mathbf{y} \triangleq \sum_i^n \mathbf{x}_i + \mathbf{y}_i \in \mathbb{R}$, $\mathbf{x} + \mathbf{y} \in \mathbb{R}^n$, $z\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_1 \triangleq \sum_1^n |\mathbf{x}_i| \in \mathbb{R}$, $\|\mathbf{x}\|_2 \triangleq \sqrt{\mathbf{x} \cdot \mathbf{x}} \in \mathbb{R}$, $\|\mathbf{x}\|_\infty \triangleq \max\{|\mathbf{x}_i| \mid i \in [1, n]\} \in \mathbb{R}$, respectively, i-th component, dot product, vector addition, scalar multiplication, ℓ_1 (i.e, Manhattan) norm, ℓ_2 (i.e., Euclidean) norm, and ℓ_∞ (i.e., maximum) norm. Moreover, if $h \colon X \to Y$ is any function, then $h^c \colon \wp(X) \to \wp(Y)$, defined by $h^c(S) \triangleq \bigcup_{x \in S} \{h(x)\}$, denotes the standard collecting lifting of h.

2.2 Machine Learning

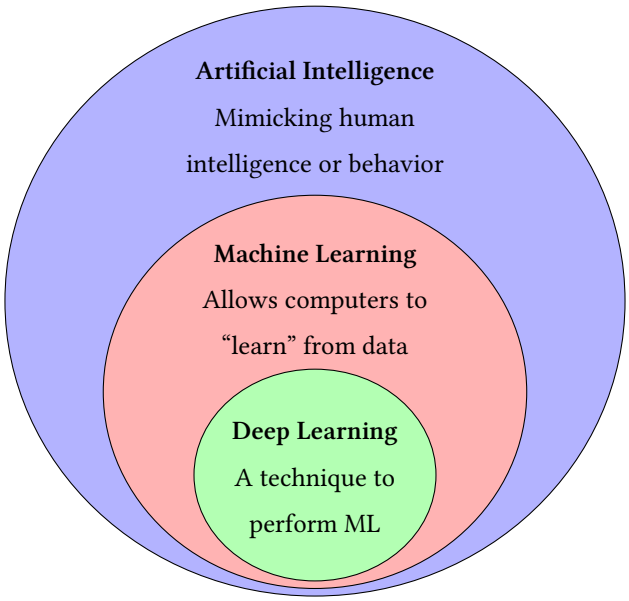

Figure 2.1: Venn diagram showing the relation between artificial intelligence, machine learning and deep learning.

Nowadays, we hear more and more about artificial intelligence (AI), machine learning and deep learning. These terms are frequently confused or used as synonyms, but they are actually quite different from one another.

Artificial intelligence is a burgeoning field with numerous practical applications and active research topics. According to its father John McCarthy [20], AI can be defined as:

> *"The science and engineering of making intelligent machines, especially intelligent computer programs."*

Its main goal is to develop intelligent software to automate mind- and time-consuming tasks. For instance, AI is suitable for solving problems that are intellectually challenging for humans, but at the same time easily definable in a formal mathematical manner that is understandable by computers. Instead, if we consider tasks that our minds can carry out effortlessly, such as, for example, recognizing object, faces or sounds, categorizing the subject of documents or identifying entities (places, titles, names, actions, etc.) in

a phrase or an image, it can be difficult to comprehend how to formalize the problem in a way that is understandable from a computer, since we learned how to do it by developing and gaining experience. The real challenge arises when it is not possible to precisely formalize the problem, when there is some level of uncertainty in the input or output, and when the solution is too complex or inefficient.

Machine learning is the branch of artificial intelligence that tackle those issues and explains squarely *how* to enable computer systems to learn and improve themselves automatically without being explicitly programmed or supervised by humans. A ML algorithm is capable of adapting to its environment while improving its performance on a specific task, which is typically defined in terms of how the system should process input data. Data are therefore crucial in the learning process, and are commonly represented as vectors in $\mathbb{R}^n$ where each vector entry is called *attribute* or *feature*. For example, the features of an image, represented as a vector, can be its pixels ranging from 0 to 255.

Data and tasks also determine which ML method is most suitable and performing for each specific case. Within machine learning, the following learning paradigms are identified:

- **supervised learning**: data are presented as a set of examples $\bigcup_{i=1}^{n}\{(\mathbf{x}_i, l_i)\}$ called training set, each of which is a pair comprising an input feature vector in $\mathbb{R}^n$ and its correct output, typically a literal. It is particularly suitable for classification and regression tasks: if the output is discrete a classification function is learnt; otherwise, if the output is continuous, a regression function is learnt. The learned function is then used to infer the output of unseen data.
- **unsupervised learning**: data are presented as a set of unlabelled vectors $\bigcup_{i=1}^{n}\{\mathbf{x}_i\}$. In this case, the outputs represent the data structure, which is determined by a cost function that must be minimized. It is particularly suitable for clustering and dimensionality reduction tasks.
- **reinforcement learning**: as for unsupervised learning, data are presented as a set of unlabelled vectors $\bigcup_{i=1}^{n}\{\mathbf{x}_i\}$. In contrast, however, a scalar reword signal is used to evaluate input-output pairs by trial and error in order to find the best outputs for each input. It is used in all fields where the system must respond to changes in the environment, therefore attributable to regression problems.

Machine learning can appear in many guises, and deep learning is one of them, but we will focus on it use for supervised classification tasks, precisely exploiting the k-nearest neighbors algorithm.

2.2.1 Supervised Classification

Classification is the process of assigning new input vectors to the class to which they are most likely to belong, using a classification model built from previously labeled training data. The goal is to learn a mapping from inputs $\mathbf{x} \in \mathbb{R}^n$ to outputs $l_{\mathbf{x}}$ representing some category. This problem can be formalized as approximating an ideal unknown function $f : X \to \wp(L)$ such that $f(\mathbf{x}) = l_{\mathbf{x}}$, by selecting a function $h : X \to \wp(L)$ in the hypothesis space containing all the possibilities, such that $h \approx f$ on all training data, to make predictions on previously unseen inputs in a test dataset. The number of possible outputs determines the type of supervised classification we need: when each instance of data can be assigned to one of two possible class labels is called *binary classification*; when classification involves more than two class labels, and each instance is assigned to only one class label, is called *multiclass classification*. In some cases, a single example can be classified as belonging to more than one class, which is referred to as *multilabel classification*. In this thesis, however, we will consider only the first two types, i.e., binary and multiclass classification.

2.2.2 kNN Classifiers

One of the simplest and trivial form of learning is the Rote learning for classification tasks [10], memorizing the entire training data and performing classification only if the attributes of the test sample exactly match the attributes of one of the training examples. There are two obvious drawbacks to using this approach: many test samples will not be classified because they do not match any of the training examples exactly, and when two or more training examples have identical attributes but different class labels it is not possible to infer which is the correct one. One possible way to mitigate these problems is to adopt a more refined approach based on similarity rather than strict equality.

The k-nearest neighbors algorithm is a *non-parametric* supervised learning method that exploits data proximity to make classifications or predictions about the grouping of an individual sample, assuming that similar data can be found nearby. Non-parametric means that no assumptions are made about the underlying data distribution, so that the model structure is uniquely determined by the dataset itself, which is extremely useful in practice, as most real-world datasets do not adhere to mathematical models. A kNN classifier assigns an unknown feature vector $\mathbf{x} \in \mathbb{R}^n$ into the class $l_{\mathbf{x}}$ where the majority of its k nearest neighbors belong, thus avoiding getting stuck when there is no identical

data. In its straight-forward application, i.e 1NN, such a classifier simply assigns a test sample the class of its nearest neighbor. The number $k \in \mathbb{N}$ of neighbors, as well as the distance function used to compare vectors in $\mathbb{R}^n$, are hyperparameters of this prediction model. As Sebastian Raschka pointed out [31], although technically *plurality voting* is used, the term *majority voting* is almost ubiquitous in literature. The difference between these two terms is that "majority voting" requires a majority of more than 50%, which only works when there are only two classes. When there are multiple classes, e.g. four categories, it is not always necessary a 50% of the vote to make a decision about a class, and it is in fact possible to assign a class label with a vote of more than 25%. As we will see in Chapter 4, this last consideration will be fundamental in discarding more labels.

The left picture in Figure 2.2 shows an example of classification performed with a kNN model instantiated at the case $k = 3$, over a dataset in $\mathbb{R}^2$ with three classes *red*, *green*, and *blue*. By applying the 3NN algorithm to every vector in the input space, the right picture represents the dataset after the classification has been completed. For an unknown input vector represented by a gray dot in the left picture, this algorithm therefore computes the 3 closest samples in the dataset, which are the ones within the dashed circle, and then the most common label among them is inferred. Following this strategy, it may happen that two or more labels receive the same number of votes, resulting in a tie. For instance, replacing a red point inside the dashed circle with a blue one would result in a tie because each label would receive exactly 1 vote.

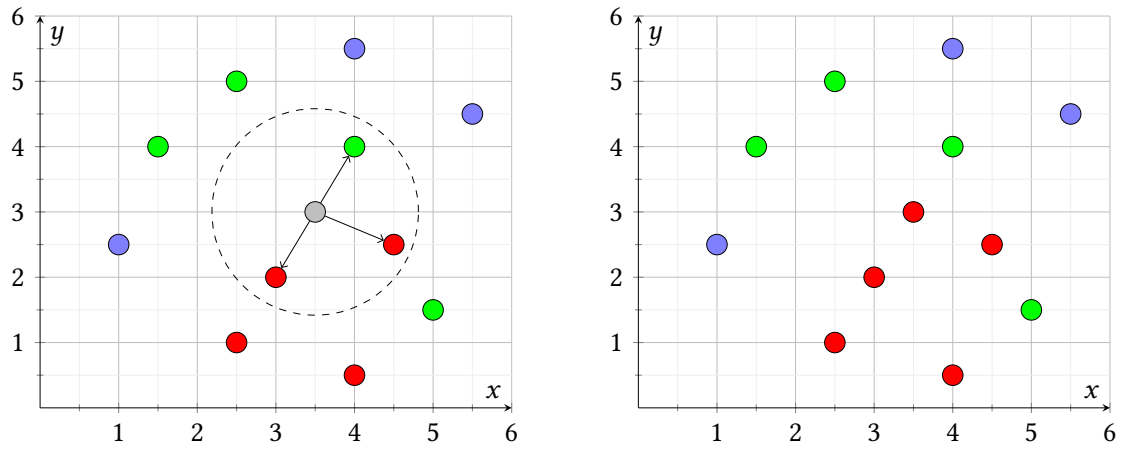

Figure 2.2: 3NN over a dataset with 3 classes, which shows how a new point is classified (left), as well as its label after being classified (right).

As can be seen from the example above, a kNN model does not need a learning phase, which is instead required in most supervised ML algorithms, because all the examples are stored and entirely used at classification time, a feature that makes kNN a so-called *lazy* (or *just-in-time*) learning algorithm [3]. While this makes it quite simple to implement, it can potentially results in a high computation time due to the effort of computing and sorting the distances, especially when so many neighbors must be found. For this reason, k is usually a low value, very often below 9 and in any case always smaller than the square root of the total number of samples in the dataset. Being lazy, on the other hand, makes it perform well in many situations. Under certain reasonable assumptions, a well-known result by Cover and Hart [9] shows that the classification error of the nearest neighbor rule is bounded above by twice the optimal Bayes error. Furthermore, the error of the generai kNN method approaches that of the Bayes error asymptotically and can be used to approximate it.

Algorithm 2.1 provides a high-level summary of the kNN algorithm for classification tasks. Given a ground truth dataset $D = \{(\mathbf{x}_1, l_1), \ldots, (\mathbf{x}_N, l_N)\} \subseteq X \times L$, a number of neighbors $k \in \mathbb{N} \setminus \{0\}$, and a distance function $\delta\colon X \times X \to \mathbb{R}_{\geq 0}$, a kNN classifier is modeled as a total function $C_{D,k,\delta}\colon X \to \wp(L)$, which maps an input sample $\mathbf{x} \in X \subseteq \mathbb{R}^n$ into a nonempty set of labels, by first selecting the k samples in D that are closest to $\mathbf{x}$ in terms of δ, and then computing the set of their most frequent labels. Since a tie vote means to output a set including more than one label, we consider sets of labels as co-domain of classifiers.

Algorithm 2.1 kNN algorithm — time complexity (for small k): $O(n \cdot |D|)$

$M, O \leftarrow \varnothing$
for all $(\mathbf{y}, l_\mathbf{y}) \in D$ ▷ Computation of distances from $\mathbf{x}$
    $d_\mathbf{y} \leftarrow \delta(\mathbf{x}, \mathbf{y})$
    O.Append$\big((d_\mathbf{y}, l_\mathbf{y})\big)$
end for
Heapify(O) ▷ Turns O into a heap structure in linear time
for all $i \in [1, k]$
    $M[i] \leftarrow O$.ExtractMin() ▷ Logarithmic time complexity
end for
return $\arg\max_{l \in L} \sum_{(\mathbf{y}, l_\mathbf{y}) \in M \text{ s.t. } l = l_\mathbf{y}} 1$ ▷ Labels with the highest number of votes

2.2.3 Adversarial Examples

Adversarial examples in machine learning are inputs to a model that are intentionally designed to cause the algorithm make mistakes in its predictions, yet appearing to be a legitimate inputs to a human. In this thesis, we will look at these kinds of inputs in the context of kNN classifiers. Given a classifier $C\colon X \to \wp(L)$, adversarial examples can be formally defined as inputs $\bar{\mathbf{x}}$ where the difference between $\bar{\mathbf{x}}$ and non-adversarial inputs $\mathbf{x}$ is minimal under some distance metric $\delta\colon X \times X \to \mathbb{R}_{\geq 0}$, but enough to make C infer a different output. To obtain such an $\bar{\mathbf{x}}$, a *perturbation* P is applied to $\mathbf{x}$ to cause a variation in the feature values of $\mathbf{x}$, defining a potential adversarial region $P(\mathbf{x}) \subseteq X$ in which $\bar{\mathbf{x}}$ belong. Generally, adversarial examples attempt to satisfy:

$$0 < \delta(\mathbf{x}, \bar{\mathbf{x}}) \leq \epsilon \text{ such that } C(\mathbf{x}) \neq C(\bar{\mathbf{x}})$$

where $\epsilon \in \mathbb{R}$ is a (small) constant bounding the magnitude of the perturbation.
Resisting adversarial perturbations is a fundamental requirement for a good ML model, and therefore the classifier is expected to keep its decision for minor feature variations. In some cases, this is taken for granted, but unfortunately, it is not always true. If the chosen distance function is too simple, kNN may significantly suffers when adversarial perturbations are applied to vectors having limited precision features. Digital images, for example, frequently use only 8 bits per pixel (with values from 0 to 255), discarding all information below $1/255$ of the dynamic range. Because of this, if we consider a value $\tau > 0$ strictly below such a threshold, it is unrealistic for the classifier to respond differently to input $\mathbf{x}$ than to input $\bar{\mathbf{x}} = \mathbf{x} + \mathbf{y}$ if every element of $\mathbf{y}$ is less then or equal to τ. However, if the distance between two vectors is computed, for example, by summing the difference feature by feature, as happens using the Manhattan metric, then $\delta(\mathbf{x}, \bar{\mathbf{x}}) = \delta(\mathbf{x}, \mathbf{x} + [n\tau, 0 \dots 0])$ holds, which is not what we want: due to limited precision, adding a value $n\tau > \tau$ to the first feature while leaving the others as they are, or adding τ to all the n features, have a completely different impact, but with this distance metric they appear to have the same weight.

When we will instantiate the generic perturbation to a specific one, we will look at the well-studied ℓ_∞-perturbation [5], which affect uniformly all features, avoiding the above problem: given an input vector $\mathbf{x} \in \mathbb{R}^n$ and a magnitude $\epsilon \geq 0$, the ℓ_∞-perturbation is defined by the adversarial region $P^\epsilon_\infty(\mathbf{x}) \triangleq \{\mathbf{w} \in \mathbb{R}^n \mid \max(|\mathbf{w}_1 - \mathbf{x}_1|, \dots, |\mathbf{w}_n - \mathbf{x}_n|) \leq \epsilon\}$, i.e., the ℓ_∞-ball of radius ϵ centered in $\mathbf{x}$.

2.2.4 Stability and Robustness

Classifiers are usually evaluated and compared through multiple metrics. A simple and intuitive metric is *accuracy* on a test set: given a ground truth test dataset $T = \{(\mathbf{x}_1, l_1), \ldots, (\mathbf{x}_N, l_N)\} \subseteq X \times L$, the accuracy of a classifier $C\colon X \to \wp(L)$ on T represents the proportion of samples with correct predictions out of the total number of samples, and is defined by the ratio:

$$\textsc{Accuracy}(C, T) \triangleq \frac{|\{(\mathbf{x}, l_\mathbf{x}) \in T \mid C(\mathbf{x}) = \{l_\mathbf{x}\}|}{|T|} \tag{2.1}$$

In adversarial scenarios, assessing a classification model solely on this standard metric is far from adequate. Although accuracy is useful in assessing the model's overall performance, it does not highlight any safety concerns. We now define two relevant properties in this context, which will be formally verified by our verification method.

Definition 2.1 (Stability). A classifier $C\colon X \to \wp(L)$ is *stable* on an input $\mathbf{x} \in X$ for a given perturbation $P\colon X \to \wp(X)$, denoted by $\textsc{Stable}(C, P, \mathbf{x})$, when $\bigcup_{\bar{\mathbf{x}} \in P(\mathbf{x})} C(\bar{\mathbf{x}}) = \{l\}$ holds for some $l \in L$.

Definition 2.2 (Robustness). A classifier $C\colon X \to \wp(L)$ is *robust* on an input $(\mathbf{x}, l_\mathbf{x}) \in X \times L$ for a given perturbation $P\colon X \to \wp(X)$, denoted by $\textsc{Robust}(C, P, \mathbf{x}, l_\mathbf{x})$, when $\forall \bar{\mathbf{x}} \in P(\mathbf{x}).\ C(\bar{\mathbf{x}}) = \{l_\mathbf{x}\}$ holds.

Stability means that a classifier does not change its output on a region of similar inputs, and it is orthogonal to accuracy in the sense that it does not require prior knowledge of the ground truth labels. Robustness, on the other hand, requires the classifier to be stable and correct, which means that it must output the same label that the input has in the dataset to which it belongs. It should be noted that for null ℓ_∞-perturbations, i.e. with $\epsilon = 0$, the definitions of accuracy and robustness coincide, leading us to conclude that the latter property expresses accuracy in adversarial scenarios.
As we did in (2.1), we define the stability and robustness of C on some test set $T \subseteq X \times L$ by the ratios:

$$\begin{aligned} \textsc{Stability}(C, T) &\triangleq \frac{|\{(\mathbf{x}, l_\mathbf{x}) \in T \mid \textsc{Stable}(C, P, \mathbf{x})\}|}{|T|} \\ \textsc{Robustness}(C, T) &\triangleq \frac{|\{(\mathbf{x}, l_\mathbf{x}) \in T \mid \textsc{Robust}(C, P, \mathbf{x}, l_\mathbf{x})\}|}{|T|} \end{aligned} \tag{2.2}$$

It is worth remarking that if we have a tie vote using the kNN algorithm, $|C(\mathbf{x})| > 1$ always holds, and hence C can be neither stable nor robust on $\mathbf{x}$ by definition.

2.2.5 Individual Fairness

When it comes to *individual fairness* of ML classifiers, as the name suggests, we want to know if similar inputs will receive a similar class label. The similarity relation on the input space X is expressed in terms of a distance δ and a threshold $\epsilon > 0$ by considering $S_{\delta,\epsilon} \triangleq \{(\mathbf{x}, \mathbf{y}) \in X \times X \mid \delta(\mathbf{x}, \mathbf{y}) \leq \epsilon\}$. According to Dwork's work [11], a model is *biased* (or *unfair*) if there is a pair of valid inputs that are close to each other for some distance function δ, but are treated differently by the model, leading to a different outcome, and it is instead *unbiased* (or *fair*) if such a pair does not exist. Consequently, given an input $\mathbf{x} \in X$, we say that a classifier $C : X \to \wp(L)$ is fair on $\mathbf{x}$ with respect to $S_{\delta,\epsilon}$ when:

$$\forall \mathbf{y} \in X.\ (\mathbf{x}, \mathbf{y}) \in S_{\delta,\epsilon} \Rightarrow C(\mathbf{x}) = C(\mathbf{y}).$$

In light of the foregoing, we now formally define the individual fairness property in adversarial scenarios.

Definition 2.3 (Individual Fairness). Given an input $\mathbf{x} \in X$ and perturbation $P_{\delta,\epsilon} : X \to \wp(X)$ such that $P_{\delta,\epsilon}(\mathbf{x}) \triangleq \{\mathbf{y} \in X \mid \delta(\mathbf{x}, \mathbf{y})\}$ for some distance function δ, a classifier $C : X \to \wp(L)$ is *fair* on $(\mathbf{x}, l_{\mathbf{x}})$ with respect to $P_{\delta,\epsilon}$, denoted by $\textsc{Fair}(C, P_{\delta,\epsilon}, \mathbf{x})$, when $\forall \bar{\mathbf{x}} \in P_{\delta,\epsilon}(\mathbf{x}).\ C(\mathbf{x}) = C(\bar{\mathbf{x}})$ holds.

By leveraging on Definition 2.3, we observe that individual fairness boils down to stability, namely, for all inputs $\mathbf{x}$, $\textsc{Fair}(C, P_{\delta,\epsilon}, \mathbf{x}) \Leftrightarrow \textsc{Stable}(C, P_{\delta,\epsilon}, \mathbf{x})$ holds.
For the individual fairness metric on a given test set $T \subseteq X \times L$ we therefore have that:

$$\textsc{Fairness}(C, T) \triangleq \frac{|\{(\mathbf{x}, l_{\mathbf{x}}) \in T \mid \textsc{Fair}(C, P_{\delta,\epsilon}, \mathbf{x})\}|}{|T|} \tag{2.3}$$

In our experiments, we have considered the Noise-Cat similarity relation as defined in [33], where two inputs $\mathbf{x}, \mathbf{y} \in X$ are similar when: (i) given a subset $N \subseteq \mathbb{N}$ of indexes of numerical features and a threshold $\epsilon \in \mathbb{R}_{\geq 0}$, for all $i \in N$, $|\mathbf{x}_i - \mathbf{y}_i| \leq \epsilon$; (ii) given a subset $C \subseteq \mathbb{N}$ of indexes of categorical features, both $\mathbf{x}$ and $\mathbf{y}$ are allowed to have any category for features with indexes in C; (iii) every other feature of $\mathbf{x}$ and $\mathbf{y}$, i.e. with index not in N nor C, must be the same, namely, for any index $i \notin N \cup C$, $\mathbf{x}_i = \mathbf{y}_i$

holds. For example, if we wanted to make a classification for statistical purposes, two individuals having two numerical features *age* and *height* and two categorical features *gender* and *country*, could be considered similar if their ages are both within the same reference range (e.g. in the age group 25-30) regardless of their country, while having the same gender and height.

2.3 Abstract Interpretation

In adversarial scenarios, to determine whether a classifier $C\colon X \to \wp(L)$ is robust on a feature vector $(\mathbf{x}, l_{\mathbf{x}}) \subseteq X \times L$ for a certain perturbation $P\colon X \to \wp(X)$, one should verify that $\forall \bar{\mathbf{x}} \in P(\mathbf{x}).\, C(\bar{\mathbf{x}}) = \{l_{\mathbf{x}}\}$ holds, namely, all (possibly infinite) feature vectors within the *adversarial region* drowned by $P(\mathbf{x})$ should be classified as $\mathbf{x}$. Clearly this is not feasible, and many people make the mistake of testing the classifier C on a finite subset $R \subset P(\mathbf{x})$, asserting it robust if $\forall \bar{\mathbf{x}} \in R.\, C(\bar{\mathbf{x}}) = \{l_{\mathbf{x}}\}$, which is obviously incorrect. We must therefore rely on an alternative strategy that does not limit itself to testing a set of vectors in $P(\mathbf{x})$, but instead verifies the robustness (and stability) of the classifier in a provable, safe, and fast manner.

The framework of *abstract interpretation*, where numerical properties can be studied with different *abstract domains*, is a good candidate for solving this kind of problem. The underlying concept in abstract interpretation is that of *over-approximation*: providing an abstraction of a complex behavior with fewer details. Over-approximations are conservative in that they can be used to prove safety properties, e.g. "inferring robustness in the abstraction" means "inferring robustness in the concrete version". Each abstract domain encodes a set of properties with a trade-off between precision and efficiency. For instance, the domain of intervals captures the constant lower and upper bounds of a variable, such as $0 \leq z \leq 1$, and provides operations like intersection or union that are linear in terms of the number of variables. In this thesis, we will leverage the interval domain to deigned the formal verification method, which turned out to be particularly suitable for the task.

In order to better understand the notions of abstraction and abstract domain, a few definitions will be provided in the sections that follow.

2.3.1 Orders and Lattices

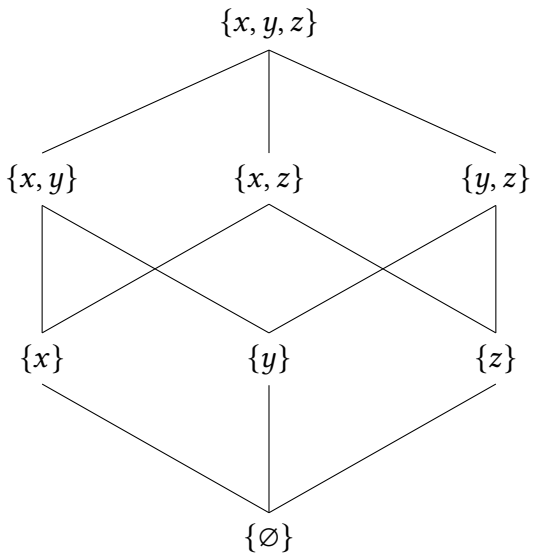

Figure 2.3: Hasse diagram of a partially-ordered complete lattice of three-element set subsets.

Definition 2.4 (Partially ordered set). A *partially ordered set* (also *poset*) is a nonempty set S together with a binary relation $\sqsubseteq$ denoting a partial order such that:

1. $\sqsubseteq$ is reflexive: $\forall a \in S.\ a \sqsubseteq a$;
2. $\sqsubseteq$ is transitive: $\forall a, b, c \in S.\ a \sqsubseteq b \wedge b \sqsubseteq c \Rightarrow a \sqsubseteq c$;
3. $\sqsubseteq$ is antisymmetric: $\forall a, b \in S.\ a \sqsubseteq b \wedge b \sqsubseteq a \Rightarrow a = b$.

Figure 2.3 shows the graphical representation (Hasse diagram) of a finite poset of three-element set subsets. Because it is possible for two elements to be incomparable, like $\{x, y\}$ and $\{x, z\}$ which lie in the same level, the ordering is not necessarily total, that is, $\forall a, b \in S.\ a \sqsubseteq b \vee b \sqsubseteq a$ not always holds.

Definition 2.5 (Lattice). A *lattice*, denoted by $\langle L, \sqsubseteq, \sqcup, \sqcap \rangle$, is a partially ordered set $\langle L, \sqsubseteq \rangle$ such that $\forall a, b \in L.\ \exists a \sqcup b \wedge \exists a \sqcap b$, namely, any pair of elements $a, b \in L$ has a *least upper bound* (or *supremum*) $a \sqcup b$ and a *greatest lower bound* (or *infimum*) $a \sqcap b$.

We remark that if a, b, c are elements of a lattice $\langle L, \sqsubseteq, \sqcup, \sqcap \rangle$, then $a, b, c \sqsubseteq (a \sqcup b) \sqcup c$, and if $a, b, c \sqsubseteq z$, then $(a \sqcup b), c \sqsubseteq z$ so $(a \sqcup b) \sqcup c \sqsubseteq z$. Hence, $(a \sqcup b) \sqcup c$ is a least upper

bound of a, b, c. By induction one shows that any finite set of elements of a lattice has a least upper bound. Similarly, any finite subset has a greatest lower bound.

Definition 2.6 (Complete Lattice). A *complete lattice*, denoted by $\langle L \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$, is a partially ordered set $\langle L, \sqsubseteq \rangle$ such that $\forall Q \subseteq L. \exists \sqcup Q \wedge \exists \sqcap Q$, namely, Q has a *least upper bound* (or *supremum*) $\sqcup Q$ and a *greatest lower bound* (or *infimum*) $\sqcap Q$. In particular, L has a least element $\bot = \sqcup \varnothing$ and a greatest element $\top = \sqcup L$.

By Definition 2.6, the poset of Figure 2.3 is also a complete lattice, where $\top = \{x, y, z\}$ and $\bot = \varnothing$. An example of a lattice that is not a complete lattice is showed in Figure 2.4: since $L = \mathbb{N}$, $\langle L, \sqsubseteq \rangle$ is an infinite totally ordered set having a least element $\bot = 0$, but not a greatest element $\top$, in contrast to the definition of complete lattice.

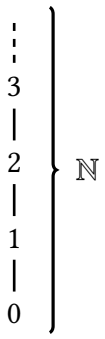

Figure 2.4: Hasse diagram of a lattice that is not a complete lattice.

2.3.2 Galois Connections

Definition 2.7 (Galois Connection). Given two posets $\langle C, \subseteq \rangle$ and $\langle A, \sqsubseteq^A \rangle$, a pair of maps $(\alpha^A, \gamma^A) \colon (C \to A) \times (A \to C)$ is a *Galois connection*, denoted by $\langle C, \subseteq \rangle \underset{\alpha^A}{\overset{\gamma^A}{\leftrightarrows}} \langle A, \sqsubseteq^A \rangle$, when:

$$\forall a \in A, c \in C.\ c \subseteq \gamma^A(a) \Leftrightarrow \alpha^A(c) \sqsubseteq^A a \tag{2.4}$$

In abstract interpretation, $\alpha^A \colon C \to A$ and $\gamma^A \colon A \to C$ are known as the *abstraction map* and the *concretization map*, respectively. If (α, γ) is given for an abstraction A of C, since $c \subseteq \gamma^A(a)$, or equivalently $\alpha^A(c) \sqsubseteq^A a$, then A is a sound abstraction of C, that is, it correctly over-approximates its information. While the monotonic concretization map γ^A is enough to reason about soundness, the property (2.4) of Galois connections provides a stronger connection between the concrete and abstract worlds in a very compact form, allowing us to design sound and accurate analyzes.

2.3.3 Numerical Abstract Domains

According to the most general definition, a *numerical abstract domain* (or *abstraction*) [23] is a partially ordered set $\langle A, \sqsubseteq^A \rangle$, very often a complete lattice, whose order relation represents the notion of approximation between abstract values. A is equipped with, at least, a concretization map $\gamma^A \colon A \to \wp(\mathbb{R}^n)$ which defines the concrete set of vectors represented by an abstract value while monotonically preserving the ordering relation, namely, $a_1 \sqsubseteq^A a_2 \Leftrightarrow \gamma^A(a_1) \subseteq \gamma^A(a_2)$ holds. Several numerical domains, including intervals and octagons, are also equipped with an abstraction map $\alpha^A \colon \wp(\mathbb{R}^n) \to A$, thus admitting a definition via Galois connections, whereas others, notably zonotopes and convex polyhedra, do not have an abstraction map. We say that a subset of vectors $S \in \wp(\mathbb{R}^n)$ is *over-approximated* by $a \in A$ when $S \subseteq \gamma(a)$, while S is *exactly* represented by a when $S = \gamma^A(a)$ holds. The main idea behind this approach is that an abstract domain serves as *symbolic representation* of a concrete domain, in our case the powerset of vectors in $\mathbb{R}^n$. Given a concrete k-ary operation on vectors $f : (\mathbb{R}^n)^k \to \mathbb{R}^n$, for some $k \in \mathbb{N}$, an abstract function $f^A : A^k \to A$ is called *sound* (or *correct*) approximation of f when for all $(a_1, ..., a_k) \in A^k$, $\{f(\mathbf{x}_1, ..., \mathbf{x}_k) \mid \forall i.\, \mathbf{x}_i \in \gamma^A(a_i)\} \subseteq \gamma^A(f^A(a_1, ..., a_k))$ holds, while f^A is defined to be *complete* (or *exact*) when equality holds. Put simply, this means that soundness holds when $f^A(a_1, ..., a_k)$ takes into account all concrete computations of f on some input $(\mathbf{x}_1, ..., \mathbf{x}_k)$, abstractly represented by $(a_1, ..., a_k)$, while completeness implies that each abstract computation $f^A(a_1, ..., a_k)$ is an exact abstract representation of the set of concrete computations of f on all the concrete inputs abstractly represented by $(a_1, ..., a_k)$.

The abstractions of the concrete domain fall into two major categories, depending on whether or not the relationships between variables are taken into account when abstracting each variable: if they are considered, we talk about *relational domains*; otherwise we refer to *non-relational domains*. The latter are clearly the simplest and include sign, constant, interval and congruence domains. Well-known examples of relational abstract domains are instead zonotopes, octagons, octahedra and polyhedra [23].

As previously anticipated, in this thesis we will consider the abstract domain of real intervals, which is one of the most popular non-relational domains. In the end, however, some considerations will be made for future improvements obtainable by exploiting a relational abstract domain.

2.3.4 The Interval Domain

Introduced for numeric analysis by Moore in 1966 [25], the interval domain is based on the homonymous arithmetic and it is currently the most used (non-relational) abstract domain. In this domain, a set of real values are (over-)approximated by the least single, possibly unbounded, real interval enclosing them. Formally, it represents a complete lattice:

$$\mathbb{I} \triangleq \{[l,u] \mid l \in \mathbb{R} \cup \{-\infty\}, u \in \mathbb{R} \cup \{+\infty\}, l \leq u\} \cup \{\bot^{\mathbb{I}}\}$$

where the top element $\top^{\mathbb{I}} = [-\infty, +\infty] = \mathbb{R}$ is implicitly defined, and the bottom element $\bot^{\mathbb{I}}$ coincides with the empty interval, for which the operators $\sqsubseteq^{\mathbb{I}}, \sqcup^{\mathbb{I}}, \sqcap^{\mathbb{I}}$ are defined as follows:

$$\begin{aligned}
[l_1,u_1] \sqsubseteq^{\mathbb{I}} [l_2,u_2] &\triangleq l_1 \geq l_2 \wedge u_1 \leq u_2 \\
[l_1,u_1] \sqcup^{\mathbb{I}} [l_2,u_2] &\triangleq [\min(l_1,l_2), \max(u_1,u_2)] \\
[l_1,u_1] \sqcap^{\mathbb{I}} [l_2,u_2] &\triangleq \begin{cases} [\max(l_1,l_2), \min(u_1,u_2)] & \text{if } \max(l_1,l_2) \leq \min(u_1,u_2) \\ \bot^{\mathbb{I}} & \text{otherwise} \end{cases}
\end{aligned}$$

The monotone concretization map $\gamma^{\mathbb{I}} \colon \mathbb{I} \to \wp(\mathbb{R})$ is standard:

$$\begin{aligned}
\gamma^{\mathbb{I}}(\bot^{\mathbb{I}}) &\triangleq \varnothing \\
\gamma^{\mathbb{I}}([l,u]) &\triangleq \{z \in \mathbb{R} \mid l \leq z \leq u\} \\
\gamma^{\mathbb{I}}([-\infty,u]) &\triangleq \{z \in \mathbb{R} \mid z \leq u\} \\
\gamma^{\mathbb{I}}([l,+\infty]) &\triangleq \{z \in \mathbb{R} \mid z \geq l\} \\
\gamma^{\mathbb{I}}(\top^{\mathbb{I}}) &\triangleq \mathbb{R}
\end{aligned}$$

Intervals are also endowed with an abstraction map $\alpha^{\mathbb{I}} \colon \wp(\mathbb{R}) \to \mathbb{I}$ providing the *best correct approximation* [7]. Its definition is:

$$\alpha^{\mathbb{I}}(X) \triangleq \begin{cases} [\inf X, \sup X] & \text{if } X \neq \varnothing \\ \bot^{\mathbb{I}} & \text{otherwise} \end{cases}$$

Although $\alpha^{\mathbb{I}}$ is not always define in $\mathbb{Q} \subset \mathbb{R}$ because the minimum and maximum may not exist, e.g. $\{z \in \mathbb{Q} \mid z \leq \sqrt{2}\}$ has no maximum in $\mathbb{Q}$, we slightly abuse the notation

$\alpha^{\mathbb{I}}$ to indicate the best computer-representable abstraction, hence $\inf X$ and $\sup X$ will be the closest computer-representable real values less than or equal to $\inf X$ and greater than or equal to $\sup X$, respectively. Thus, $(\alpha^{\mathbb{I}}, \gamma^{\mathbb{I}}) \colon (\wp(\mathbb{R}) \to \mathbb{I}) \times (\mathbb{I} \to \wp(\mathbb{R}))$ defines a Galois connection between the concrete domain $\langle \wp(\mathbb{R}), \subseteq \rangle$ and the abstract domain $\langle \mathbb{I}, \sqsubseteq^{\mathbb{I}} \rangle$.

This lattice is very intuitive and easy to deal with, but it has some severe drawbacks that we will discuss in greater detail later in the thesis. For example, the set $\{0, 1\} \subset \mathbb{R}$ with cardinality $|\{0, 1\}| = 2$, will be approximated through $\alpha^{\mathbb{I}}$ as $[0, 1]$, which concretized is a much larger set $\gamma^{\mathbb{I}}([0, 1]) = \{z \in \mathbb{R} \mid 0 \leq z \leq 1\}$ with the same cardinality as $\mathbb{R}$, resulting in an immense over-approximation.

3
Related Works

Abstract interpretation-based techniques have been successfully applied for designing precise and scalable robustness verification algorithms as well as adversarial training techniques for a wide range of ML models [27, 4, 15, 24, 26, 28, 29, 30, 35, 36, 37, 42]. To the best of our knowledge, no prior work has ever used abstract interpretation to the formal verification of k-nearest neighbors.

Formal verification methods in adversarial machine learning have been extensively researched for (deep) neural networks, whereas different major ML models, particularly non-parametric ones, have received far less attention. Specifically, adversarial examples on k-nearest neighbor algorithms have been studied only recently using various methodologies based on solving some minimization problems [45, 44, 46, 39, 41, 38]. The most relevant related work is [38], in which the authors propose a higher-order Voronoi diagram-based algorithm, called GeoAdEx, that aims to find the *smallest perturbation* that moves an input feature vector $(\mathbf{x}, l_{\mathbf{x}}) \in X \times L$ in an adversarial cell, that is, an order-k Voronoi cell with a different majority label than $l_{\mathbf{x}}$. GeoAdEx has been able to infer the smallest (up to 25% smaller than the second best result) adversarial distance compared to the baseline in most experiments. Although our goal is to verify rather than compute adversarial examples, an analyzer that outputs the shortest distance to an adversarial example can be transformed into a complete verifier, if the optimality is guaranteed, or into a sound verifier otherwise, as described in Section 5.4. Obviously, finding the optimal perturbation or a certified lower bound may often need a very long

time, owing to a combinatorial time complexity, and therefore the resulting verifier will probably be extremely inefficent but, for our case, having comparable results regardless of the total execution time is more than enough.

This chapter will provide a brief overview of the aforementioned stream of works finding adversarial examples on kNNs, up to GeoAdEx with which our experimental results have been carefully compared to.

3.1 QP-based Attack

In 2019, Wang et al. [44] studied the problem of evaluating the robustness of k-nearest neighbor classifiers, focusing more on 1NNs. They showed that finding the minimum adversarial perturbation can be formulated as a set of convex *quadratic programming* (*QP*) problems, with an exact solution in polynomial time for 1NNs. When applied to general kNN models, however, the number of constraints in the QP formulation grows exponentially with k, becoming quickly infeasible, and hence NP-hard. As a result, for $k > 1$, their method finds valid lower and upper bounds of the minimum adversarial perturbation. Given a classifier $C\colon X \to L$ and an input sample $(\mathbf{x}, l) \in X \times L$, an adversarial perturbation is defined as $\mathbf{d} \in \mathbb{R}^n$ **s.t.** $C(\mathbf{x}+\mathbf{d}) \neq l$, and is the minimum if $\forall \mathbf{d}' \in \mathbb{R}^n.\ \|\mathbf{d}'\|_2 < \|\mathbf{d}\|_2 \Rightarrow C(\mathbf{x}+\mathbf{d}') = l$. For instance, let $L = \{A, B\}$, $(\mathbf{x}, A) \in X \times L$ and $k = 1$, the problem of finding the minimum perturbation such that $\mathbf{x}+\mathbf{d}$ is closer to some $\mathbf{x_j}$ labeled with $l_j = B$ than to all class-A samples, can be formulated as the quadratic primal problem:

$$\epsilon^{(j)} \triangleq \arg\min_{\mathbf{d}} \frac{1}{2}\mathbf{d}^T\mathbf{d} \ \textbf{s.t.}\ \|\mathbf{x}+\mathbf{d}-\mathbf{x_j}\|_2^2 \leq \|\mathbf{x}+\mathbf{d}-\mathbf{x_i}\|_2^2, \quad \forall i, l_i = A$$

Although it is simple to solve, by scaling to $k = 3$ it become necessary to list all the possible combinations of $\{(j_1, j_2, j_3) \mid l_{j_1} = l_{j_2} = B, l_{j_3} = A\}$ and then solve the QP primal problem to force $\mathbf{x}+\mathbf{d}$ to be closer to $\mathbf{x_{j_1}}, \mathbf{x_{j_2}}$ than to all class-A samples except $\mathbf{x_{j_3}}$, hence:

$$\epsilon^{(j_1,j_2,j_3)} \triangleq \arg\min_{\mathbf{d}} \frac{1}{2}\mathbf{d}^T\mathbf{d} \ \textbf{s.t.}\ \|\mathbf{x}+\mathbf{d}-\mathbf{x_j}\|_2^2 \leq \|\mathbf{x}+\mathbf{d}-\mathbf{x_i}\|_2^2, \quad \forall i, i \neq j_3, l_i = A, j \in \{j_1, j_2\}$$

Thus, it is more expensive to solve. For general $k > 1$, the QP formulation will have $O(nk)$ constraints, but due to the sparsity of solutions, greedy coordinate ascent can

still solve a subproblem efficiently, computing both an upper and a lower bound, corresponding to attack and verification.

3.2 Region-based Attack

Back in 2020, Yang et al. [46] investigated adversarial examples for the most popular non-parametric classifiers, including k-nearest neighbors, decision trees and random forests. They devised a general attack technique, known as *region-based attack*, designed to work well for multiple non-parametrics. Since the difficulty in finding adversarial examples stems from the fact that these classifiers have complicated decision regions, the main idea behind this attack is to decompose the decision regions of many classifiers, such as kNN or random forest, into convex sets. Precisely, they leverage the concept of (s, m)-decomposition, that is, a partition of $\mathbb{R}^n$ into convex polyhedra $P_1, \ldots, P_s$ such that each P_i can be described by up to m linear constraints, to state a classifier $C\colon X \to L$ (s, m)-decomposable when there is an (s, m)-decomposition such that C is constant on P_i for each $i \in [1, \ldots, s]$. In this way, given such a classifier with a decomposition $P_1, \ldots, P_s$, where $C(\mathbf{y}) = l_i$ when $\mathbf{y} \in P_i$ for labels $l_i \in L$, to find an adversarial example for an input sample $\mathbf{x} \in X$ it is sufficient to output $\bar{\mathbf{x}}$ minimizing:

$$\min_{i\colon C(\mathbf{x}) \neq l_i} \; \min_{\mathbf{z} \in P_i} \; \|\mathbf{x} - \mathbf{z}\|_{p \in \{1,2,\infty\}}$$

Thus, solving the inner minimization problem results in candidates $\mathbf{z}^i \in P_i$. Taking then the outer minimum over i with $C(\mathbf{x}) \neq l_i$ leads to the optimal adversarial example:

$$\bar{\mathbf{x}} \triangleq \arg\min_{\mathbf{z}^i} \; \|\mathbf{x} - \mathbf{z}^i\|_{p \in \{1,2,\infty\}}$$

The exact attack algorithm's performance is determined by two factors: (i) the number of regions, which is determined by the complexity of the classifier, and (ii) the number of constraints and dimensionality of the polyhedra. For kNN classifiers the number of convex polyhedra scales with $O(n^k)$: when $k = 1$, this is efficiently solvable, because polyhedra have at most n constraints and the adversarial examples can be found quickly using a linear program for ℓ_∞-perturbations. Unfortunately, for $k > 1$, region-based attacks do not scale well, and an approximation algorithm for larger values of k is also discussed.

3.3 Gradient-based Attack

Inspired by their previous work [40] on the robustness of deep kNNs, in 2020 Sitawarin and Wagner [41] proposed a new state-of-the-art attack, called *gradient-based attack*, to evaluate the robustness of kNN classifiers. Their method also outperforms Yang et al [46] in that when $k > 1$, an adversarial example with a smaller perturbation is found in less than 1% of the running time. Moreover, increasing the value of k only slightly increases the algorithm's running time. In order to find the minimum $\mathbf{d}^* \in \mathbb{R}^n$ **s.t.** $\bar{\mathbf{x}} \triangleq \mathbf{x} + \mathbf{d}^*$ is an adversarial example classified differently than $\mathbf{x} \in X$, the following minimization problem is solved:

$$\mathbf{d}^* \triangleq \arg\min_{\mathbf{d}} \sum_{i=1}^{m} \max\left\{w_i \left(\|\bar{\mathbf{x}}_\mathbf{i} - (\mathbf{x} + \mathbf{d})\|_2^2 - \eta^2\right) + \Delta, 0\right\}$$

where m denote the mean of examples with different class closest to $\mathbf{x}$ in ℓ_2-norm, $w_i = 1$ if $\bar{\mathbf{x}}_\mathbf{i}$ is adversarial, otherwise $w_i = -1$, and η is the distance to the k-th nearest neighbor. The changes compared to the original version are in using the rectifier $\max(z \in \mathbb{R}, 0)$ instead of sigmoid, which avoids the need to deal with overflow and underflow issues caused by the exponential in distance computation, and the introduction of a small gap Δ to ensures that $\bar{\mathbf{x}}_\mathbf{i}$ is a bit closer to the guide samples from the incorrect class than the ones from the correct class. Unfortunately gradient-based approaches generally do not work well for finding minimum ℓ_∞-norm adversarial examples, therefore only ℓ_2-norm is considered. Furthermore, this method does not guarantee the optimality of the solution, not even for $k = 1$.

3.4 Geometric-based Attack

GeoAdEx [38], which stands for *geometric adversarial example*, is the product of the most recent work finding adversarial examples on kNNs. This novel tool was proposed by Sitawarin et al. back in 2021, and is the first using geometric approach to perform a search that expands outwards from the given input sample. The main idea of GeoAdEx is to perform a principled geometric exploration around the test sample by processing order-k Voronoi cells à la breadth-first search until it discovers an adversarial cell. Formally, the goal of this algorithm is to find the smallest perturbation $\mathbf{d}^* \in \mathbb{R}^n$ that moves

a test point $(\mathbf{x}, l_{\mathbf{x}}) \in X \times L$ to an adversarial cell, that is, an order-k Voronoi cell that has different majority than label $l_{\mathbf{x}}$. Such objective can be expressed as the following optimization problem:

$$\mathbf{d}^* \triangleq \arg\min_{\mathbf{d}} \ \|\delta\|_2^2 \quad \text{s.t.} \quad \mathbf{x} + \mathbf{d} \in A(x)$$

where $A(x)$ is the set of all adversarial cells with respect to $(\mathbf{x}, l_{\mathbf{x}})$ and ℓ_2-norm. The above constraint implies that $\mathbf{x}+\mathbf{d}$ must be a member of an adversarial cell from $A(x)$. A simple approach to solve this minimization is to build a series of optimization problems, one for each of the cells in $A(x)$, and pick the solution with the minimum adversarial distance. Unfortunately, this would require solving $O\left(\binom{n}{k}\right)$ QP problems, each of which has $k(n-k)$ constraints. While this complexity may be manageable when $k = 1$ and n is small, as it was in the other works, it does not scale well with k becoming quickly unusable. To deal with $k > 1$, the authors of this tool have added an approximated version of the algorithm which consists in bounding the number of neighboring cells taken into account according to a fast heuristic. However, the main drawback of this approach is that the approximation may affect the optimality, which is therefore no longer guaranteed. Overall this geometric-based attack finds an adversarial distance that is closer to the optimal than the baselines, but its main limitation is the excessively long runtime of the non-approximated version.

4

Abstract Verification Framework

This chapter will describe our main contribution, that is, a novel formal verification method for inferring when a kNN classifier is *provably robust* and/or *stable* for an input sample with respect to a given perturbation. Following the stream of works applying abstract interpretation for verifying ML models, we design a sound abstract version C^A_{D,k,δ^A} of a kNN classifier $C_{D,k,\delta}$ based on a ground truth dataset D of examples, and a distance function δ quantifying in $\mathbb{R}$ the proximity between two samples. The resulting over-approximate classifier C^A_{D,k,δ^A} is domain agnostic and thus defined on a numerical abstract domain A that can be instantiate to any abstract domain capable of representing input space properties, i.e., sets of vectors in $\wp(\mathbb{R}^n)$.

In this thesis, it has been instantiate to the numerical abstract domain of intervals, which represents exactly the perturbations based on Minkowski distance. Furthermore, C^A_{D,k,δ^A} leverages a sound abstract approximation $\delta^A \colon A \times A \to A$ of the distance function δ, which in turn relies on correct abstract approximations of basic numerical operations such as addition, subtraction, modulus and exponential.

Definition 4.1 (Sound Abstract Classifier). Given a numerical abstract domain A and a classifier $C \colon X \to \wp(L)$, an abstract classifier $C^A \colon A \to \wp(L)$ is a *sound abstraction* of C on A when $\forall a \in A$, $\bigcup_{\mathbf{x} \in \gamma^A(a)} C(\mathbf{x}) \subseteq C^A(a)$ holds, that is, the abstract classifier $C^A(a)$ over-approximates all the output labels of C on inputs abstractly represented by a.

Definition 4.2 (Complete Abstract Classifier). Given a numerical abstract domain A and a classifier $C\colon X \to \wp(L)$, an abstract classifier $C^A\colon A \to \wp(L)$ is a *complete abstraction* of C on A when $\forall a \in A$, $\bigcup_{\mathbf{x}\in\gamma^A(a)} C(\mathbf{x}) = C^A(a)$ holds, that is, the abstract classifier $C^A(a)$ produces all and only the output labels of C on inputs abstractly represented by a.

Given an abstract value $a \in A$ which provides a sound symbolic approximation of an adversarial region drown by a perturbation $P\colon X \to \wp(X)$ applied on an input sample $(\mathbf{x}, l_\mathbf{x}) \in X \times L$, $C^A_{D,k,\delta^A}(a)$ returns an over-approximation (superset) of the set of classes computed by $C_{D,k,\delta}$ for all the samples in $P(\mathbf{x})$.

Theorem 4.1 (Abstract Stability Verification). *Let $C^A\colon A \to \wp(L)$ be a sound abstraction of $C\colon X \to \wp(L)$ and assume that an adversarial region $P(\mathbf{x}) \subseteq X$, corresponding to some perturbation of a feature vector $\mathbf{x} \in X$, is over-approximated by some $a \in A$. If $|C^A(a)| = 1$ then* $\textsc{Stable}(C, P, \mathbf{x})$.

Proof. By hypothesis, there exists a label $l \in L$ such that $C^A(a) = \{l\}$. By the soundness hypothesis of C^A, we have that $\bigcup_{\mathbf{x}\in\gamma^A(a)} C(\mathbf{x}) \subseteq C^A(a) = \{l\}$. Since, for all $\mathbf{x}$, $C(\mathbf{x}) \neq \emptyset$, we have that for all $\mathbf{x} \in \gamma^A(a)$, $C(\mathbf{x}) = \{l\}$. Since $P(\mathbf{x})$ is over-approximated by a, $P(\mathbf{x}) \subseteq \gamma^A(a)$ holds, and we obtain that for all $\bar{\mathbf{x}} \in P(\mathbf{x})$, $C(\bar{\mathbf{x}}) = \{l\}$, namely, $\textsc{Stable}(C, P, \mathbf{x})$ holds. □

Theorem 4.2 (Abstract Robustness Verification). *Let $C^A\colon A \to \wp(L)$ be a sound abstraction of $C\colon X \to \wp(L)$ and assume that an adversarial region $P(\mathbf{x}) \subseteq X$, corresponding to some perturbation of a sample $(\mathbf{x}, l_\mathbf{x}) \in X \times L$, is over-approximated by some $a \in A$. If $C^A(a) = \{l_\mathbf{x}\}$ then* $\textsc{Robust}(C, P, \mathbf{x}, l_\mathbf{x})$.

Proof. By the soundness hypothesis of C^A, we have that $\bigcup_{\mathbf{x}\in\gamma^A(a)} C(\mathbf{x}) \subseteq C^A(a) = \{l_\mathbf{x}\}$. Since, for all $\mathbf{x}$, $C(\mathbf{x}) \neq \emptyset$, we have that for all $\mathbf{x} \in \gamma^A(a)$, $C(\mathbf{x}) = \{l_\mathbf{x}\}$. Since $P(\mathbf{x})$ is over-approximated by a, $P(\mathbf{x}) \subseteq \gamma^A(a)$ holds, and we obtain that for all $\bar{\mathbf{x}} \in P(\mathbf{x})$, $C(\bar{\mathbf{x}}) = \{l_\mathbf{x}\}$, namely, $\textsc{Stable}(C, P, \mathbf{x}, l_\mathbf{x})$ holds. □

Thus, by Theorems 4.1 and 4.2, if $|C^A_{D,k,\delta^A}(a)| = 1$ then we can safely infer that $C_{D,k,\delta}$ is *stable* on the sample $(\mathbf{x}, l_\mathbf{x})$ for the perturbation P, while $C_{D,k,\delta}$ is also *robust* for the same if $C^A_{D,k,\delta^A}(a) = \{l_\mathbf{x}\}$ holds. It is worth observing that $\textsc{Robust}(C_{D,k,\delta}, P, \mathbf{x}, l_\mathbf{x}) \implies$

$\textsc{Stable}(C_{D,k,\delta}, P, \mathbf{x})$, but $\textsc{Stable}(C_{D,k,\delta}, P, \mathbf{x}) \not\Rightarrow \textsc{Robust}(C_{D,k,\delta}, P, \mathbf{x}, l_{\mathbf{x}})$. Our abstract interpretation-based method does not satisfy Definition 4.2, therefore the converse of Theorems 4.1 and 4.2, in general, does not hold, meaning that this verification method is incomplete.

4.1 Abstract Distance

The kNN algorithm fully relies on a distance function $\delta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ to determine the k closest feature vectors to a given input sample. Although kNN is parametric on δ, hence can be freely chosen, Minkowski distance is the most common choice: given $p \in \mathbb{N}$, $\delta_p(\mathbf{x}, \mathbf{y}) \triangleq \sqrt[p]{\sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{y}_i|^p}$. In particular, we just consider the two most common instances:

$$\begin{aligned} p = 1, \text{ Manhattan distance:} \quad & \mu(\mathbf{x}, \mathbf{y}) \triangleq \textstyle\sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{y}_i| \\ p = 2, \text{ Euclidean distance:} \quad & \eta(\mathbf{x}, \mathbf{y}) \triangleq \sqrt{\textstyle\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{y}_i)^2} \end{aligned}$$

Because kNN only uses distances for relative comparisons, we can safely remove the p-th root to simplify the computations and, more importantly, to reduce the verifier's overall complexity. As a result, a numerical abstract domain must provide complete, or at least sound, abstractions of the operations used to compute distances, i.e., addition, subtraction, absolute value, and exponential, as well as a comparison test between distances. The *most precise abstractions* of these numerical operations on the interval domain $\mathbb{I}$ are well-known in literature [23]. They are:

addition: $[l_1, u_1] +^{\mathbb{I}} [l_2, u_2] \triangleq [l_1 + l_2, u_1 + u_2]$

subtraction: $[l_1, u_1] -^{\mathbb{I}} [l_2, u_2] \triangleq [l_1 - u_2, u_1 - l_2]$

multiplication: $[l_1, u_1] \cdot^{\mathbb{I}} [l_2, u_2] \triangleq [\min(l_1 l_2, l_1 u_2, u_1 l_2, u_1 u_2), \max(l_1 l_2, l_1 u_2, u_1 l_2, u_1 u_2)]$

modulus: $|[l, u]|^{\mathbb{I}} \triangleq \begin{cases} [\min(|l|, |u|), \max(|l|, |u|)] & \text{if } lu \geq 0 \\ [0, \max(|l|, |u|)] & \text{otherwise} \end{cases}$

exponential: $[l, u]^{p^{\mathbb{I}}} \triangleq \begin{cases} [l^p, u^p] & \text{if } p \text{ odd or } l \geq 0 \\ \prod_{1}^{p\,\mathbb{I}} [l, u] \sqcap^{\mathbb{I}} [0, +\infty] & \text{otherwise} \end{cases}$

with meet: $[l_1, u_1] \sqcap^{\mathbb{I}} [l_2, u_2] \triangleq [\max(l_1, l_2), \min(u_1, u_2)]$

To compare two abstract distances obtained with the aforementioned arithmetic operators, we introduce the notions of dominance and strict dominance for the interval abstract domain.

Definition 4.3 (Interval Dominance). Given two intervals $[l_1, u_1]$ and $[l_2, u_2]$, we say that:

- $[l_1, u_1]$ is *dominated* by $[l_2, u_2]$, denoted by $[l_1, u_1] \leq^{\mathbb{I}} [l_2, u_2]$, when $u_1 \leq l_2$;
- $[l_1, u_1]$ is *strictly dominated* by $[l_2, u_2]$, denoted by $[l_1, u_1] <^{\mathbb{I}} [l_2, u_2]$, when $u_1 < l_2$.

We remark that the order relation $\leq^{\mathbb{I}}$ is the so-called interval order [13], and does not coincide with the partial order $\sqsubseteq^{\mathbb{I}}$ of the domain's lattice. Figure 4.1 graphically highlights the difference.

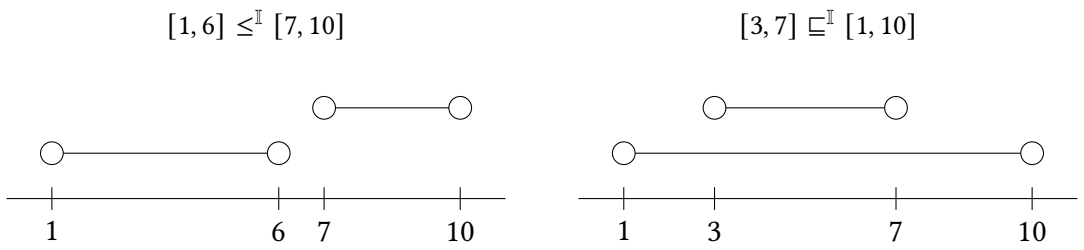

Figure 4.1: The order relation $\leq^{\mathbb{I}}$ holds only in the left picture, while the partial order $\sqsubseteq^{\mathbb{I}}$ holds only in the right picture.

Moreover, numerical operators can also be defined for the hyperrectangle product domain $\mathbb{I}^n$ simply by independently applying those for the interval domain componentwise. Such an abstract Minkowski distance computation (without the p-th root $\sqrt[p]{\cdot}$) on the hyperrectangle abstraction $\mathbb{I}^n$ does not lose precision and is thus complete.

Theorem 4.3 (Completeness of Interval Minkowski Distance). *The abstract computation for Minkowski distance over hyperrectangles is complete, namely, for all* $\mathbf{a}, \mathbf{b} \in \mathbb{I}^n$*:*

$$\bigcup_{\mathbf{x} \in \gamma^{\mathbb{I}^n}(a), \mathbf{y} \in \gamma^{\mathbb{I}^n}(b)} \{\delta_p(\mathbf{x}, \mathbf{y})\}^{\mathbb{I}} = \delta_p^{\mathbb{I}}(\mathbf{a}, \mathbf{b}) \triangleq (|\mathbf{a}_1 -^{\mathbb{I}} \mathbf{b}_1|^{\mathbb{I}})^{p^{\mathbb{I}}} +^{\mathbb{I}} \cdots +^{\mathbb{I}} (|\mathbf{a}_n -^{\mathbb{I}} \mathbf{b}_n|^{\mathbb{I}})^{p^{\mathbb{I}}}$$

Proof. Since the composition of locally complete abstract functions preserves their completeness, it is enough to show that all the abstract operations involved in the distance computation are. In fact, if A is a numerical abstraction of $\wp(\mathbb{R}^n)$ and $f^A\colon A^i \to A^j$ and $g^A\colon A^j \to A^k$ are two abstract functions that are complete with respect to, respectively, $f\colon \wp(\mathbb{R}^n)^i \to \wp(\mathbb{R}^n)^j$ and $g\colon \wp(\mathbb{R}^n)^j \to \wp(\mathbb{R}^n)^k$ then $g^A \circ f^A : A^i \to A^k$ is complete for $g \circ f\colon \wp(\mathbb{R}^n)^i \to \wp(\mathbb{R}^n)^k$ because $\gamma^A \circ g^A \circ f^A = g \circ \gamma^A \circ f^A = g \circ f \circ \gamma^A$. We observe that if $[l_1, u_1]$ and $[l_2, u_2]$ are intervals in $\mathbb{R}$, then so is $h([l_1, u_1], [l_1, u_1])$ provided $h\colon \mathbb{I}\times\mathbb{I} \to \mathbb{I}$ is continuous on a domain containing $[l_1, u_1] \times [l_2, u_2]$. This last point stems from some of the general properties of continuous functions, namely, that continuous functions map connected sets into connected sets, and map compact sets into compact sets, and since the compact sets of $\mathbb{R}$ are just the closed and bounded subsets, then h does too. Because addition, subtraction, multiplication, modulus, and exponential are continuous in both arguments, we conclude that those operation are complete. Hence, we have shown that the definition of the interval Minkowski distance $\delta_p^{\mathbb{I}}(\mathbf{a}, \mathbf{b})$ is a composition of complete abstract operations, so that $\delta_p^{\mathbb{I}}$ turns out to be complete. □

Although completeness of the distance function improves precision, it is insufficient to achieve completeness of the abstract classifier. The following example shows an abstract kNN classifier on hyperrectangles which is incomplete.

Example 4.1 (Incompleteness of Interval Abstract Classifier). Let us consider a ground truth dataset $D = \{(\mathbf{v}, l_1), (\mathbf{w}, l_2)\}$, where $\mathbf{v} = (0, 0), \mathbf{w} = (2.1, 0) \in \mathbb{R}^2$, and its corresponding 1NN classifier $C_{D,\eta,1}$ for the Euclidean distance η. Consider an adversarial region $R = \{\mathbf{x} \in \mathbb{R}^2 \mid 0 \leq \mathbf{x}_1 \leq 1, -1 \leq \mathbf{x}_2 \leq 1\}$. The Euclidean distances of a generic vector $\mathbf{x}$ from $\mathbf{v}$ and $\mathbf{w}$ are as follows:

$$\eta(\mathbf{x}, \mathbf{v}) = \sqrt{(\mathbf{x}_1 - \mathbf{v}_1)^2 + (\mathbf{x}_2 - \mathbf{v}_2)^2} = \sqrt{\mathbf{x}_1^2 + \mathbf{x}_2^2}$$
$$\eta(\mathbf{x}, \mathbf{w}) = \sqrt{(\mathbf{x}_1 - 2.1)^2 + \mathbf{x}_2^2} = \sqrt{\mathbf{x}_1^2 - 4.2\mathbf{x}_1 + 4.41 + \mathbf{x}_2^2}$$

The comparison between $\eta(\mathbf{x}, \mathbf{v})$ and $\eta(\mathbf{x}, \mathbf{w})$ is therefore as follows:

$$\eta(\mathbf{x}, \mathbf{v}) < \eta(\mathbf{x}, \mathbf{w}) \Leftrightarrow \sqrt{\mathbf{x}_1^2 + \mathbf{x}_2^2} < \sqrt{\mathbf{x}_1^2 - 4.2\mathbf{x}_1 + 4.41 + \mathbf{x}_2^2}$$
$$\Leftrightarrow 0 < -4.2\mathbf{x}_1 + 4.41 \Leftrightarrow \mathbf{x}_1 < 4.41/4.2 = 1.05$$

which holds for every adversarial sample in R, where $\mathbf{x}_1 < 1.05$. Hence, $\mathbf{v}$ is always the

closest neighbor, and every sample in R is classified by $C_{D,\eta,1}$ as l_1, so robustness holds. When performing the abstract verification on the interval abstraction $\mathbb{I}$, where R is exactly represented by $a = \langle [0,1], [-1,1] \rangle \in \mathbb{I}^2$, and the two vectors $\mathbf{v}, \mathbf{w}$ are represented, respectively., by $\mathbf{v}^{\mathbb{I}} = \langle [0,0], [0,0] \rangle$ and $\mathbf{w}^{\mathbb{I}} = \langle [2.1, 2.1], [0,0] \rangle$, the abstract Euclidean distances, without square roots, turns out to be:

$$\begin{aligned}
\eta^{2^{\mathbb{I}}}(a, \mathbf{v}^{\mathbb{I}}) &= (|[0,1] -^{\mathbb{I}} [0,0]|^{\mathbb{I}})^{2^{\mathbb{I}}} +^{\mathbb{I}} (|[-1,1] -^{\mathbb{I}} [0,0]|^{\mathbb{I}})^{2^{\mathbb{I}}} \\
&= (|[0,1]|^{\mathbb{I}})^{2^{\mathbb{I}}} +^{\mathbb{I}} (|[-1,1]|^{\mathbb{I}})^{2^{\mathbb{I}}} \\
&= [0,1]^{2^{\mathbb{I}}} +^{\mathbb{I}} [0,1]^{2^{\mathbb{I}}} = [0,1] +^{\mathbb{I}} [0,1] = [0,2] \\
\eta^{2^{\mathbb{I}}}(a, \mathbf{w}^{\mathbb{I}}) &= (|[0,1] -^{\mathbb{I}} [2.1, 2.1]|^{\mathbb{I}})^{2^{\mathbb{I}}} +^{\mathbb{I}} (|[-1,1] -^{\mathbb{I}} [0,0]|^{\mathbb{I}})^{2^{\mathbb{I}}} \\
&= (|[-2.1, -1.1]|^{\mathbb{I}})^{2^{\mathbb{I}}} +^{\mathbb{I}} (|[-1,1]|^{\mathbb{I}})^{2^{\mathbb{I}}} \\
&= [1.1, 2.1]^{2^{\mathbb{I}}} +^{\mathbb{I}} [0,1]^{2^{\mathbb{I}}} = [1.21, 4.41] +^{\mathbb{I}} [0,1] = [1.21, 5.41]
\end{aligned}$$

Hence, these abstract distances do not allow us to infer the closest vector because neither $\eta^{2^{\mathbb{I}}}(a, \mathbf{v}^{\mathbb{I}}) <^{\mathbb{I}} \eta^{2^{\mathbb{I}}}(a, \mathbf{w}^{\mathbb{I}})$ nor $\eta^{2^{\mathbb{I}}}(a, \mathbf{w}^{\mathbb{I}}) <^{\mathbb{I}} \eta^{2^{\mathbb{I}}}(a, \mathbf{v}^{\mathbb{I}})$ hold. This lack of precision is rooted in non-relational domains, as they are unable to represent any relationship between variables and therefore, at the end of the computation, we lose sight of the fact that, in this example, proximity is solely determined by $\mathbf{x}_1$. □

4.2 Formal Verification Method

In this section, we will go over our formal verification method based on abstract interpretation in detail. The algorithm steps will be divided into subsections, each with its own pseudocode. Finally, the pseudocode of the whole abstract verifier as well as the proof of soundness will be provided.

Let A be a numerical abstract domain supporting the aforementioned basic operations and having, at the very least, a concretization map $\gamma^A \colon A \to \wp(\mathbb{R})$. Then, given a ground truth dataset $D = \{(\mathbf{x}_1, l_1), ..., (\mathbf{x}_N, l_N)\} \subseteq X \times L$, a number of neighbors $k \in \mathbb{N}^* \triangleq \mathbb{N} \setminus \{0\}$, and an abstract distance function $\delta^A : A \times A \to A$, the sound abstract version of the kNN classifier that we will describe is modeled as the total function $C^A_{D,k,\delta^A} \colon A \to \wp(L)$, mapping an abstract value $a \in A$ into a nonempty set of labels.

4.2.1 Computation of Abstract Distances

Algorithm 4.1 Computation of abstract distances — time complexity: $O(n \cdot |D|)$

1: $M, O \leftarrow \varnothing$
2: **for all** $(\mathbf{y}, l_\mathbf{y}) \in D$
3: $\quad \mathbf{y}^A \leftarrow \alpha(\mathbf{y})$
4: $\quad d_\mathbf{y} \leftarrow \delta(a, \mathbf{y}^A)$
5: $\quad O \leftarrow O.\textsc{Append}\big((d_\mathbf{y}, l_\mathbf{y})\big)$
6: **end for**
7: $\textsc{Heapify}(O)$

Let $(\mathbf{x}, l_\mathbf{x}) \in X \times L$ and $P : X \to \wp(X)$ be, respectively, the input sample and an adversarial perturbation. First, we need a sound abstraction $a \in A$ for $P(\mathbf{x})$, and an abstract representation $\mathbf{y}^A \in A$ for every $\mathbf{y}$ occurring in the dataset as $(\mathbf{y}, l_\mathbf{y}) \in D$. For abstract domains endowed with an abstraction map $\alpha^A : \wp(\mathbb{R}^n) \to A$, this preliminary stage is trivial, and it is enough to use it for both $P(\mathbf{x})$ and $\mathbf{y}$. If a best-correct approximation for A does not exist, any function providing a sound over-approximation can be used.

We proceed by computing the abstract distances $d_\mathbf{y} = \delta^A(a, y^A) \in A$ for every $(\mathbf{y}, l_\mathbf{y}) \in D$, and consider the pairs $(d_\mathbf{y}, l_\mathbf{y}) \in A \times L$ to which we extend the ordering relation $<^A$ by disregarding labels: we say that $(d_\mathbf{v}, l_\mathbf{v}) <^{A\times L} (d_\mathbf{w}, l_\mathbf{w})$ if and only if $d_\mathbf{v} <^A d_\mathbf{w}$, and similarly, $(d_\mathbf{v}, l_\mathbf{v}) \leq^{A\times L} (d_\mathbf{w}, l_\mathbf{w})$ if and only if $d_\mathbf{v} \leq^A d_\mathbf{w}$. We store such pairs into a very efficient ordered data structure O, implemented using a *min-heap*, that is, a binary tree where the value of a node is greater than or equal to the value of its parent node. In this structure, if an element o_j in position j strictly dominates an element o_i in position i, then every element o_h after o_j also strictly dominates o_i, i.e., $o_i <^{A\times L} o_j \implies \forall h \geq j \colon o_i <^{A\times L} o_h$. In the optimal case, i.e. $o_1 <^{A\times L} o_2 <^{A\times L} \ldots <^{A\times L} o_{|D|}$, picking the k nearest neighbors corresponds to extracting the first k elements. In more complex scenarios, distances may overlap as described in Example 4.1. The time complexity for computing each distance is $O(n)$, where n is the number of features in a sample, therefore for $|D|$ distances is $O(n \cdot |D|)$. Furthermore, inserting (or extracting) a pair to a min-heap has a logarithmic time complexity, hence for $|D|$ pairs both actions are $O(|D| \cdot \log |D|)$. Inserting a pair requires the min-heap to place it in the correct position in the tree to ensure that the above property holds, however, if the tree is built starting from a list of pairs, this can be done in linear time. Thus, distances are computed and stored in $O(n \cdot |D|)$.

4.2.2 Computation of Score Intervals

Algorithm 4.2 Computation of score intervals — time complexity: $O(|D| \cdot \log |D|)$

1: **for all** $l \in L$
2: $\quad \text{lb}(l), \text{ub}(l) \leftarrow 0$
3: **end for**
4: **for all** $i \in [1, k]$
5: $\quad M[i] \leftarrow O.\textsc{ExtractMin}()$
6: **end for**
7: **for all** $(d_z, l_z) \in M$
8: $\quad \text{ub}(l_z) \leftarrow \text{ub}(l_z) + 1$
9: $\quad$ **if** $\nexists (d_y, l_y) \in O$ **such that** $l_z \neq l_y \wedge d_z \not<^A d_y$
10: $\quad\quad \text{lb}(l_z) \leftarrow \text{lb}(l_z) + 1$
11: $\quad$ **end if**
12: **end for**
13: $\sigma_k \leftarrow \sum_{l \in L} \text{lb}(l)$
14: **if** $\sigma_k < k$
15: $\quad$ **for all** $(d_y, l_y) \in O$
16: $\quad\quad$ **if** $\exists (d_z, l_z) \in M$ **such that** $l_z \neq l_y \wedge d_z \not<^A d_y$
17: $\quad\quad\quad$ **if** $\text{ub}(l_y) - \text{lb}(l_y) < k - \sigma_k$
18: $\quad\quad\quad\quad \text{ub}(l_y) \leftarrow \text{ub}(l_y) + 1$
19: $\quad\quad\quad$ **end if**
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: **end if**

Once the ordered structure is created, we compute abstract scores for labels using the interval abstract domain $\mathbb{I}$. We first initialize a score $s[l] = [0, 0]$ for each label $l \in L$, where $\text{lb}(l)$ denotes the lower bound of $s[l]$, and $\text{ub}(l)$ its upper bound, then we extract the first k pairs from O. For each extracted pair (d_z, l_z), we check whether exists a pair $(d_y, l_y) \in O$ such that $l_z \neq l_y$ and $d_z \not<^A d_y$. If such pair does not exist, then there are no feature vectors with label different than l_x which are closer to $P(\mathbf{x})$ than $\mathbf{z}$, hence l_z will be surely in the neighborhood of every point in $P(\mathbf{x})$. It is therefore sound to increase both lower and upper bounds of $s[l_z]$, i.e., $\text{lb}(l_z) = \text{lb}(l_z) + 1$ and $\text{ub}(l_z) = \text{ub}(l_z) + 1$.

Otherwise, it is not possible to assert whether l_z or l_y will be considered, so the lower bound of $s[l_z]$ cannot be increased. In this last case the sum $\sigma_k \triangleq \sum_{l \in L} \mathrm{lb}(l)$ of the current lower bounds will be less than k, meaning that there are unprocessed pairs (d_y, l_y) in O whose distances do not strictly dominate all the distances of the first k extracted pairs. Since those ones can potentially be neighbors, it is sound to increase their upper bounds, thus representing a possible vote.
The computation of score intervals is the computationally most relevant part of the algorithm. Comparisons are $O\big(k \cdot (|D| - k)\big)$, and for small values of k the running time of the algorithm increases linearly with the size of D. Since O could be non-linearly ordered, the worst possible case is that all the distances overlap leading us to look at all $|D|$ distances in the min-heap, for a total time complexity of $O(|D| \cdot \log |D|)$.

4.2.3 Abstract Classification

Algorithm 4.3 Abstract classification — time complexity: $O(|L|^2)$

for all $l \in L$
  if $\mathrm{ub}(l) = 0$
    $L \leftarrow L \setminus \{l\}$
  end if
end for
if $(|L| = 1$ **or** $k = 1)$
  return L
end if
for all $l \in L$
  $\mu \leftarrow \min(k, \sum_{l' \neq l} \mathrm{ub}(l'))$
  $\mathrm{lb}(l) \leftarrow \max(\mathrm{lb}(l), k - \mu)$
end for
$\tau \leftarrow \lceil \frac{k}{\min(k, |L|)} \rceil$
return $\{l \in L \mid \mathrm{ub}(l) \geq \tau,\ \forall l' \in L \setminus \{l\}.\ s[l] \not<^{\mathbb{I}} s[l']\}$

At this point, we leverage the interval scores to perform the classification. First, we remove all interval scores having $\mathrm{ub}(l) = 0$, as by soundness their labels never occur in the nearest neighbors, and then we try to narrow each of remaining interval scores.

For example, let us consider the binary classification case with $k = 7$ and three labels whose scores are $\mathrm{s}[A] = [2, 4]$, $\mathrm{s}[B] = [1, 3]$ and $\mathrm{s}[C] = [0, 0]$. We can safely remove C from the final set of labels, and then wen see that both lower bounds of $\mathrm{s}[A]$ and $\mathrm{s}[B]$ can be increased in a sound way: since the sum of the two labels must be 7, the only possible way is that $\mathrm{s}[A] = [4, 4]$ and $\mathrm{s}[B] = [3, 3]$. It follows that A is the most voted label. Formally, we can safely set each score lower bound $\mathrm{lb}(l)$ to $k - \sum_{l' \neq l} \mathrm{ub}(l')$ if this is improving. When we consider the sum of all the other upper bounds, we are explicitly taken the worst case scenario with respect to l. Thus, if we still not have k neighbors in the worst case scenario, those that remain must be of class l, as it is the only one that has not yet been considered.
After this refinement, we return the set of labels whose score intervals are numerically significant and maximals for $<^{\mathbb{I}}$:

$$output \triangleq \{l \in L \mid \mathrm{ub}(l) \geq \lceil \tfrac{k}{\min(k,|L|)} \rceil,\ \forall l' \in L \setminus \{l\}.\ \mathrm{s}[l] \not<^{\mathbb{I}} \mathrm{s}[l']\} \tag{4.1}$$

We are thus excluding only those labels whose score interval either has an upper bound strictly less than $\lceil \frac{k}{\min(k,|L|)} \rceil$ or is not maximal, i.e., the labels whose number of votes is surely less than that of another label. This definition is sound because in (4.1) no real classification label is forgotten, while the lack of precision in comparing the abstract distances may lead to an over-approximation that includes some spurious labels. For example, let us consider a 5NN and four labels such that $\mathrm{s}[A] = [0, 5]$, $\mathrm{s}[B] = [1, 5]$, $\mathrm{s}[C] = [0, 1]$ and $\mathrm{s}[D] = [1, 1]$. By applying the refinement nothing change, but we observe that both C and B will surly not appear in the concrete set of labels; in fact, if they are both taken into account, 3 other neighbors remains to be considered, and their labels must be A or B. Hence, either A or B will receive 2 votes, thus excluding C and D. The final set of labels will therefore be $\{A, B\}$, which follows from the fact that $\mathrm{ub}(C) = \mathrm{ub}(D) < 2 = \lceil \frac{5}{\min(5,4)} \rceil$ and $\mathrm{s}[A]$ as well as $\mathrm{s}[B]$ are maximal.
The time complexity of this classification process is solely determined by $|L|$, Since $|L|$ is typically very small, usually 2 (e.g. binary classification), it is asymptotically irrelevant.

4.2.4 Full Algorithm

Algorithm 4.4 shows the whole pseudocode of the abstract classifier $C^{A}_{D,k,\delta^A}(a)$ with input a abstracting $P(\mathbf{x})$. Its time complexity is the same as that of computing score intervals.

Algorithm 4.4 Abstract classifier $C^A_{D,k,\delta^A}(a)$ time complexity: $O(|D| \cdot \log |D|)$

$M, O \leftarrow \varnothing$
for all $(\mathbf{y}, l_\mathbf{y}) \in D$
  $\mathbf{y}^A \leftarrow \alpha(\mathbf{y})$
  $d_\mathbf{y} \leftarrow \delta(a, \mathbf{y}^A)$
  $O \leftarrow O.\textsc{Append}\big((d_\mathbf{y}, l_\mathbf{y})\big)$
end for
$\textsc{Heapify}(O)$
for all $l \in L$ **do** $\mathrm{lb}(l), \mathrm{ub}(l) \leftarrow 0$
for all $i \in [1, k]$ **do** $M[i] \leftarrow O.\textsc{ExtractMin}()$
for all $(d_\mathbf{z}, l_\mathbf{z}) \in M$
  $\mathrm{ub}(l_\mathbf{z}) \leftarrow \mathrm{ub}(l_\mathbf{z}) + 1$
  if $\nexists (d_\mathbf{y}, l_\mathbf{y}) \in O$ **such that** $l_\mathbf{z} \neq l_\mathbf{y} \wedge d_\mathbf{z} \not<^A d_\mathbf{y}$ **then** $\mathrm{lb}(l_\mathbf{z}) \leftarrow \mathrm{lb}(l_\mathbf{z}) + 1$
end for
$\sigma_k \leftarrow \sum_{l \in L} \mathrm{lb}(l)$
if $\sigma_k < k$
  for all $(d_\mathbf{y}, l_\mathbf{y}) \in O$
    if $\exists (d_\mathbf{z}, l_\mathbf{z}) \in M$ **such that** $l_\mathbf{z} \neq l_\mathbf{y} \wedge d_\mathbf{z} \not<^A d_\mathbf{y}$
      if $\mathrm{ub}(l_\mathbf{y}) - \mathrm{lb}(l_\mathbf{y}) < k - \sigma_k$ **then** $\mathrm{ub}(l_\mathbf{y}) \leftarrow \mathrm{ub}(l_\mathbf{y}) + 1$
    end if
  end for
end if
for all $l \in L$
  if $\mathrm{ub}(l) = 0$ **then** $L \leftarrow L \setminus \{l\}$
end for
if $(|L| = 1$ **or** $k = 1)$ **then return** L
for all $l \in L$
  $\mu \leftarrow \min(k, \sum_{l' \neq l} \mathrm{ub}(l'))$
  $\mathrm{lb}(l) \leftarrow \max(\mathrm{lb}(l), k - \mu)$
end for
$\tau \leftarrow \lceil \frac{k}{\min(k, |L|)} \rceil$
return $\{l \in L \mid \mathrm{ub}(l) \geq \tau,\ \forall l' \in L \setminus \{l\}.\ \mathrm{s}[l] \not<^{\mathbb{I}} \mathrm{s}[l']\}$

Theorem 4.4. *The abstract classifier C^A_{D,k,δ^A} described by Algorithm 4.4 is a sound approximation of $C_{D,k,\delta}$, namely, given an abstract value $a \in A$:*

$$\bigcup_{\mathbf{x}' \in \gamma^A(a)} C_{D,k,\delta}(\mathbf{x}') \subseteq C^A_{D,k,\delta^A}(a) \tag{4.2}$$

Proof. We prove that all the steps in Algorithm 4.4 are sound and no label returned by $C_{D,k,\delta}$ is forgotten. By definition of ordered structure, extracting the first k pairs from O means taking the k feature vectors that are most likely to be considered. If their distances are all strictly dominated by those in O, then these represent indeed the k closest samples and we can assign a vote to both their lower and upper bounds. If this is not the case, and therefore there is at least one distance that does not strictly dominate one of these, only if this distance is associated with a differently labeled feature vector does it make sense to give a vote to the upper bound. In fact, if there is an uncertainty between two or more identical labels, letting one of these out among the k closest to bring in another identical label does not change the score for that label. When a distance in O strictly dominates all the first k distances, the following ones will do the same by definition of O, and it is not necessary to continue iterating over them. The same is then done from the point of view of the pairs still in O: if there is a pair outside O that is not strictly dominated by a distance in O, and the relative sample has a different label, the latter can receive a vote to its upper bound, as this could potentially be considered, but we do not know it for sure due to incompleteness. Increasing the upper bound of a score by a quantity $m > 0$ is always a sound operation. Then, discarding all labels whose score upper bound is less then $\tau \triangleq \lceil \frac{k}{\min(k,|L|)} \rceil$ is also sound because being below such a threshold implies that there is at least another label with a strictly dominant score, as τ represents the maximum vote in the case of equal vote distribution. Finally, discarding the labels whose score is not maximal is sound by definition of kNN.
Although the computation of the distances may be complete, that of the bounds is not. This is due to the non-relationality of the interval domain, which does not allow to identify and manage relations between scores. One of the main relational properties that cannot be expressed with this domain is the mutual exclusion: if the label $l_{\mathbf{x}}$ is considered n times, then the label $l_{\mathbf{y}}$ is considered $m-n$ times, which cannot be abstracted precisely using real intervals as we lose the fact that the score of $l_{\mathbf{y}}$ depends on that of $l_{\mathbf{x}}$. Since all these steps are sound, namely, no relevant label is discarded, then so is the abstract classifier C^A_{D,k,δ^A} combining them all. Hence, $\bigcup_{\mathbf{x}' \in \gamma^A(a)} C_{D,k,\delta}(\mathbf{x}') \subseteq C^A_{D,k,\delta^A}(a)$ holds. □

4.2.5 Illustrative Example

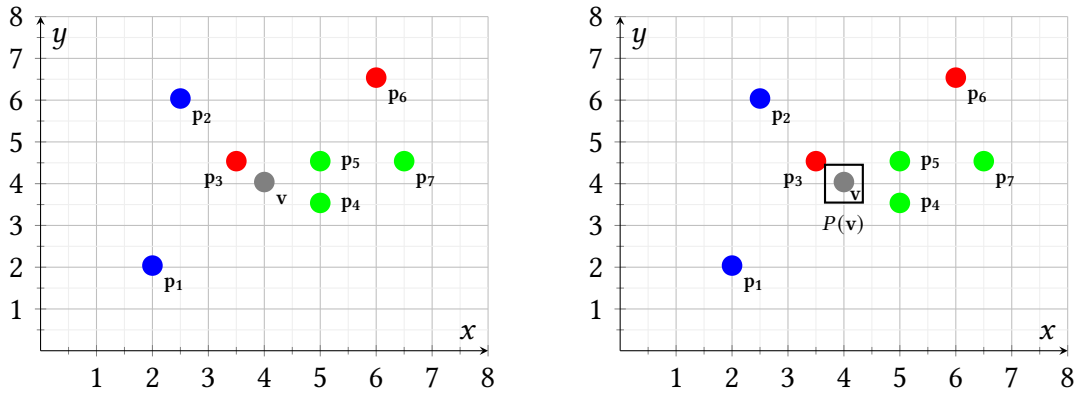

Figure 4.2: Abstract kNN classification on a toy dataset with 3 classes.

Figure 4.2 shows a toy dataset D of vectors in $\mathbb{R}^2$ consisting of points $\mathbf{p_1}, ..., \mathbf{p_7}$ having three labels: *red*, *green* and *blue*. We consider an unlabeled point $\mathbf{v} = (4, 4)$ and an adversarial region $P_\infty^{0.5}(\mathbf{v}) \triangleq \{\mathbf{w} \in \mathbb{R}^2 \mid \max(|\mathbf{w}_1 - \mathbf{v}_1|, |\mathbf{w}_2 - \mathbf{v}_2|) \leq 0.5\}$, which is the ℓ_∞ ball of radius 0.5 centered in $\mathbf{v}$. This perturbation can be exactly represented with the interval domain $\mathbb{I}$ as $a = \langle [3.5, 4.5], [3.5, 4.5] \rangle \in \mathbb{I}^2$. For the Euclidean distance η, the abstract distances between each (abstracted) point and a, computed in $\mathbb{I}$, are as follows:

$$
\begin{aligned}
\eta^{2^{\mathbb{I}}}(a, \mathbf{p_1}^{\mathbb{I}}) &= ([3.5, 4.5] -^{\mathbb{I}} [2, 2])^{2^{\mathbb{I}}} +^{\mathbb{I}} ([3.5, 4.5] -^{\mathbb{I}} [2, 2])^{2^{\mathbb{I}}} \\
&= [1.5, 2.5]^{2^{\mathbb{I}}} +^{\mathbb{I}} [1.5, 2.5]^{2^{\mathbb{I}}} \\
&= [2.25, 6.25] +^{\mathbb{I}} [2.25, 6.25] \\
&= [4.5, 12.5] \\
\eta^{2^{\mathbb{I}}}(a, \mathbf{p_2}^{\mathbb{I}}) &= ([3.5, 4.5] -^{\mathbb{I}} [2.5, 2.5])^{2^{\mathbb{I}}} +^{\mathbb{I}} ([3.5, 4.5] -^{\mathbb{I}} [6, 6])^{2^{\mathbb{I}}} \\
&= [1, 2]^{2^{\mathbb{I}}} +^{\mathbb{I}} [2, 3]^{2^{\mathbb{I}}} \\
&= [1, 4] +^{\mathbb{I}} [4, 9] \\
&= [5, 13] \\
\eta^{2^{\mathbb{I}}}(a, \mathbf{p_3}^{\mathbb{I}}) &= ([3.5, 4.5] -^{\mathbb{I}} [3.5, 3.5)^{2^{\mathbb{I}}} +^{\mathbb{I}} ([3.5, 4.5] -^{\mathbb{I}} [4.5, 4.5])^{2^{\mathbb{I}}} \\
&= [0, 1]^{2^{\mathbb{I}}} +^{\mathbb{I}} [-1, 0]^{2^{\mathbb{I}}} \\
&= [0, 1] +^{\mathbb{I}} +[0, 1] \\
&= [0, 2]
\end{aligned}
$$

$$
\begin{aligned}
\eta^{2^{\mathbb{I}}}(a, \mathbf{p_4}^{\mathbb{I}}) &= ([3.5, 4.5] -^{\mathbb{I}} [5, 5])^{2^{\mathbb{I}}} +^{\mathbb{I}} ([3.5, 4.5] -^{\mathbb{I}} [4.5, 4.5])^{2^{\mathbb{I}}} \\
&= [-1.5, -0.5]^{2^{\mathbb{I}}} +^{\mathbb{I}} [-1, 0]^{2^{\mathbb{I}}} \\
&= [0.25, 2.25] +^{\mathbb{I}} [0, 1] \\
&= [0.25, 3.25] \\
\eta^{2^{\mathbb{I}}}(a, \mathbf{p_5}^{\mathbb{I}}) &= ([3.5, 4.5] -^{\mathbb{I}} [5, 5])^{2^{\mathbb{I}}} +^{\mathbb{I}} ([3.5, 4.5] -^{\mathbb{I}} [3.5, 3.5])^{2^{\mathbb{I}}} \\
&= [-1.5, -0.5]^{2^{\mathbb{I}}} +^{\mathbb{I}} [0, 1]^{2^{\mathbb{I}}} \\
&= [0.25, 2.25] +^{\mathbb{I}} [0, 1] \\
&= [0.25, 3.25] \\
\eta^{2^{\mathbb{I}}}(a, \mathbf{p_6}^{\mathbb{I}}) &= ([3.5, 4.5] -^{\mathbb{I}} [6, 6])^{2^{\mathbb{I}}} +^{\mathbb{I}} ([3.5, 4.5] -^{\mathbb{I}} [6.5, 6.5])^{2^{\mathbb{I}}} \\
&= [-2.5, -1.5]^{2^{\mathbb{I}}} +^{\mathbb{I}} [-3, -2]^{2^{\mathbb{I}}} \\
&= [2.25, 5.25] +^{\mathbb{I}} [4, 9] \\
&= [6.25, 15.25] \\
\eta^{2^{\mathbb{I}}}(a, \mathbf{p_7}^{\mathbb{I}}) &= ([3.5, 4.5] -^{\mathbb{I}} [6.5, 6.5])^{2^{\mathbb{I}}} +^{\mathbb{I}} ([3.5, 4.5] -^{\mathbb{I}} [4.5, 4.5])^{2^{\mathbb{I}}} \\
&= [-3, -2]^{2^{\mathbb{I}}} +^{\mathbb{I}} [-1, 0]^{2^{\mathbb{I}}} \\
&= [4, 9] +^{\mathbb{I}} [0, 1] \\
&= [4, 10]
\end{aligned}
$$

Let us observe that $\mathbf{p_3}$, $\mathbf{p_4}$ and $\mathbf{p_5}$ are the closest points to a, and thus to $P^{0.5}_{\infty}(\mathbf{v})$, as their intervals are strictly dominated by the other intervals, namely, $\eta^{2^{\mathbb{I}}}(a, \mathbf{p_i}^{\mathbb{I}}) \leq^{\mathbb{I}} \eta^{2^{\mathbb{I}}}(a, \mathbf{p_j}^{\mathbb{I}})$ for all $i \in \{3, 4, 5\}, j \in \{1, 2, 6, 7\}$. It follows that these are the first 3 distances in the ordered structure. However, we do not know their relative dominance order, because these intervals overlap, e.g. $[0, 2] \not\leq [0.25, 3.25]$ and $[0.25, 3.25] \not\leq [0, 2]$.

Therefore, for $k = 1$ both labels *red* and *green* of $\mathbf{p_3}$, $\mathbf{p_4}$, $\mathbf{p_5}$ will get a score interval of $[0, 1]$, namely, $\mathrm{s}[red] = \mathrm{s}[green] = [0, 1]$, meaning that these labels may or may not be considered. This implies that $C^{\mathbb{I}}_{D,1,\eta^{\mathbb{I}}}(a) = \{red, green\}$, and we cannot infer that 1NN is robust on $P^{0.5}_{\infty}(\mathbf{v})$.

However, for $k = 3$ the robustness verification turns out to be complete, because the 3 samples $\mathbf{p_3}$, $\mathbf{p_4}$, $\mathbf{p_5}$ will all be considered, being $3 \leq k$, and will have the same weight regardless of their concrete values, so that the labels *red* and *green* get, respectively, the score intervals $[1, 1]$ and $[2, 2]$, i.e., $\mathrm{s}[red] = [1, 1]$ and $\mathrm{s}[green] = [2, 2]$. Thus, $C^{\mathbb{I}}_{D,3,\eta^{\mathbb{I}}}(a) = \{green\}$, and we can therefore infer that 3NN is robust on $P^{0.5}_{\infty}(\mathbf{v})$.

4.3 Dealing with Categorical Features

So far we have considered only numerical features, but some datasets may contain both numerical and categorical features. A categorical feature can assume an alphanumeric value belonging to a certain set of alphanumeric values, e.g. $gender \in \{male, female\}$, and cannot be processed by machine learning models based on proximity or any other parameter that needs a mathematical computation. To overcome this limitation, well-known strategies for converting categorical features to numerical values exist, the most prominent of which is *one-hot encoding*. By applying one-hot encoding every categorical feature having c possible values is substituted by a set of c numerical ones $\langle n_1, n_2 \dots n_c \rangle$ with $n_i \in \{0, 1\}$ for all $i \in [1, c]$. This operation implicitly introduces a constraint on the values of the features: if a sample has a categorical feature whose one-hot representation is spread from index i to index j, then $\sum_{m=i}^{j} x_m = 1$ must hold, otherwise a single sample will have $\sum_{m=i}^{j} x_m$ different categories at the same time. If this constraint is not honored by the abstract domain of choice, a *loss of precision* can occur as illustrated by the following example.

Example 4.2 (Loss of Precision due to One-Hot Encoding). Suppose we have a dataset D describing two features, that is, $color \in \{red, green, blue\}$ and $price \in \mathbb{R}$, and contains two vectors $\mathbf{v} = \langle red, 1 \rangle, \mathbf{w} = \langle red, 3 \rangle$, labeled as -1 and $+1$, respectively. A one-hot encoding generates features $isRed$, $isGreen$, $isBlue \in \{0, 1\}$, and converts points to $\mathbf{v}' \triangleq \langle 1, 0, 0, 1 \rangle, \mathbf{w}' \triangleq \langle 1, 0, 0, 3 \rangle$. An adversarial region R is built such that every point within it has $price \in [0, 1]$, and any color is allowed: $R = \{(r, g, b, price) \mid r, g, b \in \{0, 1\}, price \in [0, 1]\}$. We observe that $\mathbf{v}'$ is always closer than $\mathbf{w}'$ to any point $\mathbf{x}$ in R:

$$
\begin{aligned}
& \delta_p(\mathbf{v}', \mathbf{x}) < \delta_p(\mathbf{w}', \mathbf{x}) \\
\Leftrightarrow\ & \sqrt[p]{|\mathbf{v}'_1 - x_1|^p + |\mathbf{v}'_2 - x_2|^p + |\mathbf{v}'_3 - x_3|^p + |\mathbf{v}'_4 - x_4|^p} \\
<\ & \sqrt[p]{|\mathbf{w}'_1 - \mathbf{x}_1|^p + |\mathbf{w}'_2 - \mathbf{x}_2|^p + |\mathbf{w}'_3 - \mathbf{x}_3|^p + |\mathbf{w}'_4 - \mathbf{x}_4|^p} \\
\Leftrightarrow\ & \sqrt[p]{|1 - \mathbf{x}_1|^p + |-\mathbf{x}_2|^p + |-\mathbf{x}_3|^p + |1 - \mathbf{x}_4|^p} \\
<\ & \sqrt[p]{|1 - \mathbf{x}_1|^p + |-\mathbf{x}_2|^p + |-\mathbf{x}_3|^p + |3 - \mathbf{x}_4|^p} \\
\Leftrightarrow\ & \sqrt[p]{|1 - \mathbf{x}_1|^p + \mathbf{x}_2^p + \mathbf{x}_3^p + |1 - \mathbf{x}_4|^p} < \sqrt[p]{|1 - \mathbf{x}_1|^p + \mathbf{x}_2^p + \mathbf{x}_3^p + |3 - \mathbf{x}_4|^p} \\
\Leftrightarrow\ & |1 - \mathbf{x}_4| < |3 - \mathbf{x}_4|
\end{aligned}
$$

which is always satisfied by $\mathbf{x}_4 \in [0,1]$, hence a 1NN would label any point in R as -1. By running an abstract verification using the hyperrectangle domain and Manhattan distance, R is exactly abstracted as $\langle [0,1], [0,1], [0,1], [0,1] \rangle$ and abstract distances from $\mathbf{v}', \mathbf{w}'$ are $[0,1] + [0,1] + [0,1] + [0,1] = [0,4]$ and $[0,1] + [0,1] + [0,1] + [2,3] = [2,6]$, respectively. Since distance intervals overlap, it is not possible to infer which point is the closest, and therefore the abstract classifier returns $\{-1, +1\}$. This issue is caused by the hyperrectangle abstraction not being able to represent the constraints $\mathit{isRed}, \mathit{isGreen}, \mathit{isBlue} \in \{0,1\}$ and $\mathit{isRed} + \mathit{isGreen} + \mathit{isBlue} = 1$. □

To avoid loss of precision, we partition the original adversarial region R abstracted by $a \in A$ into c adversarial regions $R_i \subseteq R$ abstracted by $a_i \in A$, where c is the number of possible values of the categorical features involved in the perturbation, and execute the classification procedure for each adversarial region or until all labels are returned (at this point soundness is given by definition). Partitioning is done in such a way that each categorical feature of each component has exactly one possible value, thus eliminating the need for abstraction within the component. The classification is given by:

$$C^A_{D,k,\delta^A}(a) = \bigcup_{i \in [1,c]} C^A_{D,k,\delta^A}(a_i)$$

For instance, the partitions for the adversarial region R in Example 4.2 are:

$$R_1 = \{(1,0,0,\mathit{price}) \mid \mathit{price} \in [0,1]\},\ a_1 = \langle [1,1], [0,0], [0,0], [0,1] \rangle$$
$$R_2 = \{(0,1,0,\mathit{price}) \mid \mathit{price} \in [0,1]\},\ a_2 = \langle [0,0], [1,1], [0,0], [0,1] \rangle$$
$$R_3 = \{(0,0,1,\mathit{price}) \mid \mathit{price} \in [0,1]\},\ a_3 = \langle [0,0], [0,0], [1,1], [0,1] \rangle$$

Thus, by computing abstract distances for a_1 we obtain:

$$\mu^{\mathbb{I}}(\mathbf{v}', a_1) = [0,0] + [0,0] + [0,0] + [0,1] = [0,1]$$
$$\mu^{\mathbb{I}}(\mathbf{w}', a_1) = [0,0] + [0,0] + [0,0] + [2,3] = [2,3]$$

yielding $C^{\mathbb{I}}_{D,k,\mu^{\mathbb{I}}}(a_1) = \{-1\}$. The same can be done for a_2 and a_3, obtaining:

$$\mu^{\mathbb{I}}(\mathbf{v}', a_2) = \mu^{\mathbb{I}}(\mathbf{v}', a_2) = [2,3]$$
$$\mu^{\mathbb{I}}(\mathbf{w}', a_2) = \mu^{A}(\mathbf{w}', a_2) = [4,5]$$

hence $C^{\mathbb{I}}_{D,k,\mu^{\mathbb{I}}}(a_2) = C^{\mathbb{I}}_{D,k,\mu^{\mathbb{I}}}(a_3) = \{-1\}$. Then, by collecting all these sets, we finally get $C^{\mathbb{I}}_{D,k,\mu^{\mathbb{I}}}(a) = C^{\mathbb{I}}_{D,k,\mu^{\mathbb{I}}}(a_1) \cup C^{\mathbb{I}}_{D,k,\mu^{\mathbb{I}}}(a_2) \cup C^{\mathbb{I}}_{D,k,\mu^{\mathbb{I}}}(a_3) = \{-1\}$, proving robustness and consequently stability.

5 Experimental Evaluation

We implemented our formal verification method for k-nearest neighbor classifiers in a tool called kNAVe (*KNN Abstract Verifier*), whose name literally means "an unprincipled and dishonest person" and is a pun on our verification of adversarial perturbations caused by an attacker. It has been written entirely in Python and is available open source on the author's GitHub page [12], together with all the datasets and results. Our experimental results were run on an Ubuntu 22.04.1 LTS with AMD Ryzen 5 3500U 2.1GHz CPU, and show that kNAVe is fast and scalable, with significant high provable percentages in most of the datasets.

This chapter will present the findings of our experimental evaluations of *robustness*, *stability*, and *individual fairness*, as well as comparisons with the GeoAdEx tool by Sitawarin et al. [38].

5.1 Datasets

For our experiments, we considered 7 standard datasets used in the most recent stream of works studying the robustness of kNNs [45, 44, 46, 39, 41, 38], on which we also carried out stability tests, and 4 datasets for fairness verification of deep neural networks [21].

The datasets used for the formal verification of robustness and stability properties are taken from LIBSVM [6]. They are:

- **Australian**: dataset concerning credit card applications, consisting of categorical and numerical features. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.
- **BreastCancer**: classic and very easy binary classification dataset for diagnostic purpose, where the classification is whether the cancer is benign or malignant.
- **Diabetes**: dataset taken from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict, based on diagnostic measurements, whether a patient has diabetes.
- **Fourclass**: non-linear separable dataset consisting of 862 two-dimension data points.
- **Letter**: letter recognition dataset, where the objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet.
- **Pendigits**: digit database created by collecting 250 samples from 44 writers. The samples written by 30 writers are used for training and the digits written by the other 14 are used for testing.
- **Satimage**: database consisting of the multi-spectral values of pixels in 3×3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The aim is to predict this classification, given the multi-spectral values.

Those for individual fairness verification are instead:

- **Adult**: dataset extracted from the 1994 US Census database. Every sample assigns a yearly income (below or above 50K US$) to an individual based on personal attributes such as gender, race, and occupation.
- **Compas**: dataset containing data collected on the use of the COMPAS risk assessment tool in Broward County, FL. Each sample predicts the risk of recidivism for individuals based on personal attributes and criminal history.
- **Crime**: dataset containing socio-economic, law enforcement, and crime data for communities within the US. Each sample indicates whether a community is above or below the median number of violent crimes per population.

- **German**: dataset containing individuals with either a good or bad credit score.

The main characteristics of these datasets, as well as their accuracy, are summarized in Table 5.1 and Table 5.2, respectively.

Name	#training	#test	#features	#features (one-hot)	#classes
Australian	483	207	14	39	2
BreastCancer	479	204	10	10	2
Diabetes	556	230	8	8	2
Fourclass	604	258	2	2	2
Letter	15000	5000	16	16	26
Pendigits	7494	3498	16	16	10
Satimage	4435	2000	36	46	6
Adult	30162	15060	14	103	2
Compas	4222	1056	10	370	2
Crime	1595	399	102	147	2
German	800	200	20	56	2

Table 5.1: Datasets used in experimental evaluations.

	Accuracy (%)			
Name	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Australian	77.8	80.2	82.6	82.6
BreastCancer	92.6	94.6	93.6	93.6
Diabetes	70.9	72.2	70.0	71.3
Fourclass	100	100	100	100
Letter	95.7	94.6	94.2	94.3
Pendigits	97.7	97.8	97.5	97.5
Satimage	88.8	90.3	89.5	90.1
Adult	78.4	81.1	81.8	82.3
Compas	58.4	59.1	60.2	61.1
Crime	77.7	79.7	80.7	81.2
German	73.0	71.5	74.5	77.0

Table 5.2: Accuracy of each dataset for $k \in \{1, 3, 5, 7\}$.

5.2 Perturbations

We instantiated our parametric abstract kNN classifier C^{A}_{D,k,δ^A} to the interval abstraction $\mathbb{I}$, and we considered the ℓ_2-norm perturbation discussed in Subsection 2.2.3 for our robustness and stability experiments, with a magnitude ϵ ranging in $[0.001, 0.05]$, i.e., numerical features can be altered from $\pm 0.1\%$ to $\pm 5\%$. Individual fairness for Adult, Compas, Crime, and German datasets has been evaluated for the Noise-Cat similarity relation defined in Subsection 2.2.5, where we applied a ℓ_2-norm Noise perturbation to numerical features, also in this case with ϵ ranging in $[0, 0.05]$, and a Cat perturbation for the following categorical sensible features: for Adult and German: *gender*, having 2 categories; for Compas: *race*, having 2 categories; for Crime: *state*, having 46 categories.

5.3 Results

First, we preprocessed each dataset, in accordance with Ruoss et al. [33], as follows: (i) rows and columns with missing values are dropped; (ii) for fairness datasets, numerical features have been normalized to zero mean and unit variance; (iii) when needed, datasets are split into training ($\approx$ 70-80%) and test ($\approx$ 20-30%) sets. Letter, Pendigits and Satimage are the only datasets having an independent test set. (iv) categorical features are one-hot encoded; (v) numerical features are scaled to $[0, 1]$.
Thus, if the space X consists of n-dimensional real vectors scaled to $[0, 1]$, an ℓ_2-norm perturbation having magnitude $\epsilon \geq 0$ on a feature vector $\mathbf{x} \in X$ defines the adversarial region:

$$P^{\epsilon}_{\infty}(\mathbf{x}) \triangleq \{\mathbf{w} \in \mathbb{R}^n \mid \forall i \in [1, n].\ \mathbf{w}_i \in [\mathbf{x}_i - \epsilon, \mathbf{x}_i + \epsilon] \sqcap^{\mathbb{I}} [0, 1]\}$$

where $\sqcap^{\mathbb{I}}$ is applied only when $\gamma^{\mathbb{I}}([0, 1])$ includes all possible values for $\mathbf{x}_i$. For instance, if we have a pixel ranging in $[0, 255]$ that is scaled between 0 and 1, where the latter represent, respectively, 0 e 255, it makes no sense to have a perturbation that results in values less than 0 or greater than 1, as they correspond to values outside $[0, 255]$. For our datasets, we applied $\sqcap^{\mathbb{I}}$ to all the features (scaled in $[0, 1]$) of Letter and Pendigits, as their feature range is known, i.e., $[0, 15]$ for Letter and $[0, 100]$ for Pendigits, and holds for all of them.

Due incompleteness, the results are intended to be lower bounds of the real ones.

Table 5.3 shows the average verification time highlighting average times per sample and for the whole test set, in milliseconds and seconds, respectively. These runtimes refer to the execution of kNAVe on all $k \in \{1, 3, 5, 7\}$, and coincide with the time needed to obtain a row of Subsection 5.3.1 and Subsection 5.3.2 tables. We remark that once the distances for a test sample have been computed, kNN can exploit such distances for each value of k, without having to re-compute them as k changes. This results in a considerable saving of time in using kNAVe with multiple values of k at the same time.

Name	Avg. Verif. Time per Sample (ms)	Avg. Verif. Time per ϵ (s)
Australian	19	4
BreastCancer	13	3
Diabetes	13	3
Fourclass	4	1
Letter	399	1993
Pendigits	197	688
Satimage	239	478
Adult	3497	52667
Compas	1597	1686
Crime	3589	1432
German	80	16

Table 5.3: Average verification time.

The results for stability, robustness and individual fairness are shown in the following two subsections. As we will see, in many datasets, we are able to infer excellent stability percentages with $\epsilon < 0.02$, even greater than 90% in some of these. The percentages of robustness are obviously lower than those of stability, as a correct label is required; however, this does not depend on the quality of the abstract classifier because the label is entirely determined by the data distribution. For individual fairness the percentages will tend to decrease linearly as ϵ grows, but not all datasets turn out to be fair with respect to the chosen category; in fact with $\epsilon = 0$, meaning no Noise perturbation is applied, such percentages are already too small. Let us stress that experiments with $\epsilon = 0$ involve the solely application of a Cat perturbation to the chosen categorical features, leaving numerical features unaltered: this makes our verification method complete as stated in Section 4.3.

5.3.1 Stability and Robustness Verification

Tables below reports the percentages of provable stability and robustness for all the test samples ranging in each ground truth test dataset T, namely, $\textsc{Stability}(C^{\mathbb{I}}_{D,k,\delta^{\mathbb{I}}}, T)$ and $\textsc{Robustness}(C^{\mathbb{I}}_{D,k,\delta^{\mathbb{I}}}, T)$ metrics defined in (2.2). We performed our experiments with $k \in \{1, 3, 5, 7\}$, and, following the standard practice for the kNN algorithm, we avoided even values of k as they are more likely to introduce tie votes in the classification.

Australian Provable Stability (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	100.0	100.0	100.0	100.0
0.002	99.0	99.5	100.0	100.0
0.005	97.1	98.6	97.6	99.0
0.01	95.2	96.6	96.1	97.1
0.02	92.8	91.3	91.8	94.7
0.03	91.3	88.9	89.4	92.3
0.04	88.4	85.0	87.9	88.9
0.05	86.0	84.1	85.5	86.0

Australian Provable Robustness (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	77.8	80.2	82.6	82.6
0.002	77.8	79.7	82.6	82.6
0.005	76.8	79.2	81.2	81.6
0.01	75.4	77.8	79.7	80.7
0.02	73.9	75.8	77.8	79.2
0.03	72.9	74.4	76.8	77.8
0.04	70.5	72.5	76.3	75.4
0.05	68.6	71.5	74.4	73.9

Table 5.4: Provable stability and robustness on the whole Australian test set.

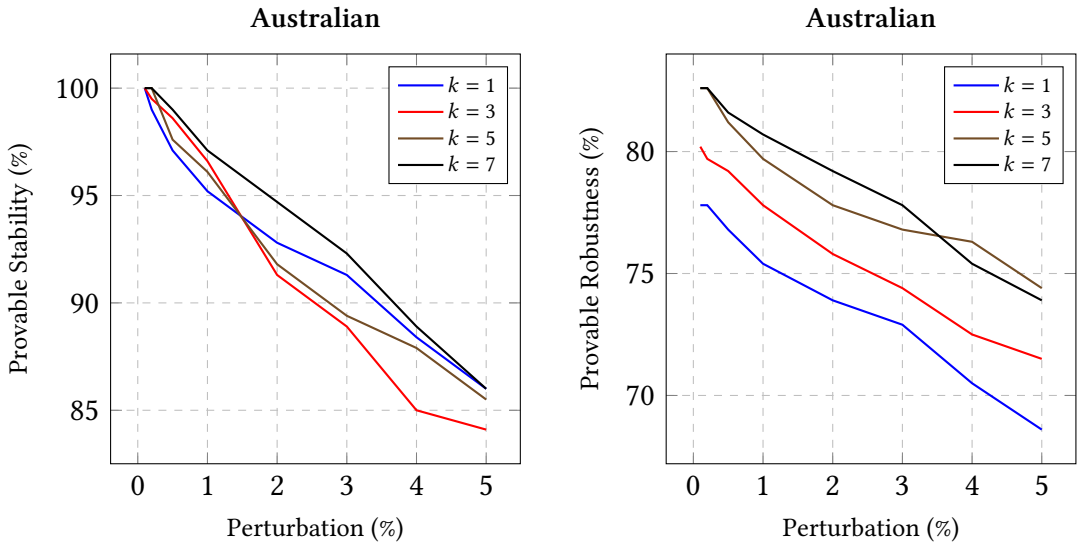

Figure 5.1: Provable stability and robustness on the whole Australian test set.

BreastCancer Provable Stability (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	99.5	100.0	99.0	99.5
0.002	98.5	100.0	99.0	99.0
0.005	96.6	99.5	98.0	98.5
0.01	96.6	99.0	96.6	97.1
0.02	92.2	93.1	91.7	93.1
0.03	86.8	88.7	87.7	88.2
0.04	80.4	83.3	83.3	85.3
0.05	75.0	77.9	81.4	83.8

BreastCancer Provable Robustness (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	92.6	94.6	93.6	93.1
0.002	92.2	94.6	93.6	92.6
0.005	91.2	94.1	93.1	92.6
0.01	91.2	93.6	91.7	92.2
0.02	88.2	89.7	88.7	89.7
0.03	84.3	86.8	86.8	87.3
0.04	78.9	82.4	82.4	84.8
0.05	74.5	77.5	80.9	83.3

Table 5.5: Provable stability and robustness on the whole BreastCancer test set.

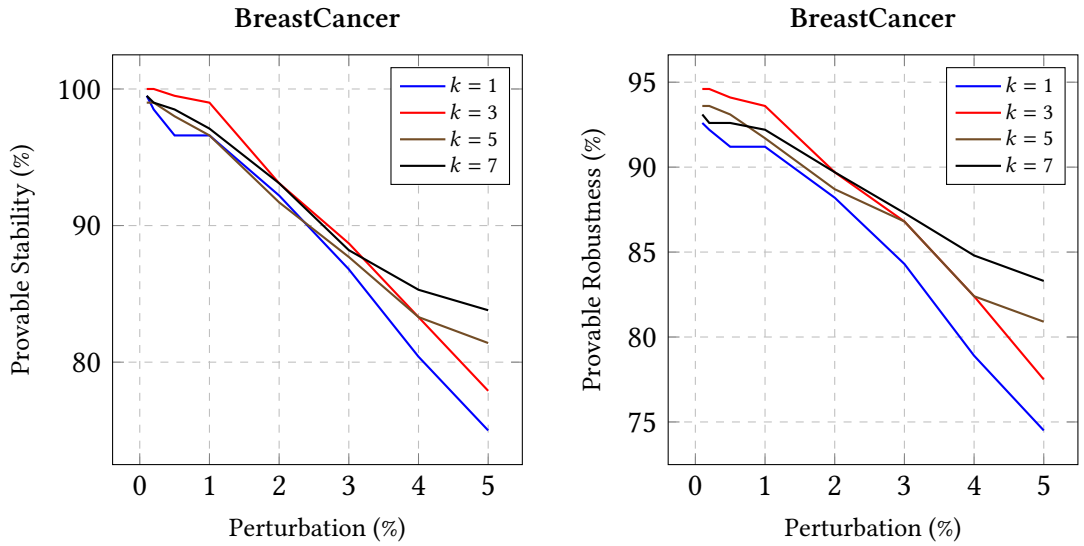

Figure 5.2: Provable stability and robustness on the whole BreastCancer test set.

Diabetes Provable Stability (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	95.2	96.1	98.7	98.7
0.002	89.1	91.3	94.8	94.3
0.005	78.7	80.4	80.0	77.4
0.01	62.6	59.6	56.5	54.3
0.02	31.3	25.7	23.9	23.0
0.03	13.0	9.1	7.8	6.1
0.04	7.8	4.3	4.8	3.0
0.05	3.0	1.7	0.9	0.9

Diabetes Provable Robustness (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	69.1	71.3	69.6	71.3
0.002	65.7	68.7	67.4	69.6
0.005	60.0	62.2	61.3	59.1
0.01	49.1	49.6	47.0	46.5
0.02	25.2	23.0	22.6	21.7
0.03	11.7	8.7	7.4	6.1
0.04	7.4	4.3	4.3	3.0
0.05	2.6	1.7	0.9	0.9

Table 5.6: Provable stability and robustness on the whole Diabetes test set.

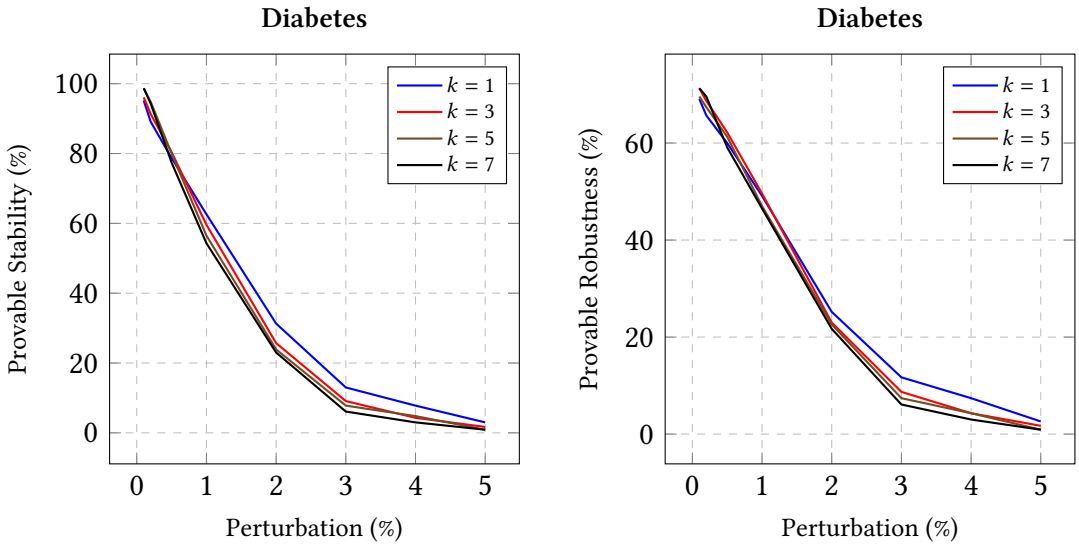

Figure 5.3: Provable stability and robustness on the whole Diabetes test set.

Fourclass Provable Stability (%)				
ϵ	$k = 1$	$k = 3$	$k = 5$	$k = 7$
0.001	100.0	100.0	100.0	100.0
0.002	99.6	99.6	100.0	100.0
0.005	99.6	99.6	99.2	98.8
0.01	99.2	99.6	98.4	97.7
0.02	93.4	91.5	86.8	85.3
0.03	78.3	75.6	75.6	72.9
0.04	62.0	60.5	59.3	55.8
0.05	45.0	42.6	40.3	37.6

Fourclass Provable Robustness (%)				
ϵ	$k = 1$	$k = 3$	$k = 5$	$k = 7$
0.001	100.0	100.0	100.0	100.0
0.002	99.6	99.6	100.0	100.0
0.005	99.6	99.6	99.2	98.8
0.01	99.2	99.6	98.4	97.7
0.02	93.4	91.5	86.8	85.3
0.03	78.3	75.6	75.6	72.9
0.04	62.0	60.5	59.3	55.8
0.05	45.0	42.6	40.3	37.6

Table 5.7: Provable stability and robustness on the whole Fourclass test set.

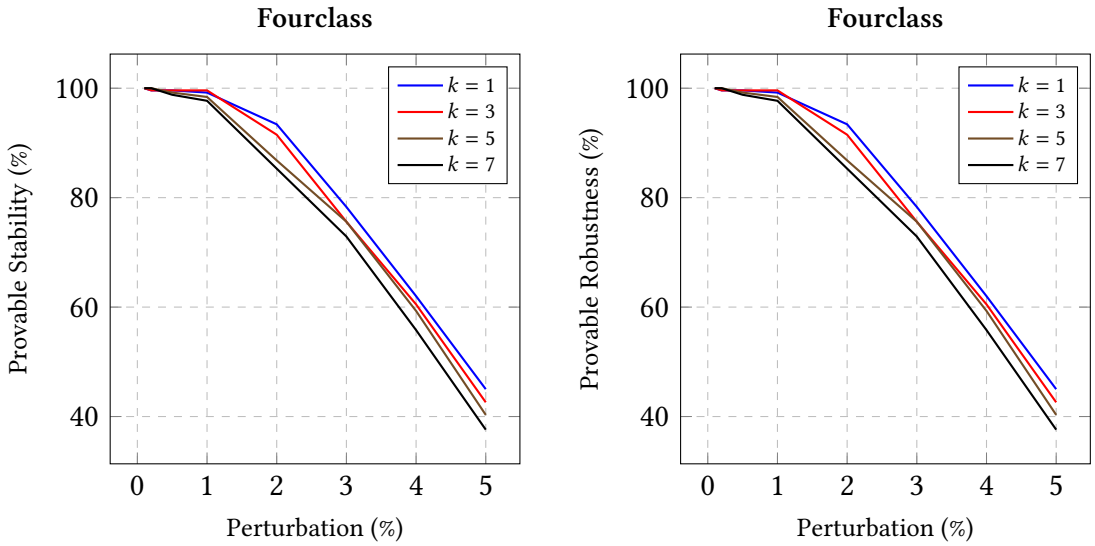

Figure 5.4: Provable stability and robustness on the whole Fourclass test set.

Letter Provable Stability (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	98.0	96.3	95.2	95.6
0.002	96.7	94.4	92.7	92.1
0.005	91.7	88.6	85.4	83.1
0.01	82.7	75.1	69.5	65.2
0.02	54.6	45.4	40.3	37.0
0.03	31.0	24.9	21.3	19.2
0.04	16.2	13.4	11.5	10.6
0.05	7.7	6.6	5.8	5.2

Letter Provable Robustness (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	94.7	93.6	92.9	93.0
0.002	93.9	92.3	91.1	90.5
0.005	90.4	87.8	84.8	82.7
0.01	82.2	75.0	69.4	65.1
0.02	54.6	45.4	40.3	37.0
0.03	31.0	24.9	21.3	19.2
0.04	16.2	13.4	11.5	10.6
0.05	7.7	6.6	5.8	5.2

Table 5.8: Provable stability and robustness on the whole Letter test set.

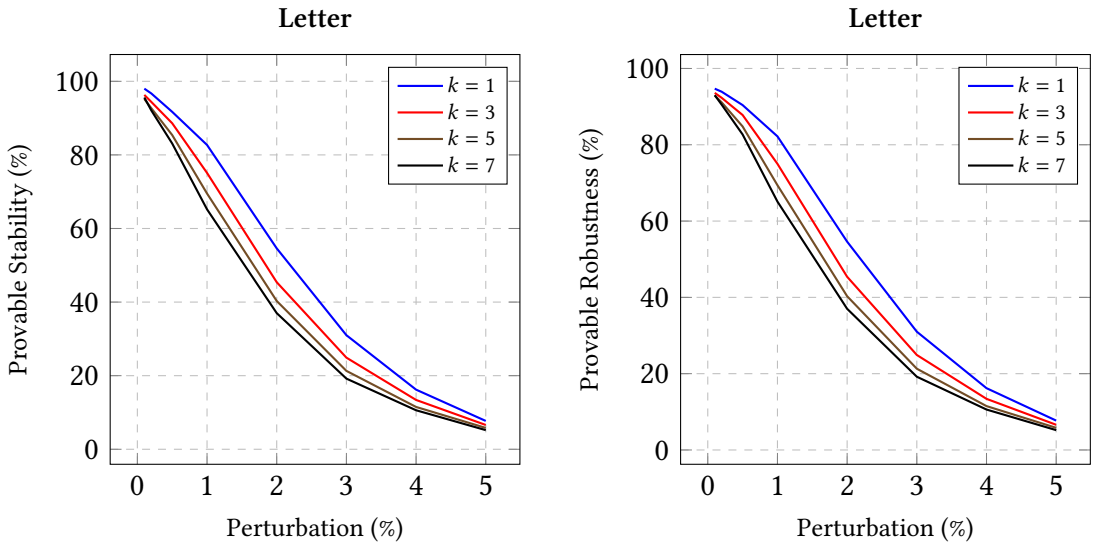

Figure 5.5: Provable stability and robustness on the whole Letter test set.

Pendigits Provable Stability (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	99.6	99.5	99.4	99.6
0.002	99.3	99.1	99.1	99.2
0.005	98.3	98.1	98.1	97.8
0.01	96.7	96.1	95.7	95.0
0.02	91.4	90.5	89.5	89.1
0.03	82.0	82.0	81.0	80.0
0.04	71.3	71.5	71.0	70.2
0.05	59.0	60.6	59.8	58.9

Pendigits Provable Robustness (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	97.6	97.5	97.4	97.4
0.002	97.4	97.5	97.3	97.2
0.005	96.8	96.8	96.7	96.3
0.01	95.6	95.3	94.9	94.1
0.02	90.9	90.2	89.2	88.7
0.03	81.7	81.7	80.8	79.8
0.04	71.2	71.4	71.0	70.2
0.05	58.9	60.6	59.7	58.9

Table 5.9: Provable stability and robustness on the whole Pendigits test set.

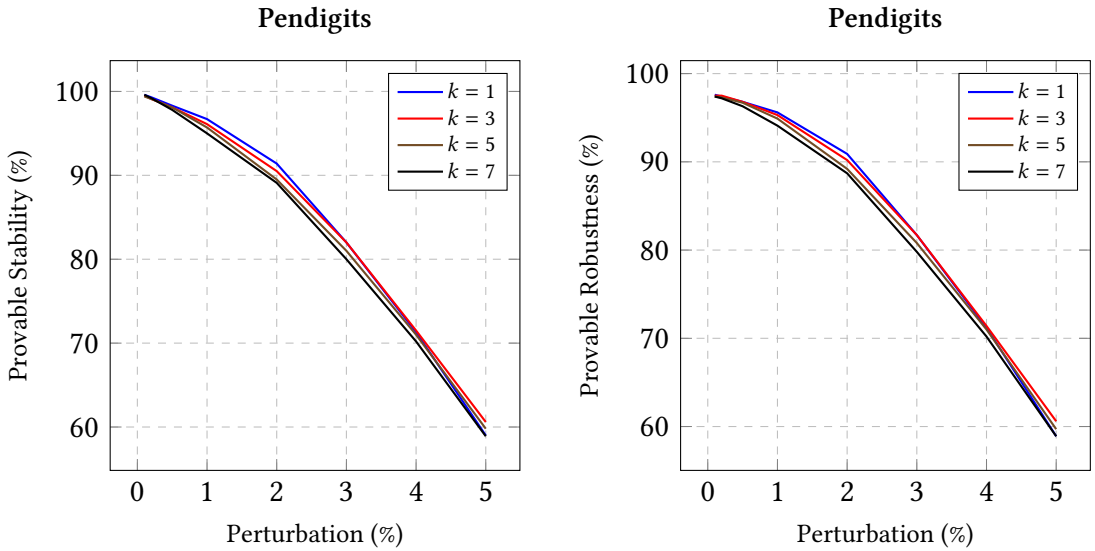

Figure 5.6: Provable stability and robustness on the whole Pendigits test set.

Satimage Provable Stability (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	93.8	93.8	93.7	94.3
0.002	88.7	88.9	88.9	88.1
0.005	73.7	72.9	72.4	72.4
0.01	56.0	54.9	54.0	54.2
0.02	31.4	31.1	31.8	32.8
0.03	19.3	18.6	18.3	18.6
0.04	12.6	12.6	12.6	12.7
0.05	9.6	9.7	9.7	9.8

Satimage Provable Robustness (%)				
ϵ	$k=1$	$k=3$	$k=5$	$k=7$
0.001	86.1	87.4	87.0	87.5
0.002	83.0	84.3	84.1	83.4
0.005	71.7	71.7	71.0	70.8
0.01	55.4	54.6	53.8	53.8
0.02	31.4	31.0	31.7	32.6
0.03	19.3	18.6	18.3	18.6
0.04	12.6	12.6	12.6	12.7
0.05	9.6	9.7	9.7	9.8

Table 5.10: Provable stability and robustness on the whole Satimage test set.

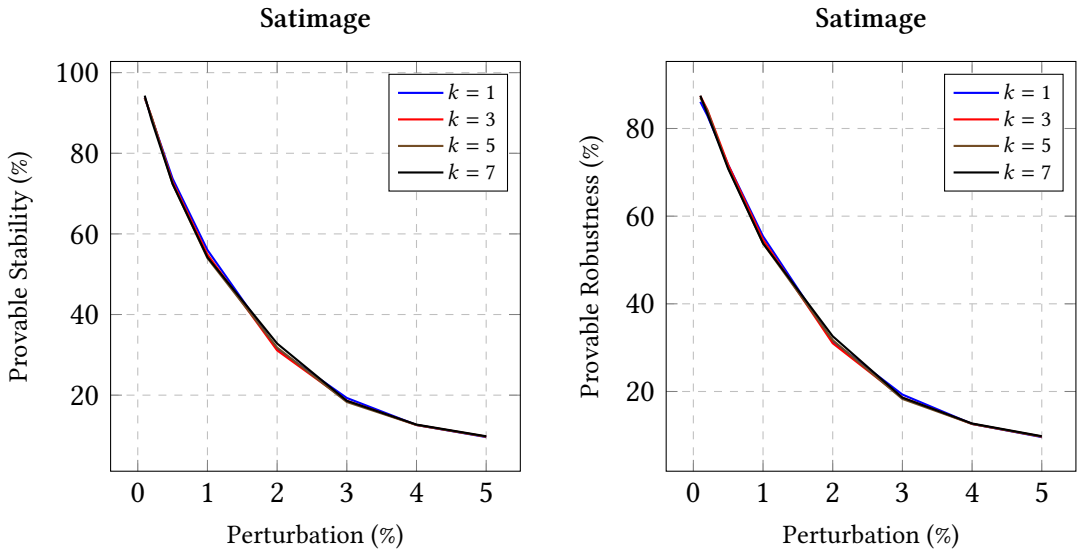

Figure 5.7: Provable stability and robustness on the whole Satimage test set.

5.3.2 Individual Fairness Verification

Tables below reports the percentages of provable individual fairness for all the test samples ranging in each ground truth test dataset T, namely, $\textsc{Fairness}(C^{\mathbb{I}}_{D,k,\delta^{\mathbb{I}}}, T)$ metric defined in (2.3). As for stability and robustness, we performed our experiments with $k \in \{1, 3, 5, 7\}$, thus avoiding even values of k as they are more likely to introduce tie votes in the classification.

Adult Provable Individual Fairness (%)				
ϵ	$k = 1$	$k = 3$	$k = 5$	$k = 7$
0	95.4	97.3	97.7	97.8
0.001	92.3	94.6	95.3	95.3
0.002	89.9	92.0	93.0	93.2
0.005	82.4	84.7	85.9	86.5
0.01	72.7	75.5	76.5	77.1
0.02	60.2	64.1	66.0	66.5
0.03	52.2	58.3	60.4	61.4
0.04	47.0	54.4	57.0	58.2
0.05	43.5	51.3	54.6	56.2

Table 5.11: Provable individual fairness on the whole Adult test set.

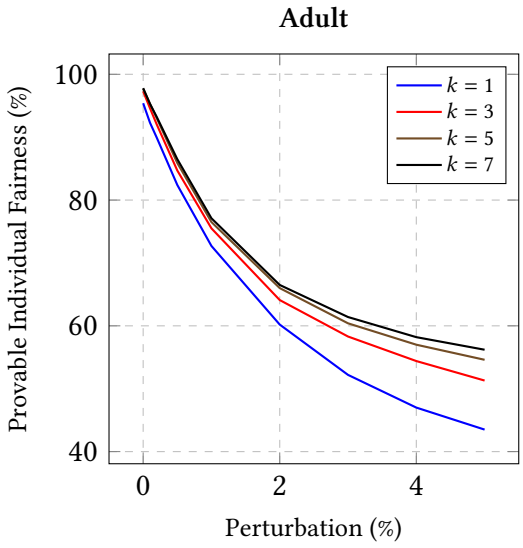

Figure 5.8: Provable individual fairness on the whole Adult test set.

Compas Provable Individual Fairness (%)				
ϵ	$k = 1$	$k = 3$	$k = 5$	$k = 7$
0	58.3	62.7	69.1	71.1
0.001	48.2	54.5	61.1	62.5
0.002	45.6	50.9	56.8	57.6
0.005	35.3	41.4	45.4	46.1
0.01	25.7	30.7	32.8	36.4
0.02	17.8	20.9	23.9	27.5
0.03	13.6	16.7	19.5	21.7
0.04	11.7	13.8	16.1	18.1
0.05	10.2	12.7	14.3	16.5

Table 5.12: Provable individual fairness on the whole Compas test set.

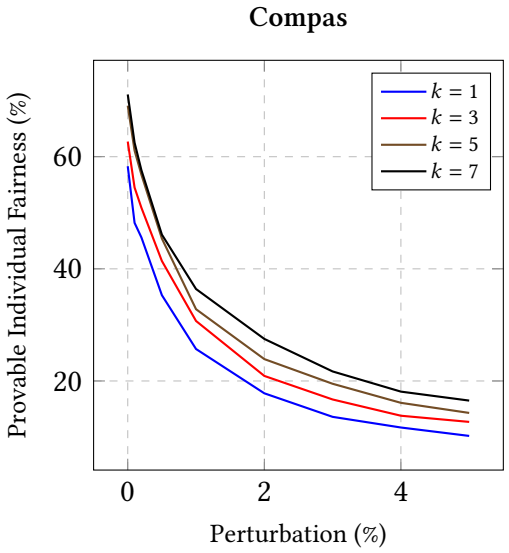

Figure 5.9: Provable individual fairness on the whole Compas test set.

Crime Provable Individual Fairness (%)				
ϵ	$k = 1$	$k = 3$	$k = 5$	$k = 7$
0	7.5	13.0	15.8	18.5
0.001	6.5	12.3	14.5	17.8
0.002	6.0	11.5	13.3	16.0
0.005	4.8	9.8	10.8	13.0
0.01	4.0	5.0	7.0	7.8

Table 5.13: Provable individual fairness on the whole Crime test set.

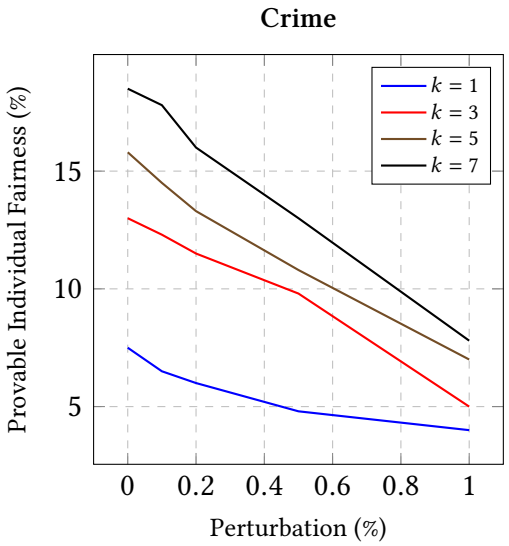

Figure 5.10: Provable individual fairness on the whole Crime test set.

German Provable Individual Fairness (%)				
ϵ	$k = 1$	$k = 3$	$k = 5$	$k = 7$
0	85.0	83.0	83.5	82.5
0.001	85.0	83.0	83.0	82.0
0.002	85.0	82.5	82.5	82.0
0.005	85.0	81.5	82.5	81.5
0.01	83.0	78.0	81.0	81.5
0.02	80.0	75.5	78.0	78.5
0.03	76.0	72.5	75.0	73.5
0.04	73.0	71.5	72.5	71.0
0.05	72.0	68.5	69.0	69.5

Table 5.14: Provable individual fairness on the whole German test set.

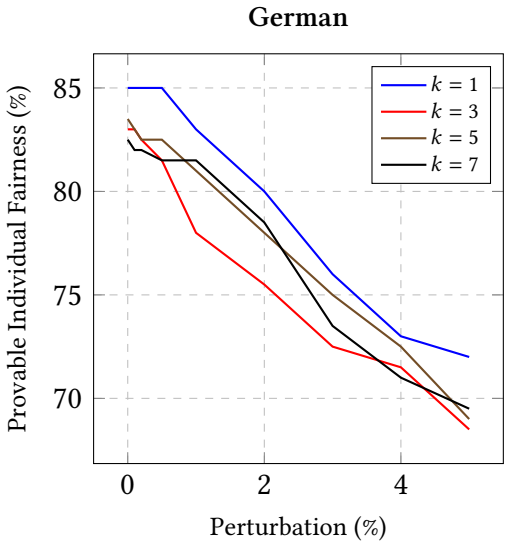

Figure 5.11: Provable individual fairness on the whole German test set.

5.4 Comparisons with GeoAdEx

To assess the quality of our lower bounds on robustness, we leverage the GeoAdEx tool [38]. As it compute the minimum-norm adversarial perturbation, as we saw in Section 3.4, a transformation was performed to achieve an output comparable with ours. More precisely, let ϵ be the minimum-norm adversarial perturbation returned by GeoAdEx. To retrieve the ϵ_∞ of the ℓ_∞-norm perturbation, we applied the following inversion:

$$\epsilon \triangleq \|\mathbf{d}^*\|_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{d}_i^*)^2} \implies \epsilon_\infty \triangleq \sqrt{\frac{\|\mathbf{d}^*\|_2^2}{n}}$$

This inversion formula is correct when for all $i, j \in [1, n]$, $\mathbf{d}_i^* = \mathbf{d}_j^*$, which is precisely the case of the ℓ_∞-norm. Once the ϵ_∞ has been computed for each verification sample in T, we can assess the quality of our robustness verification algorithm simply by using perturbation magnitudes close to ϵ_∞: a good result is obtained if magnitude perturbations slightly less than ϵ cannot affect the robustness of our abstract classifier.

GeoAdEx, on the other hand, has a few drawbacks. To begin with, GeoAdEx does not support one-hot encoding, so it assigns a progressive positive integer to each categorical value, introducing unwanted side-effects during distance computation. Second, GeoAdEx returns the optimal adversarial distance $\mathbf{d}^*$ only when either no time limit is set or the time limit is not reached; due to the high complexity of the minimization problem to be solved, in practice a time limit is always needed, and this makes GeoAdEx unsuitable for our goal on medium/large datasets and/or for large values of k, typically k > 3. As mentioned in Section 3.4, it relies on an approximation which limits the number of explored neighboring cells, obtained through a fast heuristics. The main problem of this approximation is that it makes impossible to know whether the adversarial distance $\mathbf{d}^*$ returned by GeoAdEx is actually precise or simply a lower bound.

We therefore chose to restrict our experiments on the quality of our robustness lower bounds to: (i) datasets with no categorical feature, and (ii) discharging the GeoAdEx approximation and just using a time limit of 2 minutes, so that if this time limit is reached then we still have a lower bound on the optimal adversarial distance $\mathbf{d}^*$. Because of the

limitation, it was unfeasible to go beyond $k = 1$ for datasets having more than 1000 samples. Table 5.15 reports precision percentages of kNAVe obtained by comparing the optimal result $\mathbf{d}^*$ of GeoAdEx, i.e., when the time limit was not met, to our uncertainty, according to the ratio:

$$precision \triangleq \frac{|\{(\mathbf{x}, l_\mathbf{x}) \in T \mid \neg\textsc{Robust}(C^{\mathbb{I}}_{D,k,\delta^{\mathbb{I}}}, P^{\epsilon}_{\infty}, \mathbf{x}, l_\mathbf{x}) \implies \text{GeoAdEx}(\mathbf{x}, l_\mathbf{x}) = \mathbf{d}^* \leq \epsilon\}|}{|\{(\mathbf{x}, l_\mathbf{x}) \in T \mid \neg\textsc{Robust}(C^{\mathbb{I}}_{D,k,\delta^{\mathbb{I}}}, P^{\epsilon}_{\infty}, \mathbf{x}, l_\mathbf{x}) \implies \text{GeoAdEx}(\mathbf{x}, l_\mathbf{x}) = \mathbf{d}^*\}|}$$

Thus, this metric intuitively expresses the exactness of our verification method, i.e., how many of the samples that our tool does not prove to be robust are actually not robust because there is a minimum optimal adversarial distance which is less than or equal to the distance between the center of the adversarial region and one of its corners.

Precision of our Robustness Verification on BreastCancer, Diabetes and Fourclass												
	BreastCancer				Diabetes				Fourclass			
ϵ	$k=1$	$k=3$	$k=5$	$k=7$	$k=1$	$k=3$	$k=5$	$k=7$	$k=1$	$k=3$	$k=5$	$k=7$
0.001	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
0.002	99.0	100.0	100.0	100.0	97.0	98.0	99.0	99.0	100.0	100.0	100.0	100.0
0.005	99.0	100.0	100.0	100.0	97.0	96.0	93.0	90.9	100.0	100.0	100.0	100.0
0.01	100.0	99.0	97.0	99.0	95.0	91.0	87.9	91.4	100.0	100.0	100.0	100.0
0.02	96.0	98.0	97.0	99.0	84.0	76.8	88.9	91.2	99.0	98.0	96.0	95.9
0.03	94.0	95.0	97.0	98.0	84.0	85.7	85.9	87.0	95.0	87.0	94.7	96.3
0.04	92.0	94.0	96.9	97.9	88.0	91.1	94.9	92.9	83.0	84.0	94.1	95.8
0.05	91.0	91.8	98.9	100.0	92.0	94.7	94.5	94.7	81.0	81.0	98.5	100.0

Table 5.15: Precision of our robustness verification on BreastCancer, Diabetes and Fourclass.

These data are encouraging because we reach 100% precision in 33 out of 96 case studies (34.37%), while the worst case is 76.8% of precision for Diabetes with $k = 3$ and $\epsilon = 0.02$. In particular, we observe a precision close to 100% for $\epsilon \leq 0.01$ in all datasets, and always above 90% for BreastCancer. These percentages are very accurate for $k = 1$, where GeoAdEx returns optimal adversarial distances for almost all test samples, and they gradually become less meaningful as k increases, because for them GeoAdEx returns less optimal adversarial distance.
Table 5.16 also shows the precision in datasets with more than 1000 samples. As we stated above, for the latter we have evaluated the robustness only with $k = 1$. Again, the percentages are still very promising.

Precision of our Robustness Verification on Letter, Pendigits and Satimage			
	Letter	Pendigits	Satimage
ϵ	$k = 1$		
0.001	100.0	100.0	100.0
0.002	98.0	100.0	100.0
0.005	100.0	100.0	100.0
0.01	99.0	100.0	96.0
0.02	97.0	99.0	84.0
0.03	99.0	94.0	80.0
0.04	100.0	97.0	73.0
0.05	100.0	89.0	82.0

Table 5.16: Precision of our robustness verification on Letter, Pendigits and Satimage.

6

Conclusion and Future Works

With this research work, we have shown how to design an abstract interpretation of k-nearest neighbor classifiers, and how this methodology defines, to the best of our knowledge, the first robustness verification framework for this popular ML algorithm. The experiments carried out on our implementation have highlighted that this approach is effective and precise, and that a kNN classifier is generally robust for small numerical perturbation. We are confident that the precision of our abstract interpretation-based technique can be further improved by leveraging more refined abstract domains, specifically the relational ones. Notably, the standard interval abstraction, used in this thesis, is not able to express the mutual exclusion of two or more scores. For example, if we have $k = 3$ and the score intervals $\mathrm{s}[A] = [1, 3]$, $\mathrm{s}[B] = [0, 1]$, and $\mathrm{s}[C] = [0, 1]$, to be sound we are forced to not consider the classifier robust, because the case where all labels are considered exactly once exists. However, if we knew that between B and C there is a mutual exclusion, the outcome would be different: If B and C cannot occur at the same time, then A would receive at least 2 votes, thus making it the most voted label, and therefore the classifier would have been decreed robust.

It may be a future challenge to use the so-called *reduced affine forms* (RAFs) [22] as they allow us to keep track of the correlations between features. RAFs have already been successfully exploited for the formal verification of support vector machine classifiers [28].

References

[1] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[2] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," 2016. [Online]. Available: https://arxiv.org/abs/1606.06565

[3] G. Bontempi, M. Birattari, and H. Bersini, "Lazy learning for local modelling and control design," *International Journal of Control*, vol. 72, no. 7-8, pp. 643–658, 1999.

[4] S. Calzavara, P. Ferrara, and C. Lucchese, "Certifying decision trees against evasion attacks by program analysis," in *Proc. 25th European Symposium on Research, ESORICS 2020*, ser. LNCS. Springer, 2020.

[5] N. Carlini and D. A. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *Proc. 38th IEEE Symposium on Security and Privacy, IEEE S&P 2017*, 2017, pp. 39–57.

[6] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[7] P. Cousot, *Principles of Abstract Interpretation*. MIT Press, 2021.

[8] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Proc. 4th ACM Symposium on Principles of Programming Languages, POPL 1977*, 1977, pp. 238–252.

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[10] B. Daniels, *Rote-LCS learning classifier system for classification and prediction.* Missouri University of Science and Technology, 2015.

[11] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness Through Awareness," in *Proc. 3rd Innovations in Theoretical Computer Science Conference*, 2012, pp. 214–226.

[12] N. Fassina, GitHub Page: https://github.com/nicolofassina.

[13] P. C. Fishburn, *Interval Orders and Interval Graphs: A Study of Partially Ordered Sets.* Wiley, 1985.

[14] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review/Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[15] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev, "AI2: Safety and robustness certification of neura networks with abstract interpretation," in *Proc. 2018 IEEE Symposium on Security and Privacy, IEEE S&P 2018*, 2018, pp. 3–18. [Online]. Available: https://doi.org/10.1109/SP.2018.00058

[16] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Commun. ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[17] B. R. Kowalski and C. F. Bender, "K-nearest neighbor classification rule (pattern recognition) applied to nuclear magnetic resonance spectral interpretation," *Analytical Chemistry*, vol. 44, no. 8, pp. 1405–1411, 1972. [Online]. Available: https://doi.org/10.1021/ac60316a008

[18] O. Kramer, "K-nearest neighbors," in *Dimensionality Reduction with Unsupervised Nearest Neighbors, Intelligent Systems Reference Library, vol 51.* Berlin: Springer, 2013, pp. 13–23.

[19] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. 5th Int. Conf. on Learning Representations, ICLR 2017*, 2017. [Online]. Available: https://openreview.net/forum?id=BJm4T4Kgx

[20] J. McCarthy, "What is artificial intelligence," http://www-formal.stanford.edu/jmc/whatisai.html, 2004.

[21] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–35, 2021.

[22] F. Messine, "Extensions of affine arithmetic: Application to unconstrained global optimization," *J. UCS*, vol. 8, pp. 992–1015, 01 2002.

[23] A. Miné, "Tutorial on static inference of numeric invariants by abstract interpretation," *Foundations and Trends in Programming Languages*, vol. 4, no. 3-4, pp. 120–372, 2017. [Online]. Available: https://doi.org/10.1561/2500000034

[24] M. Mirman, T. Gehr, and M. T. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018, pp. 3575–3583. [Online]. Available: http://proceedings.mlr.press/v80/mirman18b.html

[25] R. E. Moore, *Interval analysis*. USA: Prentice-Hall, Englewood Cliffs N.J, 1966.

[26] C. Müller, F. Serre, G. Singh, M. Püschel, and M. T. Vechev, "Scaling polyhedral neural network verification on gpus," in *Proceedings of Machine Learning and Systems 2021, MLSys 2021*, 2021. [Online]. Available: https://proceedings.mlsys.org/paper/2021/hash/ca46c1b9512a7a8315fa3c5a946e8265-Abstract.html

[27] F. Ranzato, C. Urban, and M. Zanella, "Fairness-aware training of decision trees by abstract interpretation," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management, CIKM 2021*. ACM, 2021, pp. 1508–1517.

[28] F. Ranzato and M. Zanella, "Robustness Verification of Support Vector Machines," in *Proc. 26th International Static Analysis Symposium, SAS 2019*, ser. LNCS, vol. 11822, 2019, pp. 271–295.

[29] ——, "Abstract interpretation of decision tree ensemble classifiers," in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 2020, pp. 5478–5486. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/5998

[30] ——, "Genetic adversarial training of decision trees," in *Proceedings of the 2021 Genetic and Evolutionary Computation Conference, GECCO 2021*. ACM, 2021, pp. 358–367.

[31] S. Raschka, "Stat 479: Machine learning lecture notes," https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf, 2018.

[32] X. Rival and K. Yi, *Introduction to Static Analysis: An Abstract Interpretation Perspective*. The MIT Press, 2020.

[33] A. Ruoss, M. Balunovic, M. Fischer, and M. Vechev, "Learning certified individually fair representations," in *Proc. 34th Annual Conference on Advances in Neural Information Processing Systems, NeurIPS 2020*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/55d491cf951b1b920900684d71419282-Abstract.html

[34] R. E. Shaffer, S. L. Rose-Pehrsson, and R. A. McGill, "A comparison study of chemical sensor array pattern recognition algorithms," *Analytica Chimica Acta*, vol. 384, pp. 305–317, 1999.

[35] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. T. Vechev, "Fast and effective robustness certification," in *Proc. Annual Conf. on Neural Information Processing Systems, NeurIPS 2018*, 2018, pp. 10 825–10 836. [Online]. Available: http://papers.nips.cc/paper/8278-fast-and-effective-robustness-certification

[36] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, no. POPL 2019, pp. 41:1–41:30, Jan. 2019. [Online]. Available: http://doi.acm.org/10.1145/3290354

[37] G. Singh, T. Gehr, M. Püschel, and M. T. Vechev, "Boosting robustness certification of neural networks," in *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*, 2019. [Online]. Available: https://openreview.net/forum?id=HJgeEh09KQ

[38] C. Sitawarin, E. M. Kornaropoulos, D. Song, and D. A. Wagner, "Adversarial examples for k-nearest neighbor classifiers based on higher-order Voronoi diagrams," in *Proc. Annual Conference on Neural Information Processing Systems,*

NeurIPS 2021, 2021, pp. 15 486–15 497. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/82ca5dd156cc926b2992f73c2896f761-Abstract.html

[39] C. Sitawarin and D. Wagner, "Defending against adversarial examples with k-nearest neighbor," 2019. [Online]. Available: https://arxiv.org/abs/1906.09525

[40] ——, "On the robustness of deep k-nearest neighbors," 2019. [Online]. Available: https://arxiv.org/abs/1903.08333

[41] ——, "Minimum-norm adversarial examples on KNN and KNN based models," in *2020 IEEE Security and Privacy Workshops, SP Workshops, 2020*. IEEE, 2020, pp. 34–40.

[42] C. Urban and A. Miné, "A Review of Formal Methods applied to Machine Learning," *CoRR*, vol. abs/2104.02466, 2021. [Online]. Available: https://arxiv.org/abs/2104.02466

[43] Y. Vorobeychik and M. Kantarcioglu, "Adversarial machine learning," in *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, August 2018, vol. 12(3), pp. 1–169.

[44] L. Wang, X. Liu, J. Yi, Z.-H. Zhou, and C.-J. Hsieh, "Evaluating the robustness of nearest neighbor classifiers: A primal-dual perspective," 2019. [Online]. Available: https://arxiv.org/abs/1906.03972

[45] Y. Wang, S. Jha, and K. Chaudhuri, "Analyzing the robustness of nearest neighbors to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 5120–5129. [Online]. Available: http://proceedings.mlr.press/v80/wang18c.html

[46] Y. Yang, C. Rashtchian, Y. Wang, and K. Chaudhuri, "Robustness for non-parametric classification: A generic attack and defense," in *Proc. 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, ser. Proceedings of Machine Learning Research, vol. 108. PMLR, 2020, pp. 941–951. [Online]. Available: http://proceedings.mlr.press/v108/yang20b.html