Robustness Verification of Support Vector Machines

Francesco Ranzato[0000−0003−0159−0068] and Marco Zanella

Dipartimento di Matematica, University of Padova, Italy

Abstract. We study the problem of formally verifying the robustness to adversarial examples of support vector machines (SVMs), a major machine learning model for classification and regression tasks. Following a recent stream of works on formal robustness verification of (deep) neural networks, our approach relies on a sound abstract version of a given SVM classifier to be used for checking its robustness. This methodology is parametric on a given numerical abstraction of real values and, analogously to the case of neural networks, needs neither abstract least upper bounds nor widening operators on this abstraction. The standard interval domain provides a simple instantiation of our abstraction technique, which is enhanced with the domain of reduced affine forms, an efficient abstraction of the zonotope abstract domain. This robustness verification technique has been fully implemented and experimentally evaluated on SVMs based on linear and nonlinear (polynomial and radial basis function) kernels, which have been trained on the popular MNIST dataset of images and on the recent and more challenging Fashion-MNIST dataset. The experimental results of our prototype SVM robustness verifier appear to be encouraging: this automated verification is fast, scalable and shows significantly high percentages of provable robustness on the test set of MNIST, in particular compared to the analogous provable robustness of neural networks.

1 Introduction

Adversarial machine learning [10,17,38] is an emerging hot topic studying vulnerabilities of machine learning (ML) techniques in adversarial scenarios and whose main objective is to design methodologies for making learning tools robust to adversarial attacks. Adversarial examples have been found in diverse application fields of ML such as image classification, speech recognition and malware detection [10]. Current defense techniques include adversarial model training, input validation, testing and automatic verification of learning algorithms (see the recent survey [10]). In particular, formal verification of ML classifiers started to be an active field of investigation [1,8,9,12,15,16,23,26,27,31,32,39,40,19] within the verification and static analysis community. Robustness to adversarial inputs is an important safety property of ML classifiers whose formal verification has been investigated for (deep) neural networks [1,9,26,31,32,40]. A classifier is robust to some (typically small) perturbation of its input objects representing an adversarial attack when it assigns the same class to all the objects within that perturbation. Thus, slight malicious alterations of input objects should not deceive a robust classifier. Pulina and Tacchella [26] first put forward the idea

of a formal robustness verification of neural network classifiers by leveraging interval-based abstract interpretation for designing a sound abstract classifier. This abstraction-based verification approach has been pushed forward by Vechev et al. [9,31,32], who designed a scalable robustness verification technique which relies on abstract interpretation of deep neural networks based on a specifically tailored abstract domain [32].

While all the aforementioned verification techniques consider (deep) neural networks as ML model, in this work we focus on support vector machines (SVMs), which is a major learning model extensively and successfully used for both classification and regression tasks [7]. SVMs are widely applied in different fields where adversarial attacks must be taken into account, notably image classification, malware detection, intrusion detection and spam filtering [2]. Adversarial attacks and robustness issues of SVMs have been defined and studied by some authors [2,3,24,37,41,43,46], in particular investigating robust training and experimental robustness evaluation of SVMs. To the best of our knowledge, no formal and automatic robustness certification technique for SVMs has been studied.

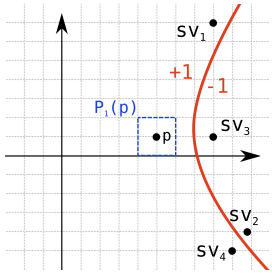
Contributions. A simple and standard model of adversarial region for a ML classifier $C : X \to L$, where $X \subseteq \mathbb{R}^n$ is the input space and L is the set of classes (or labels), is based on a set of perturbations $P(\mathbf{x}) \subseteq X$ of an input $\mathbf{x} \in X$ for C, which typically exploits some metric on $\mathbb{R}^n$ to quantify a similarity to $\mathbf{x}$. A classifier C is robust on an input $\mathbf{x}$ for a perturbation P when for all $\mathbf{x}' \in P(\mathbf{x})$, $C(\mathbf{x}') = C(\mathbf{x})$ holds, meaning that the adversary cannot attack the classification of $\mathbf{x}$ made by C by selecting input objects from $P(\mathbf{x})$ [4]. We consider the most effective SVM classifiers based on common linear and nonlinear kernels, in particular polynomial and Gaussian radial basis function (RBFs) [7]. Our technique for formally verifying the robustness of C is quite standard: by leveraging a numerical abstraction A of sets of real vectors in $\wp(\mathbb{R}^n)$, we define a sound abstract classifier $C^\sharp : A \to \wp(L)$ and a sound abstract perturbation $P^\sharp : X \to A$, in such a way that if $C^\sharp(P^\sharp(\mathbf{x})) = \{C(\mathbf{x})\}$ holds then C is proved to be robust on the input $\mathbf{x}$ for the adversarial region P. As usual in static analysis, scalability and precision are the main issues in SVM verification. A robustness verifier has to scale with the number of support vectors of the SVM classifier C, which in turn depends on the size of the training dataset for C, which may be huge (easily tens/hundreds of thousands of samples). Moreover, the precision of a verifier may crucially depend on the relational information between the components, called features in ML, of input vectors in $\mathbb{R}^n$, whose number may be quite large (easily hundreds/thousands of features). For our robustness verifier, we used an abstraction which is a product of the standard nonrelational interval domain [6] and of the so-called reduced affine form (RAF) abstraction, a relational domain representing the dependencies from the components of input vectors. A RAF for vectors in $\mathbb{R}^n$ is given by $a_0 + \sum_{i=1}^{n} a_i\epsilon_i + a_r\epsilon_r$, where ϵ_i's are symbolic variables ranging in [-1,1] and representing a dependence from the i-th component of the vector, while ϵ_r is a further symbolic variable in [-1,1] which accumulates all the approximations introduced by nonlinear operations such as multiplication and exponential. RAFs can be viewed as a restriction to a given length (here the dimension n of $\mathbb{R}^n$) of the zonotope domain used in static program analysis [13], which features an optimal abstract multiplication [33], the crucial operation of abstract nonlinear SVMs. We implemented our robustness verification method for SVMs in a

tool called *SAVer* (*S*vm *A*bstract *Ver*ifier), written in C. Our experimental evaluation of SAVer employed the popular MNIST [18] image dataset and the recent and more challenging alternative Fashion-MNIST dataset [42]. Our benchmarks provide the percentage of samples of the full test sets for which a SVM is proved to be robust (and, dually, vulnerable) for a given perturbation, the average verification times per sample, and the scalability of the robustness verifier w.r.t. the number of support vectors. We also compared SAVer to DeepPoly [32], a robustness verification tool for deep neural networks based on abstract interpretation. Our experimental results indicate that SAVer is fast and scalable and that the percentage of robustness provable by SAVer for SVMs is higher than the robustness provable by DeepPoly for deep neural networks.

Illustrative Example. The figure below shows a toy binary SVM classifier for input vectors in $\mathbb{R}^2$, with four support vectors $\mathbf{sv_1} = (8, 7)$, $\mathbf{sv_2} = (10, -4)$, $\mathbf{sv_3} = (8, 1)$, $\mathbf{sv_4} = (9, -5)$ for a polynomial kernel of degree 2. The corresponding binary classifier $C : \mathbb{R}^2 \to \{-1, +1\}$ is the following function:

$$\begin{aligned} C(\mathbf{x}) &= \text{sign}(\textstyle\sum_{i=1}^{4} \alpha_i y_i (\mathbf{sv_i} \cdot \mathbf{x})^2 + b) \\ &= \text{sign}(\alpha_1(8x_1+7x_2)^2 - \alpha_2(10x_1-4x_2)^2 - \alpha_3(8x_1+x_2)^2 + \alpha_4(9x_1-5x_2)^2 + b) \end{aligned}$$

where y_i and α_i are, resp., the classes (± 1) and weights of the support vectors $\mathbf{sv_i}$, with: $\alpha_1 \approx 5.36 \times 10^{-4}$, $\alpha_2 \approx -3.78 \times 10^{-3}$, $\alpha_3 \approx -9.23 \times 10^{-4}$, $\alpha_4 \approx 4.17 \times 10^{-3}$, $b \approx 3.33$. The set of vectors $\mathbf{x} \in \mathbb{R}^2$ such that $C(\mathbf{x}) = 0$ defines the decision curve between labels –1 and +1. We consider a point $\mathbf{p} = (5, 1)$ and an adversarial region $P_1(\mathbf{p}) = \{\mathbf{x} \in \mathbb{R}^2 \mid \max(|x_1 - p_1|, |x_2 - p_2|) \leq 1\}$, which is the L_∞ ball of radius 1 centered in $\mathbf{p}$ and can be exactly represented by the interval in $\mathbb{R}^2$ (i.e., box) $P_1(\mathbf{p}) = (x_1 \in [4, 6], x_2 \in [0, 2])$. As shown by the figure, this classifier C is robust on $\mathbf{p}$ for this perturbation because for all $\mathbf{x} \in P_1(\mathbf{p})$, $C(\mathbf{x}) = C(\mathbf{p}) =$ +1. However, it turns out that the interval abstraction $C^\sharp_{\text{Int}}$ of this classifier cannot prove the robustness of C:



$$\begin{aligned} &C^\sharp_{\text{Int}}(P_1(\mathbf{p})) = \text{sign}(\textstyle\sum_{i=1}^{4} \alpha_i y_i ((\mathbf{sv_i})_1[4,6] + (\mathbf{sv_i})_2[0,2])^2 + b) \\ &\quad = \text{sign}(\alpha_1 y_1 [1024, 3844] + \alpha_2 y_2 [1024, 3600] + \alpha_3 y_3 [1024, 2500] + \alpha_4 y_4 [676, 2916] + b) \\ &\quad = \text{sign}([-9.231596, 12.735958]) = \top \end{aligned}$$

Instead, the reduced affine form abstraction $C^\sharp_{\text{RAF}_2}$ allows us to prove the robustness of C on $\mathbf{p}$. Here, the perturbation $P_1(\mathbf{p})$ is exactly represented by the RAF ($\tilde{x}_1 = 5 + \epsilon_1, \tilde{x}_2 = 1 + \epsilon_2$), where $\epsilon_1, \epsilon_2, \epsilon_r \in [-1, 1]$, and the abstract computation is as follows:

$$\begin{aligned} &C^\sharp_{\text{RAF}_2}(P_1(\mathbf{p})) = \text{sign}(\textstyle\sum_{i=1}^{4} \alpha_i y_i [(\mathbf{sv_i})_1(5+\epsilon_1) + (\mathbf{sv_i})_2(1+\epsilon_2)]^2 + b) \\ &\quad = \text{sign}(\alpha_1 y_1 (47 + 8\epsilon_1 + 7\epsilon_2)^2 + \alpha_2 y_2 (46 + 10\epsilon_1 - 4\epsilon_2)^2 + \\ &\qquad \alpha_3 y_3 (41 + 8\epsilon_1 + \epsilon_2)^2 + \alpha_4 y_4 (40 + 9\epsilon_1 - 5\epsilon_2)^2 + b) \\ &\quad = \text{sign}(\alpha_1 y_1 (2322 + 752\epsilon_1 + 658\epsilon_2 + 112\epsilon_r) + \alpha_2 y_2 (2232 + 920\epsilon_1 - 368\epsilon_2 + 80\epsilon_r) + \\ &\qquad \alpha_3 y_3 (1746 + 656\epsilon_1 + 82\epsilon_2 + 16\epsilon_r) + \alpha_4 y_4 (1706 + 720\epsilon_1 - 400\epsilon_2 + 90\epsilon_r) + b) \\ &\quad = \text{sign}(1.635264 - 0.680779\epsilon_1 + 0.001047\epsilon_2 + 0.753025\epsilon_r) = +1 \end{aligned}$$

Hence, the RAF analysis is able to prove that C is robust on $\mathbf{p}$ for P_1, since the final RAF has an interval range $[0.200413, 3.070115]$ consisting of positive numbers.

2 Background

Notation. If $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $z \in \mathbb{R}$ and $i \in [1, n]$ then $\mathbf{x}_i = \pi_i(\mathbf{x}) \in \mathbb{R}$, $\mathbf{x} \cdot \mathbf{y} \triangleq \sum_i \mathbf{x}_i \mathbf{y}_i \in \mathbb{R}$, $\mathbf{x} + \mathbf{y} \in \mathbb{R}^n$, $z\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_2 \triangleq \sqrt{\mathbf{x} \cdot \mathbf{x}} \in \mathbb{R}$, $\|\mathbf{x}\|_\infty \triangleq \max\{|\mathbf{x}_i| \mid i \in [1, n]\} \in \mathbb{R}$, denote, resp., i-th component, dot product, vector addition, scalar multiplication, L_2 (i.e., Euclidean) and L_∞ (i.e., maximum) norms in $\mathbb{R}^n$. If $h : X \to Y$ is any function then $h^c : \wp(X) \to \wp(Y)$ defined by $h^c(S) \triangleq \{h(x) \mid x \in S\}$ denotes the standard collecting lifting of h, and, when clear from the context, we slightly abuse notation by using $h(S)$ instead of $h^c(S)$.

Classifiers and Robustness. Consider a training dataset $T = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_N}, y_N)\} \subseteq X \times L$, where $X \subseteq \mathbb{R}^n$ is the input space, $\mathbf{x_i} \in X$ is called feature (or attribute) vector and y_i is its label (or class) ranging into the output space L. A supervised learning algorithm $\mathcal{SL} : \wp(X \times L) \to (X \to L)$ (also called trainer) computes a classifier function $\mathcal{SL}(T) : X \to L$ ranging in some function subspace (also called hypothesis space). The learned classifier $\mathcal{SL}(T)$ is a function that best fits the training dataset T according to a principle of empirical risk minimization. The machine learning algorithm $\mathcal{SL}$ computes a classifier $\mathcal{SL}(T)$ by solving a complex optimization problem. The output space is assumed to be represented by real numbers, i.e., $L \subseteq \mathbb{R}$, and for binary classifiers with $|L| = 2$, the standard assumption is that $L = \{-1, +1\}$.

The standard threat model [4,10] of untargeted adversarial examples for a generic classifier $C : X \to L$ is as follows. Given a valid input object $\mathbf{x} \in X$ whose correct label is $C(\mathbf{x})$, an adversarial example for $\mathbf{x}$ is a legal input $\mathbf{x}' \in X$ such that $\mathbf{x}'$ is a small perturbation of (i.e., is similar to) $\mathbf{x}$ and $C(\mathbf{x}') \neq C(\mathbf{x})$. An adversarial region is the set of perturbations $P(\mathbf{x}) \subseteq X$ that the adversary is allowed to make to $\mathbf{x}$, meaning that a function $P : X \to \wp(X)$ models an adversarial region. A perturbation $P(\mathbf{x})$ is typically modeled by some distance metric to quantify a similarity to $\mathbf{x}$, usually a p-norm, and the most general model of perturbation simply requires that for all $\mathbf{x} \in X$, $\mathbf{x} \in P(\mathbf{x})$. A classifier C is defined to be robust on an input vector $\mathbf{x}$ for an adversarial region P when for all $\mathbf{x}' \in P(\mathbf{x})$, $C(\mathbf{x}') = C(\mathbf{x})$ holds, denoted by $\mathrm{Rob}(C, \mathbf{x}, P) \triangleq \{C(\mathbf{x}') \mid \mathbf{x}' \in P(\mathbf{x})\} = \{C(\mathbf{x})\}$. This means that the adversary cannot attack the classification of $\mathbf{x}$ made by C by selecting input objects from the region $P(\mathbf{x})$.

Support Vector Machines. Several strategies and optimization techniques are available to train a SVM, but they are not relevant for our purposes ([7] is a popular standard reference for SVMs). A SVM classifier partitions the input space X into regions, each representing a class of the output space L. In its simplest formulation, the learning algorithm produces a linear SVM binary classifier with $L = \{-1, +1\}$ which relies on a hyperplane of $\mathbb{R}^n$ that separates training vectors labeled by –1 from vectors labeled +1. The training phase consists in finding (i.e., learning) this hyperplane. While many separating hyperplanes may exist, the SVM separating hyperplane has the maximum distance (called margin) with the closest vectors in the training dataset, because a maximum-margin learning algorithm statistically reduces the generalization error. This

SVM hyperplane is univocally represented by its normal vector $\mathbf{w} \in \mathbb{R}^n$ and by a displacement scalar $b \in \mathbb{R}$, so that the hyperplane equation is $\mathbf{w} \cdot \mathbf{x} = b$. The classification of an input vector $\mathbf{x} \in X$ therefore boils down to determining the half-space containing $\mathbf{x}$, namely, the linear binary classifier is the decision function $C(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$, where the case $\text{sign}(0) = 0$ is negligible (e.g. $\text{sign}(0)$ may assign the class +1). This linear classifier $\text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$ is in so-called primal form, while nonlinear classifiers are instead in dual form and based on a so-called kernel function.

When the training set T cannot be linearly separated in a satisfactory way, T is projected into a much higher dimensional space through a projection map $\varphi : \mathbb{R}^n \to \mathbb{R}^k$, with $k > n$, where $\varphi(T)$ may become linearly separable. Training a SVM classifier boils down to a high-dimensional quadratic programming problem which can be solved either in its primal or dual form. When solving the dual problem, the projection function φ is only involved in dot products $\varphi(\mathbf{x}) \cdot \varphi(\mathbf{y})$ in $\mathbb{R}^k$, so that this projection is not actually needed if these dot products in $\mathbb{R}^k$ can be equivalently formulated through a function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, called kernel function, such that $k(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y})$. Given a dataset $T = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_N}, y_N)\}$, with $y_i \in \{-1, +1\}$, solving the dual problem for training the SVM classifier means finding a set $\{\alpha_i\}_{i=1}^N \subseteq \mathbb{R}$, called set of weights, which maximizes the following function $f : \mathbb{R}^N \to \mathbb{R}$:

$$\max f(\alpha_1, ..., \alpha_N) \triangleq \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j k(\mathbf{x_i}, \mathbf{x_j})$$

subject to: for all i, $0 \leq \alpha_i \leq c$, where $c \in \mathbb{R}_{>0}$ is a tuning parameter, and $\sum_{i=1}^N \alpha_i y_i = 0$. This set of weights defines the following SVM binary classifier C: for all input $\mathbf{x} \in X \subseteq \mathbb{R}^n$,

$$C(\mathbf{x}) \triangleq \text{sign}([\textstyle\sum_{i=1}^N \alpha_i y_i k(\mathbf{x_i}, \mathbf{x})] - b) \tag{1}$$

for some offset parameter $b \in \mathbb{R}$. By defining $D_k(\mathbf{x}) \triangleq \sum_{i=1}^N \alpha_i y_i k(\mathbf{x_i}, \mathbf{x})$, this classifier will be also denoted by $C(\mathbf{x}) = \text{sign}(D_k(\mathbf{x}) - b)$. In practice most weights α_i are 0, hence only a subset of the training vectors $\mathbf{x_i}$ is actually used by the SVM classifier C, and these are called support vectors. By a slight abuse of notation, we will assume that $\alpha_i \neq 0$ for all $i \in [1, N]$, namely $\{\mathbf{x_i}\}_{i=1}^N \subseteq \mathbb{R}^n$ denotes the set of support vectors extracted from the training set by the SVM learning algorithm for some kernel function. We will consider the most common and effective kernel functions used in SVM training: (i) linear kernel: $k(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$; (ii) d-polynomial kernel: $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^d$ (common powers are $d = 2, 3, 9$); (iii) Gaussian radial basis function (RBF): $k(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x} - \mathbf{y}\|_2^2}$, for some $\gamma > 0$.

SVM Multiclass Classification. Multiclass datasets have a finite set of labels $L = \{y_1, ..., y_m\}$ with $m > 2$. The standard approach to multiclass classification problems consists in a reduction into multiple binary classification problems using one of the following two simple strategies [14]. In the "one-versus-rest" (ovr) strategy, m binary classifiers are trained, where each binary classifier $C_{i,\bar{i}}$ determines whether an input vector $\mathbf{x}$ belongs to the class $y_i \in L$ or not by assigning a real confidence score for its decision rather than just a label, so that the class y_j with the highest-output confidence score is the class assigned to $\mathbf{x}$. Multiclass SVMs using this ovr approach might not work satisfactorily because the ovr approach often leads to unbalanced datasets already for a few classes due to unbalanced partitions into y_i and $L \setminus \{y_i\}$.

The most common solution [14] is to follow a "one-versus-one" (ovo) approach, where $m(m-1)/2$ binary classifiers $C_{\{i,j\}}$ are trained on the restriction of the original training set to vectors with labels in $\{y_i, y_j\}$, with $i \neq j$, so that each $C_{\{i,j\}}$ determines whether an input vector belongs (more) to the class y_i or (more to) y_j. Given an input vector $\mathbf{x} \in X$ each of these $m(m-1)/2$ binary classifiers $C_{\{i,j\}}(\mathbf{x})$ assigns a "vote" to one class in $\{y_i, y_j\}$, and at the end the class with the most votes wins, i.e., the argmax of the function $\mathrm{votes}(\mathbf{x}, y_i) \triangleq |\{j \in \{1, ..., m\} \mid j \neq i,\ C_{\{i,j\}}(\mathbf{x}) = y_i\}|$ is the winning class of $\mathbf{x}$. Draw is a downside of the ovo strategy because it may well be the case that for some (regions of) input vectors multiple classes collect the same number of votes and therefore no classification can be done. In case of draw, a common strategy [14] is to output any of the winning classes (e.g., the one with the smaller index). However, since our primary focus is on soundness of abstract classifiers, we need to model an ovo multiclass classifier by a function $M_{\text{ovo}} : X \to \wp(L)$ defined by $M_{\text{ovo}}(\mathbf{x}) \triangleq \{y_k \in L \mid k \in \mathrm{argmax}_{i \in \{1,\ldots,m\}} \mathrm{votes}(\mathbf{x}, y_i)\}$, so that $|M_{\text{ovo}}(\mathbf{x})| > 1$ models a draw in the ovo voting.

Numerical Abstractions. According to the most general definition, a numerical abstract domain is a tuple $\langle A, \leq_A, \gamma \rangle$ where $\langle A, \leq_A \rangle$ is at least a preordered set and the concretization function $\gamma : A \to \wp(\mathbb{R}^n)$, with $n \geq 1$, preserves the relation $\leq_A$, namely, $a \leq_A a'$ implies $\gamma(a) \subseteq \gamma(a')$ (i.e., γ is monotone). Thus, A plays the usual role of set of symbolic representations for sets of vectors of $\mathbb{R}^n$. Well-known examples of numerical abstract domains include intervals, zonotopes, octagons, convex polyhedra (we refer to the tutorial [22]). Some numerical domains just form preorders (e.g., standard representations of octagons by DBMs allow multiple representations) while other domains give rise to posets (e.g., intervals). While a monotone concretization γ is enough for reasoning about soundness of static analyses on numerical domains, the notion of best correct approximation of concrete sets relies on the existence of an abstraction function $\alpha : \wp(\mathbb{R}^n) \to A$ which requires that $\langle A, \leq_A \rangle$ is (at least) a poset and that the pair (α, γ) forms a Galois connection/insertion. Consider a concrete k-ary real operation $f : \wp(\mathbb{R}^n)^k \to \wp(\mathbb{R}^n)$, for some $k \in \mathbb{N}_{>0}$, and a corresponding abstract map $f^\sharp : A^k \to A$. Then, $f^\sharp$ is a correct (or sound) approximation of f when $f \circ \langle \gamma, ..., \gamma \rangle \subseteq \gamma \circ f^\sharp$ holds, while $f^\sharp$ is exact (or γ-complete) when $f \circ \langle \gamma, ..., \gamma \rangle = \gamma \circ f^\sharp$ holds. When a Galois connection (α, γ) for A exists, if $f^\sharp$ is exact then it coincides with the best correct approximation (bca) of f on A, which is the abstract function $\alpha \circ f \circ \langle \gamma, ..., \gamma \rangle : A^k \to A$ [28]. The abstract domain Int of numerical intervals on the poset of real numbers $\langle \mathbb{R} \cup \{-\infty, +\infty\}, \leq \rangle$ is defined as usual [6]:

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

The concretization map $\gamma : \mathrm{Int} \to \wp(\mathbb{R})$ is standard. Intervals admit an abstraction map $\alpha : \wp(\mathbb{R}) \to \mathrm{Int}$ such that $\alpha(X)$ is the least interval containing X, so that (α, γ) defines a Galois insertion between $\langle \mathrm{Int}, \sqsubseteq \rangle$ and $\langle \wp(\mathbb{R}), \subseteq \rangle$.

3 Abstract Robustness Verification Framework

Let us describe a sound abstract robustness verification framework for binary and multiclass SVM classifiers. We consider a general classifier $C : X \to L$, where L is a set

of labels, and an adversarial region $P : X \to \wp(X)$ for C. Consider a numerical abstract domain $\langle A, \leq_A \rangle$ whose abstract values represent sets of input vectors for a binary classifier C, namely $\gamma : A \to \wp(X)$, where X is the input space of C. We use A_n to emphasize that A is used as an abstraction of properties of n-dimensional vectors in $\mathbb{R}^n$, so that A_1 denotes that A is used as an abstraction of sets of scalars in $\wp(\mathbb{R})$.

Definition 3.1 (Sound Abstract Classifier). A *sound abstract classifier* on A is an algorithm $C^\sharp : A \to \wp(L)$ such that, for all $a \in A$, $\{C(\mathbf{x}) \in L \mid \mathbf{x} \in \gamma(a)\} \subseteq C^\sharp(a)$ holds. □

Thus, $C^\sharp$ is a sound abstraction of a classifier C when, given an abstract value $a \in A$ representing a set of concrete inputs, $C^\sharp(a)$ computes a superset of the labels computed by C on inputs ranging in $\gamma(a)$. In particular, the output $C^\sharp(a) = L$ plays the role of a "don't know" answer, while if $|C^\sharp(a)| = 1$ then every sample in $\gamma(a)$ must necessarily be classified by C with a same label $C^\sharp(a)$.

Definition 3.2 (Sound Abstract Perturbation). A *sound abstract perturbation* is a function $P^\sharp : X \to A$ which is sound for P, i.e., for all $\mathbf{x} \in X, P(\mathbf{x}) \subseteq \gamma(P^\sharp(\mathbf{x}))$. □

A sound abstract classifier and a sound abstract perturbation $\langle C^\sharp, P^\sharp \rangle$ allows us to define a robustness verifier as follows

Theorem 3.3 (Robustness Verifier). *If $C^\sharp$ and $P^\sharp$ are sound then $\langle C^\sharp, P^\sharp \rangle$ is a sound robustness verifier, namely, for all $\mathbf{x} \in X$, $|C^\sharp(P^\sharp(\mathbf{x}))| = 1 \Rightarrow Rob(C, \mathbf{x}, P)$.*

As multiclass SVMs combine the outputs of a number of binary classifiers, let us focus on binary classifiers $C : X \to \{-1, +1\}$, where $C(\mathbf{x}) = \mathrm{sign}(D(\mathbf{x}) - b)$ and $D : X \to \mathbb{R}$ has been trained for some kernel function k. For the sake of clarity we will use a slightly different notation for $C^\sharp : X \to \{-1, +1, \top\}$, where $\top$ is an abstract "don't know" value representing $\{-1, +1\}$. Of course, the key step for defining an abstract robustness verifier is to design a sound abstract version of the trained function $D : X \to \mathbb{R}$ on some abstraction A, namely an algorithm $D^\sharp : A_n \to A_1$ such that, for all $a \in A_n$, $D^c(\gamma(a)) \subseteq \gamma(D^\sharp(a))$. We also need that the abstraction A is endowed with a sound approximation of the Boolean test $\mathrm{sign}_b(\cdot) : \mathbb{R} \to \{-1, +1\}$ for any bias $b \in \mathbb{R}$, where $\mathrm{sign}_b(x) \triangleq$ **if** $x \geq b$ **then** $+1$ **else** -1. Hence, we require a computable abstract function $\mathrm{sign}_b^\sharp : A_1 \to \{-1, +1, \top\}$ which is sound for sign_b, that is, for all $a \in A_1$, $\mathrm{sign}_b^\sharp(a) \neq \top \Rightarrow \forall x \in \gamma(a).\, \mathrm{sign}_b(x) = \mathrm{sign}_b^\sharp(a)$. These hypotheses therefore provide a straightforward sound abstract classifier $C^\sharp : A \to \{-1, +1, \top\}$ defined as follows: $C^\sharp(a) \triangleq \mathrm{sign}_b^\sharp(D^\sharp(a))$. It turns out that these hypotheses entail the soundness of the robustness verifier.

Lemma 3.4. *If $P^\sharp$ is a sound abstract perturbation then $\langle C^\sharp, P^\sharp \rangle$ is a sound robustness verifier.*

If T is a test set for the classifier C then we may correctly assert that C is provably $q\%$-robust on T for the perturbation P when a sound abstract robustness verifier is able to check that C is robust on $q\%$ of the test samples in T. Of course, by soundness, this means that C is certainly robust on *at least* $q\%$ of the inputs in T, while on the

remaining $(100-q)\%$ of T we do not know: these could be spurious or real unrobust input vectors.

In order to design a sound abstract version of $D(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x_i}, \mathbf{x})$ we surely need sound approximations on A_1 of scalar multiplication and addition. We thus require a sound abstract scalar multiplication $\lambda a.za : A_1 \to A_1$, for any $z \in \mathbb{R}$, such that for all $a \in A_1$, $z\gamma(a) \subseteq \gamma(za)$, and a sound addition $+^\sharp : A_1 \times A_1 \to A_1$ such that for all $a, a' \in A_1$, $\gamma(a) + \gamma(a') \subseteq \gamma(a +^\sharp a')$, and we use $\sum_{i\in I}^\sharp a_i$ to denote an indexed abstract summation.

Linear Classifiers. Sound approximations of scalar multiplication and addition are enough for designing a sound robustness verifier for a linear classifier. As a preprocessing step, for a binary classifier $C(\mathbf{x}) = \mathrm{sign}([\sum_{i=1}^{N} \alpha_i y_i (\mathbf{x_i} \cdot \mathbf{x})] - b)$ which has been trained for the linear kernel, we preliminarly compute the hyperplane normal vector $\mathbf{w} \in \mathbb{R}^n$: for all $j \in [1, n]$, $\mathbf{w}_j \triangleq \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}_j$, so that for all $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{w} \cdot \mathbf{x} = \sum_{j=1}^{n} \mathbf{w}_j \mathbf{x}_j = \sum_{i=1}^{N} \alpha_i y_i (\mathbf{x_i} \cdot \mathbf{x})$. Thus, $C(\mathbf{x}) = \mathrm{sign}([\sum_{j=1}^{n} \mathbf{w}_j \mathbf{x}_j] - b)$ is the linear classifier in primal form, whose robustness can be abstractly verified by resorting to just sound abstract scalar multiplication and addition on A_1. The noteworthy advantage of abstracting a classifier in primal form is that each component of the input vector $\mathbf{x}$ occurs just once in $\mathrm{sign}([\sum_{j=1}^{n} \mathbf{w}_j \mathbf{x}_j] - b)$, while in the dual form $\mathrm{sign}([\sum_{i=1}^{N} \alpha_i y_i (\mathbf{x_i} \cdot \mathbf{x})] - b)$ each component $\mathbf{x}_j$ occurs exactly N times (one for each support vector), so that a precise abstraction of this latter dual form should be able to represent the correlation between (the many) multiple occurrences of each $\mathbf{x}_j$.

Nonlinear Classifiers. Let us consider a nonlinear kernel binary classifier $C(\mathbf{x}) = \mathrm{sign}(D(\mathbf{x}) - b)$, where $D(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x_i}, \mathbf{x})$ and $\{\mathbf{x_i}\}_{i=1}^{N} \subseteq \mathbb{R}^n$ is the set of support vectors for the kernel function k. Thus, what we additionally need here is a sound abstract kernel function $k^\sharp : \mathbb{R}^n \times A_n \to A_1$ such that for any support vector $\mathbf{x_i}$ and $a \in A_n$, $\{k(\mathbf{x_i}, \mathbf{x}) \mid \mathbf{x} \in \gamma(a)\} \subseteq \gamma(k^\sharp(\mathbf{x_i}, a))$. Let us consider the polynomial and RBF kernels.

For a d-polynomial kernel $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^d$, we need sound approximations of the unary dot product $\lambda \mathbf{y}. \mathbf{x} \cdot \mathbf{y} : \mathbb{R}^n \to \mathbb{R}$, for any given $\mathbf{x} \in \mathbb{R}^n$, and of the d-power function $(\cdot)^d : \mathbb{R} \to \mathbb{R}$. Of course, a sound nonrelational approximation of $\lambda \mathbf{y}. \mathbf{x} \cdot \mathbf{y} = \sum_{j=1}^{n} \mathbf{x}_j \mathbf{y}_j$ can be obtained simply by using sound abstract scalar multiplication and addition on A_1. Moreover, a sound abstract binary multiplication provides a straightforward definition of a sound abstract d-power function $(\cdot)^{d^\sharp} : A_1 \to A_1$. If $*^\sharp : A_1 \times A_1 \to A_1$ is a sound abstract multiplication such that for all $a, a' \in A_1$, $\gamma(a) * \gamma(a') \subseteq \gamma(a *^\sharp a')$, then a sound abstract d-power procedure can be defined simply by iterating the abstract multiplication $*^\sharp$.

For the RBF kernel $k(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x}-\mathbf{y}\|_2^2} = e^{-\gamma(\mathbf{x}-\mathbf{y})\cdot(\mathbf{x}-\mathbf{y})}$, for some $\gamma > 0$, we need sound approximations of the self-dot product $\lambda \mathbf{x}. \mathbf{x} \cdot \mathbf{x} : \mathbb{R}^n \to \mathbb{R}$, which is the squared Euclidean distance, and of the exponential $e^x : \mathbb{R} \to \mathbb{R}$. Let us observe that sound abstract addition and multiplication induce a sound nonrelational approximation of the self-dot product: for all $\langle a_1, ..., a_n \rangle \in A_n$, $\langle a_1, ..., a_n \rangle \cdot^\sharp \langle a_1, ..., a_n \rangle \triangleq \sum_{j=1}^{\sharp n} a_j *^\sharp a_j$. Finally, we require a sound abstract exponential $e^{\sharp(\cdot)} : A_1 \to A_1$ such that for all $a \in A_1$, $\{e^x \mid x \in \gamma(a)\} \subseteq \gamma(e^{\sharp a})$.

Abstract Multi-Classification. Let us consider multiclass classification for a set of labels $L = \{y_1, ..., y_m\}$, with $m > 2$. It turns out that the multi-classification approaches based on a reduction to multiple binary classifications such as ovr and ovo introduce a further approximation in the abstraction process, because these reduction strategies need to be soundly approximated.

Let us first consider the ovr strategy and, for all $j \in [1, m]$, let $C_{j,\bar{j}} : X \to \mathbb{R}$ denote the binary scoring classifier of y_j-versus-rest where $C_{j,\bar{j}}(\mathbf{x}) \triangleq D_j(\mathbf{x}) - b_j$. In order to have a sound approximation of ovr multi-classification, besides having m sound abstract classifiers $C^\sharp_{j,\bar{j}} : A_n \to A_1$ such that for all $a \in A_n$, $\{C_{j,\bar{j}}(\mathbf{x}) \in \mathbb{R} \mid \mathbf{x} \in \gamma(a)\} \subseteq \gamma(C^\sharp_{j,\bar{j}}(a))$, we need an abstract maximum function $\max^\sharp : (A_1)^m \to \{1, ..., m, \top\}$ which is sound, namely, if $(a_1, ..., a_m) \in (A_1)^m$ and $(z_1, ..., z_m) \in \gamma(a_1) \times ... \times \gamma(a_m)$ then $\max^\sharp(a_1, ..., a_m) \neq \top \Rightarrow \max(z_1, ..., z_m) \in \gamma(a_{\max^\sharp(a_1,...,a_m)})$ holds. Clearly, as soon as the abstract function $\max^\sharp$ outputs $\top$, this abstract multi-classification scheme is inconclusive.

Example 3.5 Let $m = 3$ and assume that an ovr multi-classifier M_{ovr} is robust on $\mathbf{x}$ for some adversarial region P as a consequence of the following ranges of scores: for all $\mathbf{x}' \in P(\mathbf{x})$, $-0.5 \leq C_{1,\bar{1}}(\mathbf{x}') \leq -0.2$, $3.5 \leq C_{2,\bar{2}}(\mathbf{x}') \leq 4$ and $2 \leq C_{3,\bar{3}}(\mathbf{x}') \leq 3.2$. In fact, since the least score of $C_{2,\bar{2}}$ on the region $P(\mathbf{x})$ is greater than the greatest scores of $C_{1,\bar{1}}$ and $C_{3,\bar{3}}$ on $P(\mathbf{x})$, these ranges imply that for all $\mathbf{x}' \in P(\mathbf{x})$, $M_{\text{ovr}}(\mathbf{x}') = y_2$. However, even in this advantageous scenario, on the abstract side we could not be able to infer that $C_{2,\bar{2}}$ always prevails over $C_{1,\bar{1}}$ and $C_{3,\bar{3}}$. For example, for the interval abstraction, some interval binary classifiers for a sound perturbation $P^\sharp(\mathbf{x})$ could output the following sound intervals: $C^\sharp_{1,\bar{1}}(P^\sharp(\mathbf{x})) = [-1, -0.1]$, $C^\sharp_{2,\bar{2}}(P^\sharp(\mathbf{x})) = [3.4, 4.2]$ and $C^\sharp_{3,\bar{3}}(P^\sharp(\mathbf{x})) = [1.5, 3.5]$. In this case, despite that each abstract binary classifier $C^\sharp_{i,\bar{i}}$ is able to prove that $C_{i,\bar{i}}$ is robust on $\mathbf{x}$ for P (because the output intervals do not include 0), the ovr strategy here does not allow to conclude that the multi-classifier M_{ovr} is robust on $\mathbf{x}$, because the lower bound 3.4 of the interval approximation provided by $C^\sharp_{2,\bar{2}}$ is not above the interval upper bound 3.5 of $C^\sharp_{3,\bar{3}}$. In such a case, a sound abstract multi-classifier based on ovr cannot prove the robustness of M_{ovr} for $P(\mathbf{x})$. □

Let us turn to the ovo approach which relies on $m(m-1)/2$ binary classifiers $C_{\{i,j\}} : X \to \{i, j\}$. Let us assume that for all the pairs $i \neq j$, a sound abstract binary classifier $C^\sharp_{\{i,j\}} : A \to \{y_i, y_j, \top\}$ is defined. Then, an abstract ovo multi-classifier $M^\sharp_{\text{ovo}} : A \to \wp(L)$ can be defined as follows. For all $i \in \{1, ..., m\}$ and $a \in A$, let $\text{votes}^\sharp(a, y_i) \in \text{Int}_{\mathbb{N}}$ be an interval of nonnegative integers used by the following abstract voting procedure AV, where $+^{\text{Int}}$ denotes standard interval addition:

forall $i \in [1, m]$ **do** $\text{votes}^\sharp(a, y_i) := [0, 0]$;
forall $i, j \in [1, m]$ **s.t.** $i \neq j$ **do**
  if $C^\sharp_{\{i,j\}}(a) = y_i$ **then** $\text{votes}^\sharp(a, y_i) := \text{votes}^\sharp(a, y_i) +^{\text{Int}} [1, 1]$; (2)
  elseif $C^\sharp_{\{i,j\}}(a) = y_j$ **then** $\text{votes}^\sharp(a, y_j) := \text{votes}^\sharp(a, y_j) +^{\text{Int}} [1, 1]$;
  else $\text{votes}^\sharp(a, y_i) := \text{votes}^\sharp(a, y_i) +^{\text{Int}} [0, 1]$; $\text{votes}^\sharp(a, y_j) := \text{votes}^\sharp(a, y_j) +^{\text{Int}} [0, 1]$;

Let us notice that the last else branch is taken when $C^\sharp_{\{i,j\}}(a) = \top$, meaning that the abstract classifier $C^\sharp_{\{i,j\}}(a)$ is not able to decide between y_i and y_j, so that in order to

preserve the soundness of the abstract voting procedure, we need to increment just the upper bounds of the interval ranges of votes for both classes y_i and y_j while their lower bounds are left unchanged. Let us denote $\mathrm{votes}^\sharp(a, y_i) = [v_i^{\min}, v_i^{\max}]$. Hence, at the end of the AV procedure, $[v_i^{\min}, v_i^{\max}]$ provides an interval approximation of concrete votes as follows:

$$|\{j \neq i \mid \forall \mathbf{x} \in \gamma(a).\, C_{\{i,j\}}(\mathbf{x}) = i\}| \geq v_i^{\min},$$
$$|\{j \neq i \mid \exists \mathbf{x} \in \gamma(a).\, C_{\{i,j\}}(\mathbf{x}) = i\}| \leq v_i^{\max}.$$

The corresponding abstract multi-classifier is then defined as follows:

$$M^\sharp_{\text{ovo}}(a) \triangleq \{y_i \in L \mid \forall j \neq i.\, v_j^{\min} \leq v_i^{\max}\}.$$

Hence, one may have an intuition for this definition by considering that a class y_i is not in $M^\sharp_{\text{ovo}}(a)$ when there exists a different class y_k whose lower bound of votes is certainly strictly greater than the upper bound of votes for y_i. For example, for $m = 4$, if $\mathrm{votes}^\sharp(a, y_1) = [4, 4]$, $\mathrm{votes}^\sharp(a, y_2) = [0, 2]$, $\mathrm{votes}^\sharp(a, y_3) = [4, 5]$, $\mathrm{votes}^\sharp(a, y_4) = [1, 3]$ then $M^\sharp_{\text{ovo}}(a) = \{y_1, y_3\}$.

Example 3.6 Assume that $m = 3$ and for all $\mathbf{x}' \in P(\mathbf{x})$, $M_{\text{ovo}}(\mathbf{x}') = \{y_3\}$ because we have that $\mathrm{argmax}_{i=1,2,3}\, \mathrm{votes}(\mathbf{x}', y_i) = \{3\}$. This means that a draw never happens for M_{ovo}, so that for all $\mathbf{x}' \in P(\mathbf{x})$, $C_{\{1,3\}}(\mathbf{x}') = y_3$ and $C_{\{2,3\}}(\mathbf{x}') = y_3$ certainly hold (because $m = 3$). Let us also assume that $\{C_{\{1,2\}}(\mathbf{x}') \mid \mathbf{x}' \in P(\mathbf{x})\} = \{y_1, y_2\}$. Then, for a sound abstract perturbation $P^\sharp(\mathbf{x})$, we necessarily have that $C^\sharp_{\{1,2\}}(P^\sharp(\mathbf{x})) = \top$. If we assume that $C^\sharp_{\{1,3\}}(P^\sharp(\mathbf{x})) = y_3$ and $C^\sharp_{\{2,3\}}(P^\sharp(\mathbf{x})) = \top$ then we have that $M^\sharp_{\text{ovo}}(P^\sharp(\mathbf{x})) = \{y_1, y_2, y_3\}$ because $\mathrm{votes}(P^\sharp(\mathbf{x}), y_1) = [0, 1]$, $\mathrm{votes}(P^\sharp(\mathbf{x}), y_2) = [0, 2]$ and $\mathrm{votes}(P^\sharp(\mathbf{x}), y_3) = [1, 2]$. Therefore, in this case, $M^\sharp_{\text{ovo}}$ is not able to prove the robustness of M_{ovo} on $\mathbf{x}$. Let us notice that the source of imprecision in this multi-classification is confined to the binary classifier $C^\sharp_{\{2,3\}}$ rather than the abstract voting AV strategy. In fact, if we have that $C^\sharp_{\{1,3\}}(P^\sharp(\mathbf{x})) = \{y_3\}$ and $C^\sharp_{\{2,3\}}(P^\sharp(\mathbf{x})) = \{y_3\}$ then $M^\sharp_{\text{ovo}}(P^\sharp(\mathbf{x})) = \{y_3\}$, thus proving the robustness of M. □

Lemma 3.7. *Let M_{ovo} be an ovo multi-classifier based on binary classifiers $C_{\{i,j\}}$. If the abstract ovo multi-classifier $M^\sharp_{\text{ovo}}$ is based on sound abstract binary classifiers $C^\sharp_{\{i,j\}}$ then $M^\sharp_{\text{ovo}}$ is sound for M_{ovo}.*

In our experimental evaluation we will follow the ovo approach for concrete multi-classification, which is standard for SVMs [14], and consequently we will use this abstract ovo multi-classifier for robustness verification.

On Completeness. Let $C : X \to \{-1, +1\}$ be a binary classifier, $P : X \to \wp(X)$ a perturbation and $C^\sharp : A \to \{-1, +1, \top\}$, $P^\sharp : X \to A$ be a corresponding sound abstract binary classifier and perturbation on some abstraction A.

Definition 3.8 (Complete Abstract Classifiers and Robustness Verifiers). $C^\sharp$ is *complete* for C when for all $a \in A$, $C^\sharp(a) = \top \Rightarrow \exists \mathbf{x}, \mathbf{x}' \in \gamma(a).\, C(\mathbf{x}) \neq C(\mathbf{x}')$.
$\langle C^\sharp, P^\sharp \rangle$ is a (sound and) *complete robustness verifier* for C w.r.t. P when for all $\mathbf{x} \in X$, $C^\sharp(P^\sharp(\mathbf{x})) = C(\mathbf{x})$ iff $\mathrm{Rob}(C, \mathbf{x}, P)$. □

Complete abstract classifiers can be obtained for linear binary classifiers once these linear classifiers are in primal form and the abstract operations are exact. We therefore consider a linear binary classifier in primal form $C_{\mathrm{pr}}(\mathbf{x}) \triangleq \mathrm{sign}_b(\sum_{j=1}^{n} \mathbf{w}_j \pi_j(\mathbf{x}))$ and an abstraction A of $\wp(X)$ with concretization $\gamma : A \to \wp(X)$. Let us consider the following exactness conditions for the abstract functions on A needed for abstracting C_{pr} and the perturbation P:

(E_1) Exact projection $\pi_j^\sharp$: For all $j \in [1, n]$ and $a \in A_n$, $\gamma(\pi_j^\sharp(a)) = \pi_j(\gamma(a))$;
(E_2) Exact scalar multiplication: For all $z \in \mathbb{R}$ and $a \in A_1$, $\gamma(za) = z\gamma(a)$;
(E_3) Exact scalar addition $+^\sharp$: For all $a, a' \in A_1$, $\gamma(a +^\sharp a') = \gamma(a) + \gamma(a')$;
(E_4) Exact $\mathrm{sign}_b^\sharp$: For all $b \in \mathbb{R}$, $a \in A_1$, $(\forall x \in \gamma(a).\, \mathrm{sign}_b(x) = s) \Rightarrow \mathrm{sign}_b^\sharp(a) = s$;
(E_5) Exact perturbation $P^\sharp$: For all $\mathbf{x} \in X$, $\gamma(P^\sharp(\mathbf{x})) = P(\mathbf{x})$.

Then, it turns out that the abstract classifier $C_{\mathrm{pr}}^\sharp(a) \triangleq \mathrm{sign}_b^\sharp(\sum_{j=1}^{\sharp n} \mathbf{w}_j \pi_j^\sharp(a))$ is complete and induces a complete robustness verifier.

Lemma 3.9. *Under hypotheses* (E_1)-(E_5)*, $C_{\mathrm{pr}}^\sharp$ is (sound and) complete for C_{pr} and $\langle C_{\mathrm{pr}}^\sharp, P^\sharp \rangle$ is a complete robustness verifier for C_{pr} w.r.t. P.*

Let us now focus on multi-classification. It turns out that completeness does not scale from binary to multi-classification, that is, even if all the abstract binary classifiers are assumed to be complete, the corresponding abstract multi-classification could lose the completeness. This loss is not due to the abstraction of the binary classifiers, but it is an intrinsic issue of a multi-classification approach based on binary classification. Let us show how this loss for ovr and ovo can happen through some examples.

Example 3.10 Consider $L = \{y_1, y_2, y_3\}$ and assume that for two different inputs $\mathbf{x}, \mathbf{x}' \in X$, the scoring ovr binary classifiers $C_{i,\bar{i}}$ are as follows: $C_{1,\bar{1}}(\mathbf{x}) = 3$, $C_{2,\bar{2}}(\mathbf{x}) = -1$, $C_{3,\bar{3}}(\mathbf{x}) = 2$, $C_{1,\bar{1}}(\mathbf{x}') = 1$, $C_{2,\bar{2}}(\mathbf{x}') = -1$, $C_{3,\bar{3}}(\mathbf{x}) = 0.5$. Hence, $M_{\mathrm{ovr}}(\mathbf{x}) = M_{\mathrm{ovr}}(\mathbf{x}') = \{y_1\}$, meaning that M_{ovr} is robust on $\mathbf{x}$ for a perturbation $P(\mathbf{x}) = \{\mathbf{x}, \mathbf{x}'\}$. However, it turns out that the mere collecting abstraction of binary classifiers $C_{i,\bar{i}}$, although being trivially complete according to Definition 3.8, may well lead to a (sound but) incomplete multi-classification. In fact, even if we consider no abstraction of sets of vectors/scalars and an abstract binary classifier is simply defined by a collecting abstraction $C_{i,\bar{i}}^\sharp(Y) \triangleq \{C_{i,\bar{i}}(\mathbf{x}) \in \mathbb{R} \mid \mathbf{x} \in Y\}$, then we have that while each $C_{i,\bar{i}}^\sharp$ is complete the corresponding abstract ovr multi-classifier turns out to be sound but not complete. In our example, we have that: $C_{1,\bar{1}}^\sharp(P(\mathbf{x})) = \{1, 3\}$, $C_{2,\bar{2}}^\sharp(P(\mathbf{x})) = \{-1\}$, $C_{3,\bar{3}}^\sharp(P(\mathbf{x})) = \{0.5, 2\}$. Hence, the ovr strategy can only derive that both y_1 and y_2 are feasible classes for $P(\mathbf{x})$, namely, $M_{\mathrm{ovr}}^\sharp(\{\mathbf{x}, \mathbf{x}'\}) = \{y_1, y_2\}$, meaning that $M_{\mathrm{ovr}}^\sharp$ cannot prove the robustness of M. □

The above example shows that the loss of relational information between input vectors and corresponding scores is an unavoidable source of incompleteness when abstracting ovr multi-classification. An analogous incompleteness happens in ovo multi-classification.

Example 3.11 Consider $L = \{y_1, y_2, y_3, y_4, y_5\}$ and assume that for some $\mathbf{x}, \mathbf{x}' \in X$, the ovo binary classifiers $C_{\{i,j\}}$ give the following outputs:

	$C_{\{1,2\}}$	$C_{\{1,3\}}$	$C_{\{1,4\}}$	$C_{\{1,5\}}$	$C_{\{2,3\}}$	$C_{\{2,4\}}$	$C_{\{2,5\}}$	$C_{\{3,4\}}$	$C_{\{3,5\}}$	$C_{\{4,5\}}$
$\mathbf{x}$	y_1	y_1	y_1	y_5	y_2	y_2	y_5	y_3	y_3	y_4
$\mathbf{x}'$	y_1	y_1	y_4	y_1	y_2	y_4	y_2	y_3	y_5	y_5

so that $M_{\text{ovo}}(\mathbf{x}) = M_{\text{ovo}}(\mathbf{x}') = \{y_1\}$, meaning that M_{ovo} is robust on $\mathbf{x}$ for the perturbation $P(\mathbf{x}) = \{\mathbf{x}, \mathbf{x}'\}$. Similarly to Example 3.10, the collecting abstractions of binary classifiers $C_{\{i,j\}}$ are trivially complete but define a (sound but) incomplete multi-classification. In fact, even with no numerical abstraction, if we consider the abstract collecting binary classifiers $C^{\sharp}_{\{i,j\}}(Y) \triangleq \{C_{\{i,j\}}(\mathbf{x}) \mid \mathbf{x} \in Y\}$ then we have that:

	$C_{\{1,2\}}$	$C_{\{1,3\}}$	$C_{\{1,4\}}$	$C_{\{1,5\}}$	$C_{\{2,3\}}$	$C_{\{2,4\}}$	$C_{\{2,5\}}$	$C_{\{3,4\}}$	$C_{\{3,5\}}$	$C_{\{4,5\}}$
$P(\mathbf{x})$	$\{y_1\}$	$\{y_1\}$	$\{y_1, y_4\}$	$\{y_1, y_5\}$	$\{y_2\}$	$\{y_2, y_4\}$	$\{y_2, y_5\}$	$\{y_3\}$	$\{y_3, y_5\}$	$\{y_4, y_5\}$

Thus, the ovo voting for $P(\mathbf{x})$ in order to be sound necessarily has to assign 4 votes to both classes y_1 and y_5, meaning that $M^{\sharp}_{\text{ovo}}(P(\mathbf{x})) = \{y_1, y_5\}$. As a consequence, $M^{\sharp}_{\text{ovr}}$ cannot prove the robustness of M_{ovo}. Here again, this is a consequence of the collecting abstraction which looses the relational information between input vectors and corresponding classes, and therefore is an ineluctable source of incompleteness when abstracting ovo multi-classification. □

Let us observe that when all the abstract binary classifiers $C^{\sharp}_{\{i,j\}}$ are complete, then in the abstract voting procedure AV defined by (2), for all $\text{votes}^{\sharp}(a, y_i) = [v_i^{\min}, v_i^{\max}]$, we have that $|\{j \neq i \mid \exists \mathbf{x} \in \gamma(a).\, C_{\{i,j\}}(\mathbf{x}) = i\}| = v_i^{\max}$ holds, meaning that the hypothesis of completeness of abstract binary classifiers strengthens the upper bound $v_i^{\max}$ to a precise equality, although this is not enough for preserving the completeness.

4 Numerical Abstractions for Classifiers

Interval Abstraction. The n-dimensional interval abstraction domain Int_n is simply defined as a nonrelational product of Int, i.e., $\text{Int}_n \triangleq \text{Int}^n$ (with $\text{Int}_1 = \text{Int}$), where $\gamma_{\text{Int}_n} : \text{Int}_n \to \wp(\mathbb{R}^n)$ is defined by $\gamma_{\text{Int}_n}(I_1, ..., I_n) \triangleq \times_{i=1}^{n} \gamma_{\text{Int}}(I_i)$, and, by a slight abuse of notation, this concretization map will be denoted simply by γ. In order to abstract linear and nonlinear classifiers, we will use the following standard interval operations based on real arithmetic operations.

- Projection $\pi_j : \text{Int}_n \to \text{Int}$ defined by $\pi_j(I_1, ..., I_n) \triangleq I_j$, which is trivially exact because Int_n is nonrelational.
- Scalar multiplication $\lambda\mathbf{I}.z\mathbf{I} : \text{Int}_n \to \text{Int}_n$, with $z \in \mathbb{R}$, is defined as componentwise extension of scalar multiplication $\lambda I.zI : \text{Int}_1 \to \text{Int}_1$ given by: $z\bot = \bot$ and $z[l, u] \triangleq [zl, zu]$, where $z(\pm\infty) = \pm\infty$ for $z \neq 0$ and $0(\pm\infty) = 0$. This is an exact abstract scalar multiplication, i.e., $\{z\mathbf{x} \mid \mathbf{x} \in \gamma(\mathbf{I})\} = \gamma(z\mathbf{I})$ holds.
- Addition $+^{\sharp} : \text{Int}_n \times \text{Int}_n \to \text{Int}_n$ is defined as componentwise extension of standard interval addition, that is, $\bot +^{\sharp} I = \bot = I +^{\sharp} \bot$, $[l_1, u_1] +^{\sharp} [l_2, u_2] = [l_1 + l_2, u_1 + u_2]$. This abstract interval addition is exact, i.e., $\{\mathbf{x}_1 + \mathbf{x}_2 \mid \mathbf{x}_i \in \gamma(\mathbf{I}_i)\} = \gamma(\mathbf{I}_1 +^{\sharp} \mathbf{I}_2)$ holds.

– One-dimensional multiplication $*^\sharp : \mathrm{Int}_1 \times \mathrm{Int}_1 \to \mathrm{Int}_1$ is enough for our purposes, whose definition is standard: $\bot *^\sharp I = \bot = I *^\sharp \bot$, $[l_1, u_1] *^\sharp [l_2, u_2] = [\min(l_1l_2, l_1u_2, u_1l_2, u_1u_2), \max(l_1l_2, l_1u_2, u_1l_2, u_1u_2)]$. As a consequence of the completeness of real numbers, this abstract interval multiplication is exact, i.e., $\{x_1x_2 \mid x_i \in \gamma(I_i)\} = \gamma(I_1 *^\sharp I_2)$.

It is worth remarking that since all these abstract functions on real intervals are exact and real intervals have the abstraction map, it turns out that all these abstract functions are the best correct approximations on intervals of the corresponding concrete functions.

For the exponential function $e^x : \mathbb{R} \to \mathbb{R}$ used by RBF kernels, let us consider a generic real function $f : \mathbb{R} \to \mathbb{R}$ which is assumed to be continuous and monotonically either increasing ($x \leq y \Rightarrow f(x) \leq f(y)$) or decreasing ($x \leq y \Rightarrow f(x) \geq f(y)$). Its collecting lifting $f^c : \wp(\mathbb{R}) \to \wp(\mathbb{R})$ is approximated on the interval abstraction by the abstract function $f^\sharp : \mathrm{Int}_1 \to \mathrm{Int}_1$ defined as follows: for all possibly unbounded intervals $[l, u]$ with $l, u \in \mathbb{R} \cup \{-\infty, +\infty\}$,

$$f_i([l,u]) \triangleq \inf\{f(x) \in \mathbb{R} \mid x \in \gamma([l,u])\} \in \mathbb{R} \cup \{-\infty\}$$
$$f_s([l,u]) \triangleq \sup\{f(x) \in \mathbb{R} \mid x \in \gamma([l,u])\} \in \mathbb{R} \cup \{+\infty\}$$
$$f^\sharp([l,u]) \triangleq [\min(f_i([l,u]), f_s([l,u])), \max(f_i([l,u]), f_s([l,u])] \qquad f^\sharp(\bot) \triangleq \bot$$

Therefore, for bounded intervals $[l, u]$ with $l, u \in \mathbb{R}$, $f^\sharp([l, u]) = [\min(f(l), f(u)), \max(f(l), f(u))]$. As a consequence of the hypotheses of continuity and monotonicity of f, it turns out that this abstract function $f^\sharp$ is exact, i.e., $\{f(x) \in \mathbb{R} \mid x \in \gamma([l, u])\} = \gamma(f^\sharp([l, u]))$ holds, and it is the best correct approximation on intervals of f^c.

Reduced Affine Arithmetic Abstraction. Even if all the abstract functions of the interval abstraction are exact, it is well known that the compositional abstract evaluation of an inductively defined expression *exp* on Int can be imprecise due to the so-called dependency problem, meaning that if the syntactic expression *exp* includes multiple occurrences of a variable x and the abstract evaluation of *exp* is performed by structural induction on *exp*, then each occurrence of x in *exp* is taken independently from the others and this can lead to a significant loss of precision in the output interval. This loss of precision may happen both for addition and multiplication of intervals. For example, the abstract compositional evaluations of the simple expressions $x - x$ and $x * x$ on an input interval $[-c, c]$, with $c \in \mathbb{R}_{>0}$, yield, resp., $[-2c, 2c]$ and $[-c^2, c^2]$, rather than the exact results $[0, 0]$ and $[0, c^2]$. This dependency problem can be a significant source of imprecision for the interval abstraction of a polynomial SVM classifier $C(\mathbf{x}) = \mathrm{sign}([\sum_{i=1}^{N} \alpha_i y_i (\sum_{j=1}^{n} (\mathbf{y_i})_j \mathbf{x}_j + c)^d] - b)$, where each attribute $\mathbf{x}_j$ of an input vector $\mathbf{x}$ occurs for each support vector $\mathbf{y_i}$. The classifiers based on RBF kernels suffer from an analogous issue.

Affine Forms. Affine arithmetic [35,36] mitigates this dependency problem of the non-relational interval abstraction. An interval $[l, u] \in \mathrm{Int}$ which approximates the range of some variable x is represented by an *affine form* (AF) $\hat{x} = a_0 + a_1\epsilon_x$, where $a_0 = (l + u)/2$, $a_1 = (u - l)/2$ and ϵ_x is a symbolic (or "noise") real variable ranging in $[-1, 1] \in \mathrm{Int}$ which explicitly represents a dependence from the parameter x. This

solves the dependency problem for a linear expression such as $x - x$ because the interval $[-c, c]$ for x is represented by $0 + c\epsilon_x$ so that the compositional evaluation of $x - x$ for $0 + c\epsilon_x$ becomes $(0 + c\epsilon_x) - (0 + c\epsilon_x) = 0$, while for nonlinear expressions such as $x * x$, an approximation is still needed.

In general, the domain AF_k of affine forms with $k \geq 1$ noise variables consists of affine forms $\hat{a} = a_0 + \sum_{i=1}^{k} a_i\epsilon_i$, where $a_i \in \mathbb{R}$ and each ϵ_i represents either an external dependence from some input variable or an internal approximation dependence due to a nonlinear operation. An affine form $\hat{a} \in \mathrm{AF}_k$ can be abstracted to a real interval in Int, as given by a map $\alpha_{\mathrm{Int}} : \mathrm{AF}_k \to \mathrm{Int}$ defined as follows: for all $\hat{a} = a_0 + \sum_{i=1}^{k} a_i\epsilon_i \in \mathrm{AF}_k$, $\alpha_{\mathrm{Int}}(\hat{a}) \triangleq [c_{\hat{a}} - \mathrm{rad}(\hat{a}), c_{\hat{a}} + \mathrm{rad}(\hat{a})]$, where $c_{\hat{a}} \triangleq a_0$ and $\mathrm{rad}(\hat{e}) \triangleq \sum_{i=1}^{k} |a_i|$ are called, resp., center and radius of the affine form $\hat{a}$. This, in turn, defines the interval concretization $\gamma_{\mathrm{AF}_k} : \mathrm{AF}_k \to \mathbb{R}$ given by $\gamma_{\mathrm{AF}_k}(\hat{a}) \triangleq \gamma_{\mathrm{Int}\to\mathbb{R}}(\alpha_{\mathrm{Int}}(\hat{a}))$. Vectors of affine forms may be used to represent zonotopes, which are center-symmetric convex polytopes and have been used to design an abstract domain for static program analysis [13] endowed with abstract functions, joins and widening.

Reduced Affine Forms. It turns out that affine forms are exact for linear operations, namely additions and scalar multiplications. If $\hat{a}, \hat{b} \in \mathrm{AF}_k$ and $c \in \mathbb{R}$ then abstract additions and scalar multiplications are defined as follows: $\hat{a} +^\sharp \hat{b} \triangleq (a_0 + b_0) + \sum_{j=1}^{k}(a_j + b_j)\epsilon_j$ and $c\hat{a} \triangleq ca_0 + \sum_{j=1}^{k} ca_j\epsilon_j$. They are exact, namely, $\{x + y \in \mathbb{R} \mid x \in \gamma_{\mathrm{AF}_k}(\hat{a}), y \in \gamma_{\mathrm{AF}_k}(\hat{b})\} = \gamma_{\mathrm{AF}_k}(\hat{a} +^\sharp \hat{b})$ and $c\gamma_{\mathrm{AF}_k}(\hat{a}) = \gamma_{\mathrm{AF}_k}(c\hat{a})$.

For nonlinear operations, in particular multiplication, in general the result cannot be represented exactly by an affine form. Then, the standard strategy for defining the multiplication of affine forms is to approximate the precise result by adding a fresh noise symbol whose coefficient is typically computed by a Taylor or Chebyshev approximation of the nonlinear part of the multiplication (cf. [13, Section 2.1.5]). Similarly, for the exponential function used in RBF kernels, an algorithm for computing an affine approximation of the exponential e^x evaluated on an affine form $\hat{x}$ for the exponent x is given in [35, Section 3.11] and is based on an optimal Chebyshev approximation (that is, w.r.t. L_∞ distance) of the exponential which introduces a fresh noise symbol. However, the need of injecting a fresh noise symbol for each nonlinear operation raises a critical space and time complexity issue for abstracting polynomial and RBF classifiers, because this would imply that a new but useless noise symbol should be added for each support vector. For example, for a 2-polynomial classifier, we need to approximate a square operation $x * x$ for each of the N support vectors, and a blind usage of abstract multiplication for affine forms would add N different and useless noise symbols. This drawback would be even worse for d-polynomial classifers with $d > 2$, while an analogous critical issue would happen for RBF classifiers. This motivates the use of so-called *reduced affine forms* (RAFs), which have been introduced in [20] as a remedy for the increase of noise symbols due to nonlinear operations and still allow us to keep track of correlations between the components of the input vectors of classifiers.

A reduced affine form $\tilde{a} \in \mathrm{RAF}_k$ of length $k \geq 1$ is defined as a sum of a standard affine form in AF_k with a specific rounding noise ϵ_a which accumulates all the errors introduced by nonlinear operations. Thus, $\mathrm{RAF}_k \triangleq \{a_0 + \sum_{j=1}^{k} a_j\epsilon_j + a_r\epsilon_a \mid$

$a_0, a_1, ..., a_k \in \mathbb{R}$, $a_r \in \mathbb{R}_{\geq 0}\}$. The key point is that the length of $\tilde{a} \in \mathrm{RAF}_k$ remains unchanged during the whole abstract computation and $a_r \in \mathbb{R}_{\geq 0}$ is the radius of the accumulative error of approximating all nonlinear operations during abstract computations. Of course, each $\tilde{a} \in \mathrm{RAF}_k$ can be viewed as a standard affine form in AF_{k+1} and this allows us to define the interval concretization $\gamma_{\mathrm{RAF}_k}(\tilde{a})$ and the linear abstract operations of addition and scalar multiplication of RAFs simply by considering them as standard affine forms. In particular, linear abstract operations in RAF_k are exact w.r.t. interval concretization γ_{RAF_k}.

Nonlinear abstract operations, such as multiplication, must necessarily be approximated for RAFs. Several algorithms of abstract multiplication of RAFs are available, which differ in precision, approximation principle and time complexity, ranging from linear to quadratic complexities [33, Section 3]. Given $\tilde{a}, \tilde{b} \in \mathrm{RAF}_k$, we need to define an abstract multiplication $\tilde{a} *^\sharp \tilde{b} \in \mathrm{RAF}_k$ which is sound, namely, $\{xy \in \mathbb{R} \mid x \in \gamma_{\mathrm{RAF}_k}(\tilde{a}),\ y \in \gamma_{\mathrm{RAF}_k}(\tilde{b})\} \subseteq \gamma_{\mathrm{RAF}_k}(\tilde{a} *^\sharp \tilde{b})$, where it is worth pointing out that this soundness condition is given w.r.t. interval concretization γ_{RAF_k} and scalar multiplication. Time complexity is a crucial issue for using $*^\sharp$ in abstract polynomial and RBF kernels, because in these abstract classifiers at least an abstract multiplication must be used for each support vector, so that quadratic time algorithms in $O(k^2)$ cannot scale when the number of support vectors grows, as expected for realistic training datasets. We therefore selected a recent linear time algorithm by Skalna and Hladík [33] which is optimal in the following sense. Given $\tilde{a}, \tilde{b} \in \mathrm{RAF}_k$, we have that their concrete symbolic multiplication is as follows:

$$\begin{aligned}\tilde{a} * \tilde{b} &= (a_0 + \textstyle\sum_{j=1}^{k} a_j\epsilon_j + a_r\epsilon_a) * (b_0 + \sum_{j=1}^{k} b_j\epsilon_j + b_r\epsilon_b)\\ &= a_0b_0 + \textstyle\sum_{j=1}^{k}(a_0b_j + b_0a_j)\epsilon_j + (a_0b_r\epsilon_b + b_0a_r\epsilon_a) \ + f_{\tilde{a},\tilde{b}}(\epsilon_1, ..., \epsilon_k, \epsilon_a, \epsilon_b)\end{aligned}$$

where $f_{\tilde{a},\tilde{b}}(\epsilon_1, ..., \epsilon_k, \epsilon_a, \epsilon_b) \triangleq (\sum_{j=1}^{k} a_j\epsilon_j + a_r\epsilon_a)(\sum_{j=1}^{k} b_j\epsilon_j + b_r\epsilon_b)$. An abstract multiplication $*_e^\sharp$ on RAF_k can be defined as follows: if $R_{\max}, R_{\min} \in \mathbb{R}$ are, resp., the minimum and maximum of $\{f_{\tilde{a},\tilde{b}}(\mathbf{e}) \in \mathbb{R} \mid \mathbf{e} \in [-1, 1]^{k+2}\}$ then

$$\begin{aligned}\tilde{a} *_e^\sharp \tilde{b} \triangleq a_0b_0 + 0.5(R_{\max} + R_{\min}) + \textstyle\sum_{j=1}^{k}(a_0b_j + b_0a_j)\epsilon_j +\\ (|a_0|b_r + |b_0|a_r + 0.5(R_{\max} - R_{\min}))\epsilon_{a*b}\end{aligned}$$

where $0.5(R_{\max} + R_{\min})$ and $0.5(R_{\max} - R_{\min})$ are, resp., the center and the radius of the interval range of $f_{\tilde{a},\tilde{b}}(\epsilon_1, ..., \epsilon_k, \epsilon_a, \epsilon_b)$. As argued in [33, Proposition 3], this defines an optimal abstract multiplication of RAFs. Skalna and Hladík [33] put forward two algorithms for computing $R_{\max}$ and $R_{\min}$, one with $O(k)$ time bound and one in $O(k \log k)$: the $O(k)$ bound is obtained by relying on a linear time algorithm to find a median of a sequence of real numbers, while the $O(k \log k)$ algorithm is based on (quick)sorting that sequence of numbers. The details of these algorithms are here omitted and can be found in [33, Section 4]. In abstract interpretation terms, it turns out that this abstract multiplication algorithm $*_e^\sharp$ of RAFs provides the best approximation among the RAFs which correctly approximate the multiplication with the same coefficients for ϵ_1,...,ϵ_k of $\tilde{a} *_e^\sharp \tilde{b}$.

Finally, let us consider the exponential function e^x used in RBF kernels. The algorithm in [35, Section 3.11] for computing the affine form approximation of e^x and

based on Chebyshev approximation of e^x can be also applied when the exponent is represented by a RAF $\tilde{x} = x_0 + \sum_{j=1}^{k} x_j\epsilon_j + x_r\epsilon_x \in \mathrm{RAF}_k$, provided that the radius of the fresh noise symbol produced by computing $e^{\tilde{x}}$ is added to the coefficient of the rounding noise ϵ_x of $\tilde{x}$.

Floating Point Soundness. The interval and RAF abstractions and the corresponding abstract functions described above rely on precise real arithmetic on $\mathbb{R}$, in particular soundness and exactness of abstract functions depend on real arithmetic. These abstract functions may yield unsound results for floating point arithmetic such as the standard IEEE 754 [34]. These domains therefore need some suitable adjustments to make them "floating-point" sound [21], which are described in the full version of this paper [29].

5 Verifying SVM Classifiers

Perturbations. We consider robustness of SVM classifiers against a standard adversarial region defined by the L_∞ norm, as considered in Carlini and Wagner's robustness model [4] and used by Vechev et al. [9,31,32] in their verification framework. Given a generic classifier $C : X \to L$ and a constant $\delta \in \mathbb{R}_{>0}$, a L_∞ δ-perturbation of an input vector $\mathbf{x} \in \mathbb{R}^n$ is defined by $P_\delta^\infty(\mathbf{x}) \triangleq \{\mathbf{x}' \in X \mid \|\mathbf{x}' - \mathbf{x}\|_\infty \leq \delta\}$. Thus, if the space X consists of n-dimensional real vectors normalized in $[0, 1]$ (our datasets follow this standard) and $\delta \in (0, 1]$ then $P_\delta^\infty(\mathbf{x}) = \{\mathbf{x}' \in \mathbb{R}^n \mid \forall i.\ \mathbf{x}'_i \in [\mathbf{x}_i - \epsilon, \mathbf{x}_i + \epsilon] \cap [0, 1]\}$. Let us observe that, for all $\mathbf{x}$, $P_\delta^\infty(\mathbf{x})$ is an exact perturbation for intervals and therefore for RAFs as well (cf. (E_5)). The datasets of our experiments consist of $h \times w$ grayscale images (with 8 bits per pixel, i.e., the pixel depth allows 256 different gray intensities) where each image is represented as a normalized real vector in $[0, 1]^{hw}$ whose components encode the light values of pixels. Increasing (decreasing) the value of a vector component means brightening (darkening) that pixel, so that a brightening of $+0.01$ means $+2.55$ pixel depth. Hence, a perturbation $P_\delta^\infty(\mathbf{x})$ of an image $\mathbf{x}$ represents all the images where every possible subset of pixels is brightened or darkened up to δ.

We also consider robustness of image classifiers for the so-called *adversarial framing* on the border of images, which has been recently shown to represent an effective attack for deep convolutional networks [44]. Consider an image represented as a $h \times w$ matrix $M \in \mathbb{R}_{h,w}$ with normalized real values in $[0, 1]$. Given an integer framing thickness $t \in [1, \min(h, w)/2]$, the "occlude" t-framing perturbation of M is defined by

$$P_t^{\mathrm{frm}}(M) \triangleq \{M' \in \mathbb{R}_{h,w} \mid \forall i \in [t+1, h-t], j \in [w+1, w-t].\ M'_{i,j} = M_{i,j},\\ \forall i \notin [t+1, h-t], j \notin [w+1, w-t].\ M'_{i,j} \in [0, 1]\}.$$

This framing perturbation models the uniformly distributed random noise attack in [44]. Also in this case $P_t^{\mathrm{frm}}(M)$ is a perturbation which can be exactly represented by intervals and consequently by RAFs.

Linear Classifiers. As observed in Section 4, for the interval abstraction it turns out that all the abstract functions which are used in abstract linear binary classifiers in primal form are exact, so that, by Lemma 3.9, these abstract linear binary classifiers are complete. This completeness implies that there is no need to resort to the RAF abstraction for linear binary classifiers. However, as argued in Section 3, this completeness for

binary classifiers does not scale to multi-classification. Nevertheless, it is worth pointing out that for each binary classifier $C_{\{i,j\}}$ used in ovo multi-classification, since L_∞ and frame perturbations are exact for intervals, we have a complete robustness verifier for each $C_{\{i,j\}}$. As a consequence, this makes feasible to find adversarial examples of linear binary classifiers as follows. Let us consider a linear binary classifier in primal form $C(\mathbf{x}) = \mathrm{sign}([\sum_{j=1}^{n} \mathbf{w}_j \mathbf{x}_j] - b)$ and a perturbation P which is exact on intervals, i.e., for all $\mathbf{x}$, $P(\mathbf{x}) = \gamma_{\mathrm{Int}_n}(P^\sharp(\mathbf{x}))$, where $P^\sharp(\mathbf{x}) = \langle [l_1, u_1], ..., [l_n, u_n] \rangle) \in \mathrm{Int}_n$. Completeness of robustness linear verification means that if the interval abstraction $\sum_{j=1}^{\sharp n} \mathbf{w}_j [l_j, u_j]$ outputs an interval $[l, u] \in \mathrm{Int}_1$ such that $0 \in [l, u]$, then C is surely not robust on $\mathbf{x}$ for P. It is then easy to find two input vectors $\mathbf{y}, \mathbf{z} \in P(\mathbf{x})$ which provide a concrete counterexample to the robustness, namely such that $C(\mathbf{y}) \neq C(\mathbf{z})$. For all $i \in [1, n]$, if $\mathbf{y}_i \triangleq$ **if** $\mathrm{sign}(\mathbf{w}_i) \geq 0$ **then** u_i **else** l_i and $\mathbf{z}_i \triangleq$ **if** $\mathrm{sign}(\mathbf{w}_i) \geq 0$ **then** l_i **else** u_i then we have defined $\mathbf{y}, \mathbf{z} \in P(\mathbf{x})$ such that $\sum_{j=1}^{n} \mathbf{w}_j \mathbf{y}_j = u$ and $\sum_{j=1}^{n} \mathbf{w}_j \mathbf{z}_j = l$, so that $C(\mathbf{y}) =$ +1 and $C(\mathbf{z}) =$ –1. This pair of inputs $(\mathbf{y}, \mathbf{z})$ therefore represents the strongest adversarial example to the robustness of C on $\mathbf{x}$.

Nonlinear Classifiers. Let us first point out through an example that interval and RAF abstractions are incomparable for nonlinear operations.

Example 5.1 Consider the 2-polynomial in two variables $f(x_1, x_2) \triangleq (1 + 2x_1 - x_2)^2 - \frac{1}{4}(2 + x_1 + x_2)^2$, which could be thought of as a 2-polynomial classifier in $\mathbb{R}^2$. Assume that x_1 and x_2 range in the interval $[-1, 1]$. The abstract evaluation of f on the intervals $I_{x_1} = [-1, 1] = I_{x_2}$ is as follows:

$$\begin{aligned} f^\sharp_{\mathrm{Int}}(I_{x_1}, I_{x_2}) &= (1 + 2[-1, 1] - [-1, 1])^2 - \tfrac{1}{4}(2 + [-1, 1] + [-1, 1])^2 \\ &= [-2, 4]^2 - \tfrac{1}{4}[0, 4]^2 = [0, 16] + [-4, 0] = [-4, 16] \end{aligned}$$

On the other hand, for the RAF_2 abstraction we have that $\tilde{x}_1 = \epsilon_1$, $\tilde{x}_2 = \epsilon_2$ and the abstract evaluation of f is as follows:

$$\begin{aligned} f^\sharp_{\mathrm{RAF}_2}(\tilde{x}_1, \tilde{x}_2) &= (1 + 2\epsilon_1 - \epsilon_2)^2 - \tfrac{1}{4}(2 + \epsilon_1 + \epsilon_2)^2 \\ &= [1 + 0.5(R_{1\,\max} + R_{1\,\min}) + 4\epsilon_1 - 2\epsilon_2 + 0.5(R_{1\,\max} - R_{1\,\min})\epsilon_r] - \\ &\quad \tfrac{1}{4}[4 + 0.5(R_{2\,\max} + R_{2\,\min}) + 4\epsilon_1 + 4\epsilon_2 + 0.5(R_{2\,\max} - R_{2\,\min})\epsilon_r] \\ &\text{where} \quad R_{1\,\max} = \max((2\epsilon_1 - \epsilon_2)^2) = 9,\ R_{1\,\min} = \min((2\epsilon_1 - \epsilon_2)^2) = 0, \\ &\quad R_{2\,\max} = \max((\epsilon_1 + \epsilon_2)^2) = 4,\ R_{2\,\min} = \min((\epsilon_1 + \epsilon_2)^2) = 0 \\ &= [5.5 + 4\epsilon_1 - 2\epsilon_2 + 4.5\epsilon_r] - \tfrac{1}{4}[6 + 4\epsilon_1 + 4\epsilon_2 + 2\epsilon_r] \\ &= [5.5 + 4\epsilon_1 - 2\epsilon_2 + 4.5\epsilon_r] + [-1.5 - \epsilon_1 - \epsilon_2 - 0.5\epsilon_r] \\ &= 4 + 3\epsilon_1 - 3\epsilon_2 + 4\epsilon_r \end{aligned}$$

Thus, it turns out that $\gamma_{\mathrm{RAF}_2}(f^\sharp_{\mathrm{RAF}_2}(\tilde{x}_1, \tilde{x}_2)) = [4 - 10, 4 + 10] = [-6, 14]$, which is incomparable with $\gamma_{\mathrm{Int}}(f^\sharp_{\mathrm{Int}}(I_{x_1}, I_{x_2})) = [-4, 16]$. □

In view of Example 5.1, for a nonlinear binary classifier $C(\mathbf{x}) = \mathrm{sign}(D(\mathbf{x}) - b)$, with $D(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})$, we will use both the interval and RAF abstractions of C in order to combine their final abstract results. More precisely, if $D^\sharp_{\mathrm{Int}_n}$ and $D^\sharp_{\mathrm{RAF}_n}$ are, resp., the interval and RAF abstractions of D, assume that $P : X \to \wp(X)$ is a

perturbation for C which is soundly approximated by $P^\sharp_{\text{Int}} : X \to \text{Int}_n$ on intervals and by $P^\sharp_{\text{RAF}} : X \to \text{RAF}_n$ on RAFs, so that $P^\sharp : X \to \text{Int}_n \times \text{RAF}_n$ is defined by $P^\sharp(\mathbf{x}) \triangleq \langle P^\sharp_{\text{Int}}(\mathbf{x}), P^\sharp_{\text{RAF}}(\mathbf{x})\rangle$. Then, for each input vector $\mathbf{x} \in X$, our combined verifier first will run both $D^\sharp_{\text{Int}_n}(P^\sharp_{\text{Int}}(\mathbf{x}))$ and $D^\sharp_{\text{RAF}_n}(P^\sharp_{\text{RAF}}(\mathbf{x}))$. Next, the output $D^\sharp_{\text{RAF}_n}(P^\sharp_{\text{RAF}}(\mathbf{x})) = \hat{a} \in \text{RAF}_n$ is abstracted to the interval $[c_{\hat{a}} - \text{rad}(\hat{a}), c_{\hat{a}} + \text{rad}(\hat{a})]$ which is then intersected with the interval $D^\sharp_{\text{Int}_n}(P^\sharp_{\text{Int}}(\mathbf{x})) = [l, u]$. Summing up, our combined abstract binary classifier $C^\sharp : \text{Int}_n \times \text{RAF}_n \to \{-1, +1, \top\}$ is defined as follows:

$$C^\sharp(P^\sharp(\mathbf{x})) \triangleq \begin{cases} +1 & \text{if } \max(l, c_{\hat{a}} - \text{rad}(\hat{a})) \geq 0 \\ -1 & \text{if } \min(u, c_{\hat{a}} + \text{rad}(\hat{a})) < 0 \\ \top & \text{otherwise} \end{cases}$$

As shown in Section 4, it turns out that all the linear and nonlinear abstract operations for polynomial and RBF kernels are sound, so that by Lemma 3.4, this combined abstract classifier $C^\sharp$ induces a sound robustness verifer for C. Finally, for multi-classification, in both linear and nonlinear cases, we will use the sound abstract ovo multi-classifier defined in Lemma 3.7.

6 Experimental Results

We implemented our robustness verification method for SVM classifiers in a tool called *SAVer* (*S*vm *A*bstract *Ver*ifier), which has been written in C (approximately 2.5k LOC) and whose source code together with all the datasets, trained SVMs and results is available on GitHub [30]. We assessed the percentage of samples of the full test sets for which a SVM classifier is proved to be robust (and, dually, vulnerable) for a given perturbation, as well as the average verification time per sample. We also evaluated the impact of using subsets of the training set on the robustness of the corresponding classifiers and on verification times. We compared SAVer to DeepPoly [32], a robustness verification tool for convolutional deep neural networks based on abstract interpretation. Our experimental results indicate that SAVer is fast and scalable and that the percentage of robustness provable by SAVer for SVMs on MNIST is higher than the robustness provable by DeepPoly for deep neural networks. Our experiments were run on a AMD Ryzen 7 1700X 3.0GHz CPU.

Datasets and Classifiers. For our experimental evaluation of SAVer we used the standard and widespread MNIST [18] image dataset together with the recent alternative Fashion-MNIST (F-MNIST) image dataset [42]. They both contain grayscale images of $28 \times 28 = 784$ pixels (of depth 256) which are represented as normalized vectors of floating-point numbers in $[0, 1]^{784}$ (0 is black, 1 is white). MNIST contains images of handwritten digits, while F-MNIST comprises professional images of fashion dress products from 10 categories taken from the popular Zalando's e-commerce website. F-MNIST has been recently put forward as a more challenging alternative for the original MNIST dataset for benchmarking machine learning algorithms, since the extensive experimental results reported in [42] showed that the test accuracy of most machine learning classifiers significantly decreases (a rough average is about 10%) from

MNIST to F-MNIST. In particular, [42] reports that the average test accuracy (on the whole test set) of linear, polynomial and RBF SVMs on MNIST is 95.4% while for F-MNIST drops to 87.4%, where RBF SVMs are reportedly the most precise classifiers on F-MNIST with an accuracy of 89.7%. Both datasets include a training set of 60000 images and a test set of 10000 images, with no overlap. Our tests are run on the whole test set, where, following [32], these 10000 images of MNIST and F-MNIST have been filtered out of those misclassified by the SVMs (ranging from 3% of RBF and polynomial kernels to 7% for linear kernel), while the experiments comparing SAVer with DeepPoly are conducted on the same small test subset of MNIST used in [32]. We trained a number of SVM classifiers using different subsets of the training sets and different kernel functions. We trained our SVMs with linear, RBF and (2, 3 and 9 degrees) polynomial kernels, and in order to benchmark the scalability of the verifiers we used the first 1k, 2k, 4k, 8k, 16k, 30k, 60k samples of the training set (training times never exceeded 3 hours). For training we used Scikit-learn [25], a popular machine learning library for Python, which relies on the standard Libsvm C library [5].

Results. The results of our experimental evaluation are summarized by the following tables and charts.

P^{∞}_{δ}	Provable Robustness %				
	Linear	**Poly2**	**Poly3**	**Poly9**	**RBF**
0.01	82.23	98.64	99.07	98.51	99.83
0.02	38.95	94.82	96.96	96.34	99.57
0.03	12.77	82.14	91.80	92.85	99.19
0.04	3.22	57.44	78.95	87.33	97.27
0.05	0.71	30.52	57.31	77.69	93.58
0.06	0.13	14.89	34.80	61.12	82.21
0.07	0.00	7.89	18.36	39.75	67.76
0.08	0.00	4.08	10.64	23.70	48.02
0.09	0	1.61	6.28	12.86	28.10
0.10	0	0.58	3.33	7.18	16.38

(a): Comparison of kernel functions

P^{∞}_{δ}	Provable Robustness %		
	Interval	**RAF**	**Combined**
0.01	47.73	99.83	99.83
0.02	14.95	99.57	99.57
0.03	6.26	99.19	99.19
0.04	2.42	97.27	97.27
0.05	0.82	93.58	93.58
0.06	0.17	82.21	82.21
0.07	0.04	67.76	67.76
0.08	0	48.02	48.02
0.09	0	28.10	28.10
0.10	0	16.38	16.38

(b): Comparison of abstractions for RBF kernel

P^{∞}_{δ}	Provable Robustness %						
	1k	**2k**	**4k**	**8k**	**16k**	**30k**	**60k**
0.01	99.50	99.72	99.69	99.74	99.73	99.77	99.83
0.02	99.01	99.20	99.36	99.35	99.49	99.42	99.57
0.03	98.28	98.58	98.81	98.80	98.95	98.94	99.19
0.04	97.01	97.73	97.91	97.99	98.02	97.71	97.27
0.05	95.28	96.42	96.71	96.50	96.32	95.73	93.58
0.06	93.15	94.69	95.13	94.33	93.90	91.72	82.21
0.07	90.30	92.15	91.16	91.07	88.51	84.43	67.76
0.08	86.69	87.92	87.89	84.86	78.40	66.84	48.02
0.09	81.79	82.47	81.01	74.29	62.38	49.45	28.10
0.10	75.45	74.66	70.83	58.80	45.87	30.86	16.38

(c): Comparison of training set sizes (thousands of samples)

P^{∞}_{δ}	Provable Robustness %		Provable Vulnerability %	
	MNIST	**F-MNIST**	**MNIST**	**F-MNIST**
0.01	99.83	88.59	94.48	39.20
0.02	99.57	60.63	73.62	11.80
0.03	99.19	42.13	48.47	5.50
0.04	97.27	27.24	32.51	3.00
0.05	93.58	18.36	20.25	1.50
0.06	82.21	12.18	9.86	0.90
0.07	67.76	8.22	3.68	0.60
0.08	48.02	5.23	0.61	0.40
0.09	28.10	1.96	0	0.10
0.10	16.38	0.48	0	0

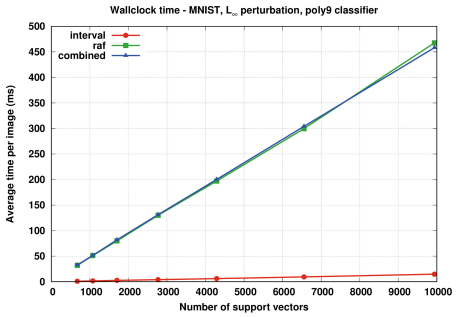
(d): MNIST vs F-MNIST

P^{frm}_{t}	Provable Robustness	
	MNIST	**F-MNIST**
1	100.00%	49.56%
2	99.64%	4.71%
3	87.34%	0.00%
4	40.35%	0.00%

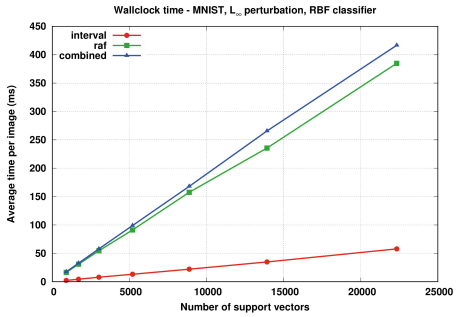
(e): MNIST vs F-MNIST

P^{∞}_{δ}	SAVer		DeepPoly	
	poly9	**RBF**	**Sigmoid**	**Small**
0.005	100%	100%	100%	100%
0.010	98.9%	100%	98%	95%
0.015	98.9%	100%	97%	75%
0.020	97.8%	100%	95%	50%
0.025	97.8%	100%	92%	25%
0.030	96.7%	100%	80%	10%

(f): SAVer vs DeepPoly



(g)



(h)

Table (a) compares the provable robustness to a P_δ^∞ adversarial region of SVMs which have been trained with different kernels. It turns out that the RBF classifier is the most provably robust: even with $\delta = 0.03$, meaning a perturbation of pixel depth of ± 7, SAVer can prove that more than 99% of the full test set of MNIST is robust. The RBF classifier is therefore taken as reference classifier for the successive experiments. Table (b) compares the relative precisions of robustness verification which can be obtained by changing the abstraction of the RBF classifier. As expected, the relational information of the RAF abstraction makes it significantly more precise than interval abstraction, although in a few cases (which do not affect the reported percentages) intervals can help in refining RAF analysis, and this justifies their combined use. Table (c) shows how the provable robustness depends on the size of the training subset. We may observe here that using more samples for training a SVM classifier tends to overfit the model, making it more sensitive to perturbations, i.e. less robust. Table (d) shows what we call *provable vulnerability* of a classifier C: we first consider all the samples $(\mathbf{x}, y)$ in the test set which are misclassified by C, i.e., $C(\mathbf{x}) = y' \neq y$ holds, then our robustness verifier is run on the perturbations $P_\delta^\infty(\mathbf{x})$ of these samples for checking whether the region $P_\delta^\infty(\mathbf{x})$ can be proved to be consistently misclassified by C to y'. Provable vulnerability is significantly lower than provable robustness, meaning that when the classifier is wrong on an input vector, it is more likely to assign different labels to similar inputs, rather than assigning the same (wrong) class. Charts (g) and (h) show the average verification time per image, in milliseconds, with respect to the size of the classifier, given by the number of support vectors, and compared for different abstractions. Let N and n denote, resp., the number of support vectors and the size of input vectors. The interval-based abstract d-polynomial classifier is in $O(dN)$ time, while the RBF classifier is in $O(N)$, because the interval multiplication is constant-time. Hence, interval analysis is very fast, just a few milliseconds per image. On the other hand, the RAF-based abstract d-polynomial and RBF classifiers are, resp., in $O(dNn \log n)$ and $O(Nn \log n)$, since RAF multiplication is in $O(n \log n)$, so that RAF-based verification is slower although it never takes more than 0.5 seconds.

The same experiments have been replicated on the F-MNIST dataset and Table (d) shows a comparison of the results between MNIST and F-MNIST. As expected, robustness is harder to prove (and very likely to achieve) on F-MNIST than on MNIST, while SAVer proved that F-MNIST is less vulnerable than MNIST. Moreover, Table (e) shows the percentage of provable robustness for MNIST and F-MNIST for the frame adversar-

ial region defined in Section 5, for some widths of the frame. F-MNIST is significantly harder to prove robust under this attack than MNIST: this is due to the fact that the borders of MNIST images do not contain as much information as their centers so that classifiers can tolerate some perturbation in the border. By contrast, F-MNIST images often carry information on their borders, making them less robust to adversarial framing. Finally, Table (f) compares SAVer with DeepPoly, a robustness verifier for feedforward neural networks [32]. This comparison used the same test set of DeepPoly, consisting of the first 100 images of the MNIST test set, and the same perturbations P_δ^∞. Although a strict comparison is not possible, as SAVer and DeepPoly operates on different ML models, we argue that percentages of provable robustness achieved by SAVer are competitive with respect to other state-of-the-art tools. Moreover, we point out the fact that a verification of a single image by DeepPoly can take as long as 10s [32], while the maximum verification time per image on SAVer is 0.5s. Among the benchmarks reported in [32, Section 6], we selected the FFNNSmall and FFNNSigmoid deep neural networks, denoted, resp., by DeepPoly Small and Sigmoid. FFNNSmall has been trained using a standard technique and achieved the best accuracies in [32], while FFNNSigmoid was trained using PGD-based adversarial training, a technique explicitly developed to make a classifier more robust. It turns out that the percentages of robustness provable by SAVer are higher than those provable by DeepPoly (precise percentages are not provided in [32], we extrapolated them from the charts). In particular, both 9-polynomial and RBF SVMs can be proved more robust that FFNNSigmoid networks, despite the fact that these classifiers are defended by a specific adversarial training.

7 Future Work

We believe that this work represents a first step in applying formal analysis and verification techniques to machine learning based on support vector machines. We envisage a number of challenging research topics as subject for future work. Generating adversarial examples to machine learning methods is important for designing more robust classifiers [11,41,45] and we think that the completeness of robustness verification of linear binary classifiers (cf. Section 3) could be exploited for automatically detecting adversarial examples in linear multiclass SVM classifiers. The main challenge here is to design more precise, ideally complete, techniques for abstracting multi-classification based on binary classification. Adversarial SVM training is a further stimulating research challenge. Mirman et al. [23] put forward an abstraction-based technique for adversarial training of robust neural networks. A similar approach could also work for SVMs, namely applying abstract interpretation to SVM training models rather than to SVM classifiers.

Acknowledgements. We are grateful to the anonymous referees for their helpful remarks. The doctoral fellowship of Marco Zanella is funded by Fondazione Bruno Kessler (FBK), Trento, Italy. This work has been partially funded by the University of Padova, under the SID2018 project "Analysis of STatic Analyses (ASTA)" and by the Italian Ministry of Research MIUR, under the PRIN2017 project no. 201784YSZ5 "AnalysiS of PRogram Analyses (ASPRA)".

References

1. G. Anderson, S. Pailoor, I. Dillig, and S. Chaudhuri. Optimization and abstraction: A synergistic approach for analyzing neural network robustness. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI2019)*, pages 731–744. ACM, 2019.
2. B. Biggio, I. Corona, B. Nelson, B. I. P. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli. Security evaluation of support vector machines in adversarial environments. In Y. Ma and G. Guo, editors, *Support Vector Machines Applications*, pages 105–153. Springer, 2014.
3. B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In *Proceedings of the 3rd Asian Conference on Machine Learning (ACML2011)*, pages 97–112, 2011.
4. N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. In *Proc. of 2017 IEEE Symposium on Security and Privacy (SP2017)*, pages 39–57, 2017.
5. C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.
6. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL1977)*, pages 238–252. ACM, 1977.
7. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
8. R. Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *Proc. 15th Intern. Symp, on Automated Technology for Verification and Analysis (ATVA2017)*, pages 269–286, 2017.
9. T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *Proc. 2018 IEEE Symposium on Security and Privacy (SP2018)*, pages 3–18, 2018.
10. I. Goodfellow, P. McDaniel, and N. Papernot. Making machine learning robust against adversarial inputs. *Commun. ACM*, 61(7):56–66, 2018.
11. I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. International Conference on Learning Representations (ICLR2015)*, 2015.
12. D. Gopinath, G. Katz, C. S. Pasareanu, and C. Barrett. DeepSafe: A data-driven approach for assessing robustness of neural networks. In *Proceedings of the 16th Int. Symp. on Automated Technology for Verification and Analysis (ATVA2018)*, pages 3–19, 2018.
13. E. Goubault and S. Putot. A zonotopic framework for functional abstractions. *Formal Methods in System Design*, 47(3):302–360, 2015.
14. C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neur. Netw.*, 13(2):415–425, 2002.
15. X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In *Proc. Intern. Conf. on Computer Aided Verification (CAV2017)*, pages 3–29. Springer, 2017.
16. G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proc. Intern. Conf. on Computer Aided Verification (CAV2017)*, pages 97–117. Springer, 2017.
17. A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *Proceedings of the 5th International Conference on Learning Representations (ICLR2017)*, 2017.
18. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

19. F. Leofante and A. Tacchella. Learning in physical domains: Mating safety requirements and costly sampling. In *Proc. of the Conference of the Italian Association for Artificial Intelligence*, pages 539–552. Springer, 2016.
20. F. Messine. Extentions of affine arithmetic: Application to unconstrained global optimization. *J. Universal Computer Science*, 8(11):992–1015, 2002.
21. A. Miné. Relational abstract domains for the detection of floating-point run-time errors. In *Proc. European Symposium on Programming (ESOP2004)*, pages 3–17. Springer, 2004.
22. A. Miné. Tutorial on static inference of numeric invariants by abstract interpretation. *Foundations and Trends in Programming Languages*, 4(3-4):120–372, 2017.
23. M. Mirman, T. Gehr, and M. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *Proc. of the International Conference on Machine Learning (ICML2018)*, pages 3575–3583, 2018.
24. G. P. Nam, B. J. Kang, and K. R. Park. Robustness of face recognition to variations of illumination on mobile devices based on SVM. *KSII Transactions on Internet and Information Systems*, 4(1):25–44, 2010.
25. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
26. L. Pulina and A. Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In *Proc. of the Intern. Conf. on Computer Aided Verification (CAV2010)*, pages 243–257. Springer, 2010.
27. L. Pulina and A. Tacchella. Challenging SMT solvers to verify neural networks. *AI Commun.*, 25(2):117–135, 2012.
28. F. Ranzato. Complete abstractions everywhere (invited paper). In *Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)*, LNCS vol. 7737, pages 15–26. Springer, 2013.
29. F. Ranzato and M. Zanella. Robustness verification of support vector machines. http://arxiv.org/abs/1904.11803, CoRR arXiv, April 2019.
30. F. Ranzato and M. Zanella. SAVer GitHub Repository. https://github.com/svm-abstract-verifier, 2019.
31. G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. T. Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems 31: Proc. Annual Conference on Neural Information Processing Systems 2018, (NeurIPS2018)*, pages 10825–10836, 2018.
32. G. Singh, T. Gehr, M. Püschel, and M. Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL2019):41:1–41:30, Jan. 2019.
33. I. Skalna and M. Hladík. A new algorithm for Chebyshev minimum-error multiplication of reduced affine forms. *Numerical Algorithms*, 76(4):1131–1152, Dec 2017.
34. I. C. Society. *IEEE standard for binary floating-point arithmetic*. Institute of Electrical and Electronics Engineers, New York, 1985. Note: Standard 754–1985.
35. J. Stolfi and L. H. de Figueiredo. *Self-Validated Numerical Methods and Applications*. Brazilian Mathematics Colloquium Monograph, IMPA, Rio de Janeiro, Brazil, 1997.
36. J. Stolfi and L. H. de Figueiredo. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37(1):147–158, Dec 2004.
37. T. B. Trafalis and R. C. Gilbert. Robust support vector machines for classification and computational issues. *Optimisation Methods and Software*, 22(1):187–198, 2007.
38. Y. Vorobeychik and M. Kantarcioglu. Adversarial machine learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*, volume 12(3), pages 1–169. Morgan & Claypool Publishers, August 2018.

39. S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Formal security analysis of neural networks using symbolic intervals. In *Proceedings of the 27th USENIX Conference on Security Symposium*, (SEC2018), pages 1599–1614. USENIX Association, 2018.
40. T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. S. Boning, and I. S. Dhillon. Towards fast computation of certified robustness for ReLU networks. In *Proceedings of the 35th International Conference on Machine Learning, (ICML2018)*, pages 5273–5282, 2018.
41. H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015.
42. H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. CoRR arXiv, abs/1708.07747, 2017.
43. H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10:1485–1510, 2009.
44. M. Zajac, K. Zolna, N. Rostamzadeh, and P. O. Pinheiro. Adversarial framing for image and video classification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI2019)*, 2019.
45. Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. In *Proc. 6th International Conference on Learning Representations (ICLR2018)*, 2018.
46. Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi. Adversarial support vector machine learning. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2012)*, pages 1059–1067. ACM, 2012.