2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Turing meets Machine Learning: Uncomputability of Zero-Error Classifiers

Holger Boche, Yannik N. Böck, Stefanie Speidel, Frank H. P. Fitzek

Abstract— **In almost all areas of information technology, the importance of automated decision-making based on intelligent algorithms has been increasing steadily within recent years. Since many of the envisioned near-future applications of these algorithms involve critical infrastructure or sensitive human goods, a sound theoretical basis for integrity assessment is required, if for no other reason than the legal accountability of system operators. This article aims to contribute to the understanding of integrity of automated decision-making under the aspect of fundamental mathematical models for computing hardware. To this end, we apply the theory of Turing machines to the problem of separating the support sets of smooth functions, which provides a simple yet mathematically rigorous framework for support-vector machines on digital computers. Further, we investigate characteristic quantities and objects, such as the distance between two separated support sets, or separating hyperplanes themselves, with regards to their computability properties, and provide non-technical interpretations of our findings in the context of machine learning and technological trustworthiness.**

I. Introduction

Throughout recent years, the importance of automated decision-making within any domain of society that involves modern information technology has been increasing steadily. Developing the underlying hard- and software components in view of growingly complex systems and applications is often key feature of contemporary engineering professions. Additionally, the development process often has to take a range of other constraints into account, including requirements that concern the perception and acceptance of technological systems by society. For good reasons, these requirements are particularly high in the domain of critical infrastructure and, more general, whenever sensitive human goods and rights are potentially affected.

Among others, the societal requirements towards intelligent near-future systems can be observed in the communications sector. The recent COVID-19 pandemic made it particularly clear that mobile communications is a mainstay of modern society, and its significance can be expected to increase further once the envisioned next-generation technologies have been put to practice. In this context, the upcoming 6G mobile-communications standard supposed to be finalized by 2030 is currently the main reference for those infrastructural concepts that are planned to be realized throughout the next years. In particular, it aims at formalizing the implementation of remote and distributed control functions orchestrated through wireless networking, as well as joint communications and sensing. As the complexity of the resulting networks is expected to be considerably high, 6G involves contemporary machine-learning techniques for optimization, decision-making, and control, to a large extent [1]. In fact, paradigm shifts towards employing intelligent systems in wireless networking have already been advocated in the context of the 5G communications standard [2]. On the customer-level, autonomous driving or the personal tactile internet are well-known examples of technologies hoped to be established by 6G infrastructure. Considering the accompanied impact on e.g. physical inviolability, privacy, and civil rights, it has already been recognized that *technological trustworthiness* – an umbrella-term for technological *privacy, security, integrity, resilience, reliability, availability, accountability, authenticity* and *device independence* – has to be understood in a new context in order for the next-generation of communication technologies to gain widespread societal acceptance [3], [4].

Technological trustworthiness encompasses the concepts of integrity and accountability, which the mathematical problems investigated in this article relate to. On the level of system functionality, *integrity* in short refers to the guarantee that the system in question remains within the prescribed margin of operation, whereas *accountability* refers to the operator's (legal) obligation to bear responsibility for the possible consequences of operating the system. Accordingly, integrity and accountability are interdependent aspects of trustworthiness. Once the 6G mobile-communications standard has been established, its physical infrastructural components,

The work of *Holger Boche* was supported in part by the German Federal Ministry of Education and Research (BMBF) within the National Initiative on 6G Communication Systems through the Research Hub 6G-life under Grant 16KISK002.

The work of *Yannik Böck* was supported in part by the BMBF within NewCom under Grant 16KIS1004, in part by the BMBF within 6G-life under Grant 16KISK002.

The work of *Stefanie Speidel* and *Frank Fitzek* was funded in part by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy—EXC 2050/1—Project ID 390696704—Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden.

Furthermore, *Stefanie Speidel* and *Frank Fitzek* acknowledge the financial support by the BMBF in the programme of "Souverän. Digital. Vernetzt.". Joint project 6G-life, project identification number: 16KISK001K.

Holger Boche and *Yannik N. Böck* are with the Lehrstuhl für Theoretische Informationstechnik and the BMBF Research HUB 6G-life (TU Munich), Technische Universität München, Munich, Germany {boche, yannik.boeck}@tum.de

Holger Boche is further with the Munich Center for Quantum Science and Technology (MCQST), Munich, Germany.

Stefanie Speidel is with the National Center for Tumor Diseases (NCT/UCC) Dresden, the Cluster of Excellence CeTI, and the BMBF Research HUB 6G-life (TU Dresden), Technische Universität Dresden, Dresden, Germany stefanie.speidel@nct-dresden.de

F. H. P. Fitzek is with the Deutsche Telekom Chair of Communication Network, the Cluster of Excellence CeTI, and the BMBF Research HUB 6G-life (TU Dresden), Technical University of Dresden, Dresden, Germany, frank.fitzek@tu-dresden.de

e.g., base stations, data processing nodes, and mobile agents, will start to be implemented. To this end, appropriate hardware platforms and network architectures have to be selected, which will later realize the functionalities provided by the 6G network. These functionalities are specified by protocols, which have to be implemented in terms of algorithms. In turn, the range of feasible algorithms depends on the chosen hardware platform, which leads to the problem of whether integrity with respect to a certain functionality can be provided at all. To a large extent, our evaluation of a system's integrity is based on mathematical models that describe the physical laws involved in the considered engineering problem. The hardware components employed to implement the system are themselves subject to such models. In fact, the models that characterize the employed hardware platform determine the class of feasible algorithms. Thus, in order to establish a mathematically sound technology assessment, choosing a hardware platform that is theoretically capable of capturing the physical nature of the functionality to be implemented is a basic and necessary requirement. Furthermore, it is crucial for legal accountability, since it must be possible to decide in a transparent and unambiguous manner whether a certain software implementation is considered defective or not.

This work aims at contributing to the understanding of the fundamental limitations of digital technology in automated decision-making. To this end, we apply the theory of Turing machines to the problem of separating the support sets of smooth functions. Since Turing machines are widely accepted as a complete and definite mathematical model of digital computers, our theory provides a simple yet mathematically rigorous framework for support-vector machines on real-world digital hardware, and allows for the characterization of their fundamental theoretical limitations. The remainder of the article is structured as follows. In Section II, we provide a non-technical problem introduction as well as a non-technical preview of our results, which allows the reader to interpret our theory in the context of technological integrity. Sections III and IV are dedicated to mathematical preliminaries. Then, in Sections V, VI, VII, and VIII, we present our findings in a formal mathematical manner. We then conclude the article by a brief subsumption and some additional remarks in Section IX.

For the sake of brevity, we refrain from presenting detailed proofs. Theorem 1 is obtained from an essential generalization of the methods applied to derive [5, Theorem 6, p. 697, Theorem 7, p. 698]. Theorems 2, 4, and 5 are obtained through contradiction: By exploiting the relevant functions' discontinuity concerning convex combinations of elements of their domains, it would be possible to construct a decision algorithm for the *halting problem*. Similar methods were used to derive [6, Theorem 3, p. 2457, Theorem 6, p. 2459] and [5, Theorem 3, p. 693f, Theorem 4, p. 694], to which we refer for details.

II. Problem Formulation and Outline

Intelligent automated decision-making is essentially related to data classification, a well-known problem in the community of machine learning. For comprehensive overviews, we refer to [7], [8]. The data space $\mathcal{D}$ contains a collection of regions $\mathcal{D}_0, \mathcal{D}_1, \ldots \mathcal{D}_N$, $N \in \mathbb{N}_+$, such that $\mathcal{D}_0 \cup \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_N \subseteq \mathcal{D}$ is satisfied. Each region is associated to a label, which, for simplicity, we assume to equal the associated region's index. That is, we have $L(\boldsymbol{x}) = 0$ iff $\boldsymbol{x} \in \mathcal{D}_0$, $L(\boldsymbol{x}) = 1$ iff $\boldsymbol{x} \in \mathcal{D}_1$, and so on, with $\mathcal{L} := \{0, 1, \ldots, N\} \subset \mathbb{N}$ being the set of labels. Throughout this article, we consider the simple case of $\mathcal{L} = \{0, 1\}$. The general problem then consists of finding a suitable classification algorithm that, when presented with a description of an object $\boldsymbol{x} \in \mathcal{D}_0 \cup \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_N$, tries to infer the associated label $L(\boldsymbol{x})$. Depending on the context, this classifier can be interpreted as a decision rule, i.e., the inferred label corresponds to the automated decision made on the basis of the available data.

As a key feature of machine learning, the classifier algorithm is itself computed by a digital machine, as opposed to being implemented by a human programmer. In more formal terms, the considered machine-learning algorithm gives rise to a class of mappings

$$\mathfrak{L} \subseteq \left\{ \widehat{L} : \mathcal{D} \supseteq\!\rightarrow \mathcal{L}, \boldsymbol{x} \mapsto \widehat{L}(\boldsymbol{x}) \right\},$$

where the symbol '$\supseteq\!\rightarrow$' indicates a *partial mapping*, and tries to select a data classifier $\widehat{L} \in \mathfrak{L}$ that, according to some fixed quality criterion, best matches the true labeling $L : \mathcal{D}_0 \cup \mathcal{D}_1 \rightarrow \mathcal{L}$ among the available choices. The selection is based on training data $\boldsymbol{y}$ and (possibly) auxiliary information about the regions $\mathcal{D}_0$ and $\mathcal{D}_1$. Commonly, the training data is of the form

$$\boldsymbol{y} := \big((\boldsymbol{x}_1, l_1), \ldots, (\boldsymbol{x}_N, l_N)\big) \in (\mathcal{D} \times \mathcal{L})^N, \ N \in \mathbb{N}_+,$$

such that $L(\boldsymbol{x}_n) = l_n$ for $n = 1, \ldots, N$, i.e., $\boldsymbol{y}$ consists of a finite list of (correctly) labeled data points. The process is visualized in Figure 1 (top).

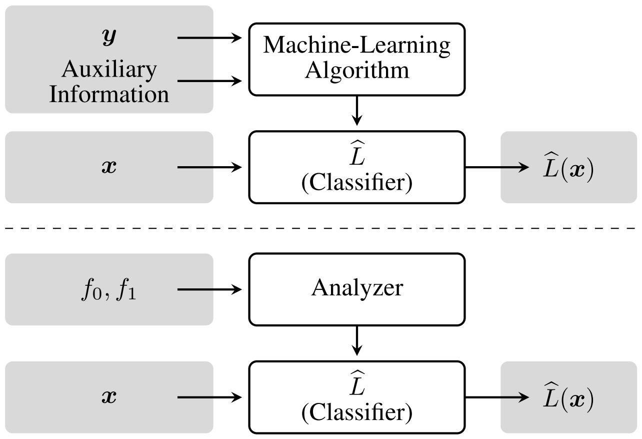
$\boldsymbol{y}$
Auxiliary Information
Machine-Learning Algorithm
$\boldsymbol{x}$
$\widehat{L}$ (Classifier)
$\widehat{L}(\boldsymbol{x})$
f_0, f_1
Analyzer
$\boldsymbol{x}$
$\widehat{L}$ (Classifier)
$\widehat{L}(\boldsymbol{x})$

Fig. 1. *Machine learning for data classification.* **Top:** Based on a set of training data and some auxiliary information, the machine-learning algorithm computes a classifier. **Bottom:** An analyzer computes a classifier by extracting the relevant information from an algorithmic characterization of suitable probability density functions.

Albeit a large amount of research has been invested into machine learning, it suffers from a list of drawbacks that are yet to be overcome. These include, but are not limited to, the lack of *stability*, *performance guarantees*, and *stopping criteria*. That is, even tho a classifier $\widehat{L}$ may perform well on regular data, a small amount of "unnatural" deviations in the input $\boldsymbol{x}$ can make the output $\widehat{L}(\boldsymbol{x})$ become almost completely arbitrary. Even when restricted to noiseless and regular data, there is generally no way to know for certain how well the classifier $\widehat{L}$ will generalize to new data points not contained in the training set $\boldsymbol{y}$. Furthermore, many machine learning algorithms are based on iterative computations, but there is no general method to determine a suitable amount of iterations. While a small number of iterations may result in the choice of a classifier that performs poorly even on the training data, a large number of iterations may lead lead to phenomena overfitting. Often, the listed issues are interdependent, and cannot be addressed one at a time.

Without any auxiliary information present, it is not possible to solve any of the listed issues, since the classification regions $\mathcal{D}_0$ and $\mathcal{D}_1$ may behave arbitrarily pathological. In most practical applications, such information is indeed available, even though it is commonly not referred to by other names. In the simplest case, auxiliary information may be present in terms of an additional list of labeled data points that is not used for training. The performance of the computed classifier is then evaluated on those data points. If it yields a good rate of accurate classifications, this is usually taken as evidence for performance on new data points. Other forms of auxiliary information include the simple assumption that the classification regions $\mathcal{D}_0$ and $\mathcal{D}_1$ are "well-behaved" in one way or another, or general mathematical regularization techniques. However, these forms of auxiliary information are informal and thus only yield heuristics on the actual performance of the computed classifier.

If provable solutions to any of the previously listed issues are required, the auxiliary information must be available in a mathematically formalized form. For example, the classification regions $\mathcal{D}_0$ and $\mathcal{D}_1$ may be associated to probability density functions $f_0, f_1 : \mathcal{D} \to [0,1]$ with $\operatorname{supp} f_0 = \mathcal{D}_1, \operatorname{supp} f_1 = \mathcal{D}_1$ that characterize the probability of data points. In an optimistic scenario, these functions may be accessible to the machine learning algorithm. In fact, similar scenarios have recently gained attention under the name of *synthetically generated data*. That is, rather than trying to gather the training data $\boldsymbol{y}$ in the real-world, one tries to characterize the functions f_0 and f_1 by an algorithm themselves, which, in principle, can then produce an infinite sequence of labeled samples $(\boldsymbol{x}_1, l_1), (\boldsymbol{x}_2, l_2), (\boldsymbol{x}_3, l_3), \ldots$ that is dense in $\mathcal{D}_0 \cup \mathcal{D}_1$. In the simplest case, this sequence is successively fed into a usual machine learning algorithm, similar to the case of real-world training data. However, this is not the only conceivable method. A more elaborated machine learning approach may instead extract the relevant information from the algorithmic description of f_0 and f_1 directly, by analyzing their source code. Note that this approach is a *generalization* of the aforementioned technique, since an algorithm that has access to f_0 and f_1 may "analyze" their source code by simply drawing a sequence $(\boldsymbol{x}_1, l_1), (\boldsymbol{x}_2, l_2), (\boldsymbol{x}_3, l_3), \ldots$ of sampling points by itself, and then calculate $\widehat{L}$ in the usual manner. The generalized approach is visualized in Figure 1 (bottom). It also avoids another more practical limitation of machine learning: Real-world training data is often costly if not impossible to obtain in large amounts, and the structurally relevant "outlier" data points may only occur in a fraction of cases. Among others, this is the case for applications in healthcare and autonomous driving, c.f. [9], [10] for contemporary examples.

Despite numerous advances in machine learning, such as the previously discussed methods of learning by synthetically generated data, the various drawbacks mentioned above remain. In light of these difficulties, it has been speculated whether more fundamental mathematical reasons may hinder the implementation of stable and reliable machine learning techniques. In theoretical computer science, it is well-established that the algorithmic capabilities of digital hardware is fundamentally limited, and, in many cases, insufficient to capture the structure of theories formalized by means of pure analytic mathematics, such as electromagnetism, classical mechanics, or fluid dynamics. Yet, the design of technology is ultimately based on such theories. Accordingly, it has been conjectured that the drawbacks regarding stability and reliability are related to the fundamental nature of digital algorithms itself. In view of technological integrity, this has critical consequences, since it implies the unsuitability of digital hardware for certain engineering problems as a matter of principle.

While most of the results established in the scope of this work are applicable to machine learning for data classification in general, provided a digital hardware platform is considered, we will put some emphasis on the study of a type of data classifiers known as support-vector machines. Their mathematics belong to the most well-established theories in supervised machine learning, c.f. [11] for details. The data space is a cuboid in the N-dimensional Euclidean space, i.e, $\mathcal{D} = [0,b]^N \subset \mathbb{R}^N$, $b \in \mathbb{Q}_+$, $N \in \mathbb{N}_+$, and we restrict ourselves to the particularly well-behaved case of strictly separated classification regions $\mathcal{D}_0 = \operatorname{supp} f_0$ and $\mathcal{D}_1 = \operatorname{supp} f_1$, i.e., $\mathcal{D}_0 \cap \mathcal{D}_1 = \emptyset$, and arbitrarily smooth functions $f_0, f_1 \in \mathcal{C}_N^0(b)$ (the mathematical terminology will be introduced in Section III).

Definition 1. *Consider $\mathcal{D} = [0,b]^N \subset \mathbb{R}^N$ with $b \in \mathbb{Q}_+$ and $N \in \mathbb{N}_+$. Classifiers of the form*

$$\widehat{L}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \langle \boldsymbol{x}, \boldsymbol{v} \rangle - w > 0, \\ 0, & \text{if } \langle \boldsymbol{x}, \boldsymbol{v} \rangle - w < 0, \end{cases}$$

are referred to as support-vector machines, *with $\boldsymbol{v} \in [0,b]^N$ and $w \in \mathbb{R}$ being the computed parameters of the classifier. The set*

$$\mathcal{H} := \left\{ \boldsymbol{x} \in \mathbb{R}^N : \langle \boldsymbol{x}, \boldsymbol{v} \rangle - w = 0 \right\}$$

is called a separating hyperplane *for $\mathcal{D}_0$ and $\mathcal{D}_1$ if $\langle \boldsymbol{x}, \boldsymbol{v} \rangle < w$ is satisfied for all $\boldsymbol{x} \in \mathcal{D}_0$ and $\langle \boldsymbol{x}, \boldsymbol{v} \rangle > w$ is satisfied for all $\boldsymbol{x} \in \mathcal{D}_1$.*

Definition 2. *Given strictly separated classification regions $\mathcal{D}_0 \subset [0,b]^N$ and $\mathcal{D}_1 \subset [0,b]^N$, we refer to the number*

$$d_{\inf}(\mathcal{D}_0, \mathcal{D}_1) := \inf_{x_0 \in \mathcal{D}_0} \inf_{x_1 \in \mathcal{D}_1} \|x_0 - x_1\|_2$$

as the optimum distance. *If $\boldsymbol{v}$ and w are the parameters of any separating hyperplane $\mathcal{H}$ for $\mathcal{D}_0$ and $\mathcal{D}_1$, we have*

$$\inf_{x_0 \in \mathcal{D}_0} w - \langle \boldsymbol{x}_0, \boldsymbol{v} \rangle \leq 1/2 \cdot d_{\inf}(\mathcal{D}_0, \mathcal{D}_1), \quad (1)$$

$$\inf_{x_1 \in \mathcal{D}_1} \langle \boldsymbol{x}_1, \boldsymbol{v} \rangle - w \leq 1/2 \cdot d_{\inf}(\mathcal{D}_0, \mathcal{D}_1). \quad (2)$$

The separating hyperplane is called optimal *if both* (1) *and* (2) *are satisfied with equality.*

In order to obtain a meaningful analysis, the mathematics of support-vector machines must be established within a formalism for digital computing, which is provided by the theory of Turing machines. Introduced in [12], [13], Turing machines form an abstraction of today's real-world digital computers. Furthermore, the widely accepted Church-Turing thesis implies that this abstraction is, albeit idealized, indeed a complete and definitive model of real-world digital hardware. Consequently, any algorithm that can be executed by a real-world computer can in theory be simulated by a Turing machine. In contrast to real-world computers, however, Turing machines are not subject to restrictions concerning energy consumption, computation time or memory size. All computation steps on a Turing machine are furthermore assumed to be executed with zero chance of error. Computability in the sense of Turing is the exact characterization of what can be achieved by digital hardware, e.g., central processing units (CPUs), digital signal processors (DSPs), or field programmable gate arrays (FPGAs), if practical limitations are disregarded. Thus, by employing the mathematical framework of Turing machines, we obtain a formal theory of digitally computable support-vector machines. The subsequent sections are, after providing a brief overview on the mathematical preliminaries, dedicated to presenting several core results of this theory.

III. Preliminaries: Smooth Functions

In the present Section, we provide a brief introduction to the theory of smooth functions. We then continue to investigate some underlying properties of the associated support sets and the quantity $d_{\inf}$. In particular, this concerns monotonicity and limit properties.

We denote set of continuous functions $f : [0,b]^N \to \mathbb{R}$ that vanish at the boundary of $[0,b]^N$ by $\mathcal{C}_N^0(b)$. That is, we have $f(x) = 0$ for all $x \in [0,b]^N \setminus (0,b)^N$ and all $f \in \mathcal{C}_N^0(b)$. For $K \in \mathbb{N}$, the set $\mathcal{C}_N^K(b)$ is defined recursively. A function $f \in \mathcal{C}_N^0(b)$ is an element of $\mathcal{C}_N^{K+1}(b)$ if for all $i \in \{1, \dots, N\}$, we have

$$\frac{\partial f}{\partial x_i} \in \mathcal{C}_N^K(b).$$

Accordingly, we have $\mathcal{C}_N^{K+1}(b) \subset \mathcal{C}_N^K(b)$ for all $K \in \mathbb{N}$. Analogously, the set $\mathcal{C}_N^\infty(b)$ is defined by

$$\mathcal{C}_N^\infty(b) := \bigcap_{K \in \mathbb{N}} \mathcal{C}_N^K(b).$$

Equivalently, $\mathcal{C}_N^\infty(b)$ is the largest subset of $\mathcal{C}_N^0(b)$ such that

$$\frac{\partial f}{\partial x_i} \in \mathcal{C}_N^\infty(b)$$

is satisfied for all $f \in \mathcal{C}_N^\infty(b)$ and all $i \in \{1, \dots, N\}$. That is, $\mathcal{C}_N^\infty(b)$ is closed with respect to partial differentiation in any of its variables.

Although the spaces $\mathcal{C}_N^K(b)$ are larger than the space $\mathcal{C}_N^\infty(b)$, they exhibit an additional topological structure. Define the mapping $\|\cdot\|_{\mathcal{C},0} : \mathcal{C}_N^0(b) \to \mathbb{R}_0^+$, $f \mapsto \|f\|_{\mathcal{C},0}$ according to

$$\|f\|_{\mathcal{C},0} := \sup_{x \in [0,b]^N} |f(x)| = \max_{x \in [0,b]^N} |f(x)|.$$

Then, the pair $(\mathcal{C}_N^0(b), \|\cdot\|_{\mathcal{C},0})$ becomes a Banach-space. The norm $\|\cdot\|_{\mathcal{C},0}$ can be used to equip the spaces $\mathcal{C}_N^K(b)$, $K \in \mathbb{N}$, with a Banach-space structure as well, by evaluating $\|\cdot\|_{\mathcal{C},0}$ for all derivatives of f up to order K and selecting the maximum of the values obtained in this manner. In formal terms, the norm $\|\cdot\|_{\mathcal{C},K} : \mathcal{C}_N^K(b) \to \mathbb{R}_0^+$ is, as is the case with the set $\mathcal{C}_N^K(b)$ itself, defined recursively by

$$\|f\|_{\mathcal{C},K+1} := \max\left\{ \|f\|_{\mathcal{C},0}, \left\| \frac{\partial f}{\partial x_1} \right\|_{\mathcal{C},K}, \dots, \left\| \frac{\partial f}{\partial x_N} \right\|_{\mathcal{C},K} \right\}.$$

This way, the pairs $(\mathcal{C}_N^K(b), \|\cdot\|_{\mathcal{C},K})$, $K \geq 1$, become Banach-spaces for all $N \in \mathbb{N}$ with $N \geq 1$ as well.

In the following, we denote the closure of a set $\mathcal{D} \subset [0,b]^N$, $N \in \mathbb{N}$, by $\overline{\mathcal{D}}$, as well as its boundary by $\partial\mathcal{D}$, if there is no danger of confusion with the partial derivative. For $f \in \mathcal{C}_N^K(b)$, $K \in \mathbb{N}$ and $\epsilon \geq 0$, we define the sets

$$\mathcal{M}_>(\epsilon, f) : = \Big\{ x \in [0,b]^N : f(x) > \epsilon \Big\},$$

$$\mathcal{M}_\geq(\epsilon, f) : = \Big\{ x \in [0,b]^N : f(x) \geq \epsilon \Big\},$$

as well as $\mathcal{M}_0(f) = [0,b]^N \setminus \mathcal{M}_>(0,f)$. Both $\mathcal{M}_0(f)$ and $\mathcal{M}_\geq(\epsilon, f)$ are closed for all $\epsilon \geq 0$. For notational convenience, we also define

$$\mathcal{M}_+(f) := \overline{\mathcal{M}_>(0,f)} = \mathcal{M}_>(0,f) \cup \partial\mathcal{M}_>(0,f).$$

If f is non-negative, as is the case for probability density functions, we have $\operatorname{supp}(f) = \mathcal{M}_+(f)$. For the sake of generality, we consider $\mathcal{M}_+(f)$ in the following, instead of $\operatorname{supp}(f)$. *However, all results presented in this paper hold analogously if the restriction of $\mathcal{C}_N^K(b)$ to non-negative functions is considered ab initio instead.* Next, for $0 < \epsilon \leq \min\{\|f_0\|_{\mathcal{C},0}, \|f_1\|_{\mathcal{C},0}\}$ and $f_0, f_1 \in \mathcal{C}_N^K(b)$, define

$$\begin{aligned} F(\epsilon, f_0, f_1) : &= \inf_{x_1 \in \mathcal{M}_\geq(\epsilon, f_0)} \inf_{x_2 \in \mathcal{M}_\geq(\epsilon, f_1)} \|x_1 - x_2\|_2 \\ &= \min_{x_1 \in \mathcal{M}_\geq(\epsilon, f_0)} \min_{x_2 \in \mathcal{M}_\geq(\epsilon, f_1)} \|x_1 - x_2\|_2, \quad (3) \end{aligned}$$

where (3) follows from the compactness of $\mathcal{M}_\geq(\epsilon, f_0)$ and $\mathcal{M}_\geq(\epsilon, f_1)$. Observe that $F(\epsilon, f_0, f_1)$ is, in fact, well-defined for all $0 \leq \epsilon \leq \min\{\|f_0\|_{\mathcal{C},0}, \|f_1\|_{\mathcal{C},0}\}$. We have

- $F(\epsilon, f_0, f_1) \geq d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ for all $f_0, f_1 \in \mathcal{C}_N^K(b)$ and all $0 \leq \epsilon \leq \min\{\|f_0\|_{\mathcal{C},0}, \|f_1\|_{\mathcal{C},0}\}$;

- $\lim_{\epsilon \to 0} F(\epsilon, f_0, f_1) = d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ for all for all $f_0, f_1 \in \mathcal{C}_N^K(b)$.

In other words, for $\epsilon \to 0$, the value of $F(\epsilon, f_0, f_1)$ converges towards $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ in a monotonically decreasing manner. However, depending on f_0 and f_1, F may or may not to be continuous in ϵ.

Finally, consider sequences $(f_{0,n})_{n\in\mathbb{N}} \subset \mathcal{C}_N^K(b)$ and $(f_{1,n})_{n\in\mathbb{N}} \subset \mathcal{C}_N^K(b)$ such that

$$\lim_{n\to\infty} \|f_0 - f_{0,n}\|_{\mathcal{C},0} = 0 \text{ and } \lim_{n\to\infty} \|f_1 - f_{1,n}\|_{\mathcal{C},0} = 0$$

are satisfied, where f_0 and f_1 are suitable functions in $\mathcal{C}_N^K(b)$. Then, we have

$$\limsup_{n\to\infty} d_{\inf}\big(\mathcal{M}_+(f_{0,n}), \mathcal{M}_+(f_{1,n})\big) \leq d_{\inf}\big(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1)\big),$$

which is due to the fact that $\mathcal{M}_+(f_m)$, $m \in \{1, 2\}$ are subsets of a union of specific cuts of the sets $\mathcal{M}_+(f_{m,n})$, $m \in \{0, 1\}$, $n \in \mathbb{N}$. In particular, we have

$$\mathcal{M}_+(f_0) \subseteq \bigcup_{M=0}^{\infty} \bigcap_{n=M}^{\infty} \mathcal{M}_+(f_{0,n}),$$
$$\mathcal{M}_+(f_1) \subseteq \bigcup_{M=0}^{\infty} \bigcap_{n=M}^{\infty} \mathcal{M}_+(f_{1,n}).$$

Observe that depending on the choice of $(f_{0,n})_{n\in\mathbb{N}} \subset \mathcal{C}_N^K(b)$ and $(f_{1,n})_{n\in\mathbb{N}} \subset \mathcal{C}_N^K(b)$, the above inequality and inclusions may or may not be strict.

IV. Preliminaries: Computability Theory

Throughout this section, we provide a brief formal introduction to the theory of Turing machines, recursive functions and computable analysis, which are well-established fields in theoretical computer science. A comprehensive treatment of the topic may be found in e.g. [14]–[16]. Subsequently, we will employ the introduced formalism to establish a theory of computable support-vector machines.

In the following, we refer to a mapping of the form $g : \mathbb{N}^n \supseteq\!\!\to \mathbb{N}$, $n \in \mathbb{N}$, as a *natural number function*. Any abstract object (real and complex numbers, matrices, signals, etc.) that is to be processed on a digital computer has, on the lowest level, to be represented by by a bit-string. In turn, any bit-string is the binary representation of some natural number. An algorithm executed by a digital computer can thus be characterized by a natural number function. In the theoretical domain, the same holds true for Turing machines equivalently. The class of natural number functions that can be computed in this manner coincides with the set of *recursive functions*. Recursive functions, more specifically referred to as *μ-recursive functions*, were, amongst others, considered in [17]. Contrary to Turing machines, they are based on forming the closure of a set of axiomatically computable functions – the *successor function* as well as all *constant* and *identity functions* on tuples of natural numbers [14, Definition 2.1, p. 8] – with respect to a number of axiomatically computable operations – *composition*, *primitive recursion* and *unbounded search* [14, Definition 2.1, p. 8, Definition 2.2, p. 10]. Yet, as indicated above, Turing machines and recursive functions are equivalent in the following sense: The class of natural number functions that can be computed by a Turing machine coincides with the set of recursive functions [18].

Recall that for digital computers and Turing machines alike, an object $x \in \mathcal{D}$ that belongs to some abstract set $\mathcal{D}$ has, on the lowest level, to be represented by a bit-string or natural number, respectively. Throughout this article, details on how to represent abstract objects will be implicit, since they are not of essence for our considerations. For two sets $\mathcal{D}$ and $\mathcal{D}'$ and a partial mapping $H : \mathcal{D} \supseteq\!\!\to \mathcal{D}'$, we say that there exists a Turing machine TM_H that computes H, i.e., we have $\mathrm{TM}_H(x) = H(x)$ for all $x \in \mathcal{D}$ that belong to the domain of H, if there exists a recursive function that returns a natural number that represents $H(x)$, whenever its input is a natural number that represents x. Equivalently, we call H *Turing computable*.

Definition 3. *A sequence $(r_n)_{n\in\mathbb{N}} \subset \mathbb{Q}$ is called* computable *if there exist recursive functions $g, h_1, h_2 : \mathbb{N} \to \mathbb{N}$ such that*

$$r_n = \frac{(-1)^{g(n)} \cdot h_1(n)}{1 + h_2(n)}$$

is satisfied for all $n \in \mathbb{N}$. An m-fold computable sequence of rational numbers is analogously defined by recursive functions $g, h_1, h_2 : \mathbb{N}^m \to \mathbb{N}$.

Definition 4. *A number $x \in \mathbb{R}$ is called* computable *if there exists a computable sequence $(r_n)_{n\in\mathbb{N}}$ of rational numbers and a recursive function $\xi\colon \mathbb{N} \to \mathbb{N}$ such that $|x - r_n| \leq 2^{-M}$ holds true for all $n, M \in \mathbb{N}$ that satisfy $n \geq \xi(M)$.*

The above type of convergence is referred to as *effective*, which indicates the existence of a recursive function ξ that provides a quantitative estimate of the approximation error. The function ξ is called a (recursive) *modulus of convergence* for the sequence $(r_n)_{n\in\mathbb{N}}$..

Observe that the set of computable numbers – denoted by $\mathbb{R}_c$ – is countably infinite. Hence, it follows from a cardinality argument that almost all real numbers are *not* computable. An almost trivially necessary requirement for a real number to be computable is its *definability*, i. e., informally speaking, it must be possible to characterize the number by some proper mathematical statement. In more formal terms, this commonly refers whether the number can be (formally) defined by means of some well-formed expression in Peano arithmetics. All mathematical quantities that are relevant in the context of science and engineering, are definable in this sense. However, not all definable real numbers are computable, as we will see in the following.

The class of definable real numbers can be divided into a countable hierarchy $\Sigma_n, \Pi_n : n \in \mathbb{N}$, where the mth level is obtained by alternately applying sup- and inf-operations on m-fold computable sequences of rational numbers [19]. We have $\Sigma_0 = \Pi_0 = \mathbb{Q}$ and $\mathbb{R}_c = \Sigma_1 \cap \Pi_1$ as well as $\Sigma_n \cup \Pi_n \subset \Sigma_{n+1} \cap \Pi_{n+1}$ for all $n \in \mathbb{N}$. A quantity's hierarchical level can be interpreted in terms of the defining expression's structural complexity. In our work, the set Π_1 will be of special relevance. A real numbers x_* satisfies $x_* \in \Pi_1$ if and only if there exist a computable sequence $(r_n)_{n\in\mathbb{N}}$ of rational numbers such that $x_* = \inf_{n\in\mathbb{N}} r_n$ is satisfied. While this definition refers to computable sequences of rational numbers, it can equivalently be expressed in terms of computable sequences of computable numbers: A real numbers x_* satisfies $x_* \in \Pi_1$ if and only if there exist a computable sequence $(x_n)_{n\in\mathbb{N}}$ of computable numbers such that $x_* = \inf_{n\in\mathbb{N}} x_n$ is satisfied. Analogously, a real numbers x_* satisfies $x_* \in \Sigma_1$ if and only if there exist a computable sequence $(x_n)_{n\in\mathbb{N}}$ of computable numbers such that $x_* = \sup_{n\in\mathbb{N}} x_n$ is satisfied. This equivalence can easily be extended to higher hierarchical levels. For details, we refer to [5, Lemma 1, p. 689, Remark 9, p. 690].

Finally, we introduce a notion of computability for the set $\mathcal{C}_N^K(b)$, $N, K \in \mathbb{N}$, $b \in \mathbb{Q}$. Consider points $x_0, x_1 \in [0,b]^N \cap \mathbb{Q}^N$ such that x_0 and x_1 do not coincide in any of their components. Then, x_0 and x_1 define an N-dimensional cuboid $\mathcal{Q}(x_0, x_1)$ with volume $\mathrm{Vol}(x_0, x_1) > 0$, such that x_0 and x_1 constitute to two diametrically opposed corner points of $\mathcal{Q}(x_0, x_1)$.

Definition 5. *A function $f \in \mathcal{C}_N^K(b)$ is called* computable *if there exist Turing machines*

$$\underline{\mathrm{TM}}_f : \left([0,b]^N \cap \mathbb{Q}^N\right) \times \left([0,b]^N \cap \mathbb{Q}^N\right) \to \mathbb{Q},$$
$$\overline{\mathrm{TM}}_f : \left([0,b]^N \cap \mathbb{Q}^N\right) \times \left([0,b]^N \cap \mathbb{Q}^N\right) \to \mathbb{Q},$$

such that the following holds true for all (not necessarily computable) sequences $(x_{0,n})_{n\in\mathbb{N}} \subset [0,b]^N \cap \mathbb{Q}^N$ and $(x_{1,n})_{n\in\mathbb{N}} \subset [0,b]^N \cap \mathbb{Q}^N$ that satisfy $\mathrm{Vol}(x_{0,n}, x_{1,n}) > 0$ for all $n \in \mathbb{N}$:

- *For all $n \in \mathbb{N}$ and all $x \in \mathcal{Q}(x_{0,n}, x_{1,n})$, the inequality $\underline{\mathrm{TM}}_f(x_{0,n}, x_{1,n}) \leq f(x) \leq \overline{\mathrm{TM}}_f(x_{0,n}, x_{1,n})$ holds true.*
- *If $\mathcal{Q}(x_{0,n}, x_{1,n}) \subseteq \mathcal{Q}(x_{0,n+1}, x_{1,n+1})$ for all $n \in \mathbb{N}$ and $\lim_{n\to\infty} \mathrm{Vol}(x_{0,n}, x_{1,n}) = 0$ are satisfied, we have $\lim_{n\to\infty} \underline{\mathrm{TM}}_f(x_{0,n}, x_{1,n}) = \lim_{n\to\infty} \overline{\mathrm{TM}}_f(x_{0,n}, x_{1,n})$.*

We denote the set of computable functions in $\mathcal{C}_N^K(b)$ by $\mathcal{C}_{\mathrm{c},N}^K(b)$. The present notion of computability is often referred to as *Borel-Turing computability*.

V. Optimum Distance is Π_1-Complete

In this section, we present our first main result. Given two arbitrarily smooth functions f_0 and f_1 with strictly separated support, the optimum distance $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ is generally an *uncomputable number*. That is, in contrast to numbers such as π, e, or $\sqrt{2}$, it may be fundamentally impossible to compute $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ up to arbitrary precision. More precisely, for $N \geq 1$, $K \in \mathbb{N}$, the range of values of attained by

$$(f_0, f_1) \mapsto d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$$

for $f_0, f_1 \in \mathcal{C}_{\mathrm{c},N}^K(b)$ with $\mathcal{M}_+(f_0) \cap \mathcal{M}_+(f_1) = \emptyset$ can be shown to coincide with an interval in Π_1. The fact that $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ is generally an uncomputable number then follows as an immediate consequence.

Theorem 1. *For all $N, K \in \mathbb{N}$ and all $b \in \mathbb{Q}$ that satisfy $b > 0$, we have*

$$\left\{x \in \Pi_1 : 0 \leq x < b\sqrt{N}\right\}$$
$$= \left\{d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1)) : f_0, f_1 \in \mathcal{C}_{\mathrm{c},N}^K(b)\right\}.$$

In Section II, we have hinted towards the issue of finding suitable stopping criteria and performance guarantees in intelligent decision making. For arbitrarily smooth functions f_0 and f_1, we can always find computable sequences of rational numbers that converge monotonically towards $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ from above. However, in order to evaluate the maximally achievable performance and stability of a classifier algorithm for the classification regions $\mathcal{D}_0 = \mathcal{M}_+(f_0)$ and $\mathcal{D}_1 = \mathcal{M}_+(f_1)$, we require a stopping criteria for any such sequence. That is, given an accuracy $\epsilon = 1/2^N$, $N \in \mathbb{N}$, we require a second algorithm that computes the number of elements of the sequence we need to compute in order to obtain a rational number that is ϵ-close to $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$. Theorem 1 implies that such a stopping criterion does not exist in general.

VI. Uncomputability of Non-Trivial Lower Bounds for the Optimum Distance of Support Sets

In this section, we present our second main result. Every non-trivial lower bound for $(f_0, f_1) \mapsto d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ is *not* a Turing computable function. That is, any (terminating) algorithm that supposedly computes a number smaller than or equal to $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ based on representations of f_0 and f_1 either returns 0 for all inputs or returns erroneous results for some inputs. We first define the notion of a "trivial" lower bound, and then formalize the statement in terms of Turing machines.

In the following, we consider functions $f_0, f_1 \in \mathcal{C}_{\mathrm{c},N}^K(b)$ such that

$$d_{\inf}\big(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1)\big) > 0 \tag{4}$$

is satisfied. Then, consider a Turing machine

$$\underline{\mathrm{TM}} : \mathcal{C}_{\mathrm{c},N}^K(b) \times \mathcal{C}_{\mathrm{c},N}^K(b) \to \mathbb{R}_{\mathrm{c},0}^+$$

such that for all functions $f_0, f_1 \in \mathcal{C}_{\mathrm{c},N}^K(b)$ that satisfy (4), we have

$$d_{\inf}\big(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1)\big) \geq \underline{\mathrm{TM}}(f_0, f_1). \tag{5}$$

Clearly, any Turing machine $\underline{\mathrm{TM}}$ that satisfies $\underline{\mathrm{TM}}(f_0, f_1) = 0$ for all functions f_0, f_1 satisfies this requirement, but is of no practical or mathematical interest. We call such a Turing machine the *trivial* lower bound. In the following, we will investigate the computability of lower bounds that, at least for one specific feasible pair of functions, attain non-zero values, i.e., *non-trivial* bounds.

Theorem 2. *Let $\underline{\mathrm{TM}} : \mathcal{C}_{\mathrm{c},N}^K(b) \times \mathcal{C}_{\mathrm{c},N}^K(b) \to \mathbb{R}_{\mathrm{c},0}^+$ be any Turing machine that satisfies* (5) *for all functions $f_0, f_1 \in$*

$\mathcal{C}^K_{\mathrm{c},N}(b)$ *that satisfy* (4). *Then,* $\underline{\mathrm{TM}}$ *computes a trivial lower bound.*

In Section V, we have discussed the impossibility of finding suitable stopping criteria in the context of evaluating the the maximally achievable performance and stability of a classifier algorithm for the classification regions $\mathcal{D}_0 = \mathcal{M}_+(f_0)$ and $\mathcal{D}_1 = \mathcal{M}_+(f_1)$. If the optimal performance and stability are not of interest, but only a (possibly pessimistic) margin of tolerated error that is guaranteed to be achievable, it is sufficient to have a lower bound for $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ available. According to Theorem 2, however, any such lower bound is either uncomputable or meaningless. Observe that given a classifier $\widehat{L}$ for $\mathcal{D}_0$ and $\mathcal{D}_1$, computing its margin of error would already provide such a bound. Thus, as a corollary of Theorem 2, it follows that the margin of error of a classifier is not algorithmically computable as well.

VII. Identifying Support-Set Points, Computability of Non-Trivial Upper Bounds for the Optimum Distance of Support Sets

In this section, we present our third and fourth main results, which are directly related. For some restricted classes of smooth functions, there exist Turing computable upper bounds for $d_{\inf}$, for example, if there exists a Turing computable function that maps smooth functions (of the restricted class under consideration) to a point within their support set. However, such a Turing computable function does not exist in general. As in the case for lower bounds, we first define the notion of a "trivial" upper bound, and then formalize the first statement in terms of Turing machines and in relation to the second one.

For $N, K \in \mathbb{N}$, $b \in \mathbb{Q}$ with $b > 0$, consider a subset $\mathcal{F} \subseteq \mathcal{C}^K_{\mathrm{c},N}(b)$. such that there exists a computable function $S : \mathcal{F} \to \mathbb{R}^N,\ f \mapsto S(f)$ that satisfies $S(f) \in \mathcal{M}_>(0, f)$ is satisfied for all $f \in \mathcal{F}$. If such a function exists, it immediately provides a computable, non-trivial upper bound for $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+ f_1),\ f_0, f_1 \in \mathcal{F}$. Define

$$\mathcal{M}_+(\mathcal{F}) := \bigcup_{f\in\mathcal{F}} \mathcal{M}_>(0, f).$$

Then, an upper bound for $d_{\inf}$ on $\mathcal{F}$ is called *trivial* if it evaluates to

$$\sup_{x_1\in\mathcal{M}_+(\mathcal{F})} \sup_{x_2\in\mathcal{M}_+(\mathcal{F})} \|x_1 - x_2\|_2$$

for all $f_0, f_1 \in \mathcal{F}$

Theorem 3. *If, for a subset* $\mathcal{F} \subseteq \mathcal{C}^K_{\mathrm{c},N}(b)$ *that contains at least one non-zero element, there exists a computable function* $S : \mathcal{F} \to \mathbb{R}^N,\ f \mapsto S(f)$ *such that* $S(f) \in \mathcal{M}_>(0, f)$ *is satisfied for all* $f \in \mathcal{F}$*, then there exists a Turing machine* $\overline{\mathrm{TM}}_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \to \mathbb{R}^+_{\mathrm{c},0}$ *that satisfies*

$$d_{\inf}\big(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1)\big) \le \overline{\mathrm{TM}}_{\mathcal{F}}(f_0, f_1)$$

for all $f_0, f_1 \in \mathcal{F}$ *and is non-trivial in the above sense.*

The set of non-negative, strictly concave functions in $\mathcal{C}^K_{\mathrm{c},1}(b)$ constitute an example of such a set, since it is always possible to compute $\arg\max_{x\in[0,b]} f(x)$ for non-negative, strictly concave computable functions $f : [0, b] \to \mathbb{R}$. In general, however, the non-existence of computable mappings S of the above type can already be proven under mild assumptions on the set $\mathcal{F}$

Theorem 4. *Consider a subset* $\mathcal{F} \subseteq \mathcal{C}^K_{\mathrm{c},N}(b)$ *that contains at least two elements* f_0, f_1 *with* $\mathcal{M}_+(f_0) \cap \mathcal{M}_+(f_1) = \emptyset$ *and is closed under convex combinations. Then, any mapping* $S : \mathcal{F} \to \mathbb{R}^N,\ f \mapsto S(f)$ *such that* $S(f) \in \mathcal{M}_>(0, f)$ *is satisfied for all* $f \in \mathcal{F}$ *is* not *Turing computable.*

For many reasons, it can be desirable to compute functions $S : \mathcal{F} \to \mathbb{R}^N,\ f \mapsto S(f)$ such that $S(f) \in \mathcal{M}_>(0, f)$ is satisfied for all $f \in \mathcal{F}$. For example, in the context of finding suitable starting points for iterative optimization algorithms. According to Theorem 4, such a functions can only exist for very limited subsets of $\mathcal{C}^K_{\mathrm{c},1}(b)$.

VIII. Uncomputability of Separating Hyperplanes

Last but not least, we present the fifth main result in this section. Every function that maps smooth functions f_0 and f_1 with strictly separated support to a separating hyperplane for $\mathcal{M}_+(f_0)$ and $\mathcal{M}_+(f_1)$ is *not* a Turing computable function. That is, any (terminating) algorithm that supposedly computes a separating hyperplane for $\mathcal{M}_+(f_0)$ and $\mathcal{M}_+(f_1)$ based on descriptions of f_0 and f_1 necessarily returns erroneous results for some inputs.

Form Section II, recall that any hyperplane $\mathcal{H}$ is fully determined by a pair (v, w). This pair is unique up to a scalar multiple. In particular, given any pair (v, w), it is always possible to compute the unique characterization (v', w') of the same hyperplane that satisfies $\|v'\|_2 = 1$.

In the following, we expound that any mapping $H : (f_0, f_1) \mapsto (v, w)$ that returns a separating hyperplane for all functions $f_0, f_1 \in \mathcal{C}^K_{\mathrm{c},N}(b)$ with $\mathcal{M}_+(f_0) \cap \mathcal{M}_+(f_1) = \emptyset$ is necessarily uncomputable.

Theorem 5. *Consider* $N, K \in \mathbb{N}$ *and* $b \in \mathbb{Q}$ *with* $b > 0$*. Let* $H : (f_0, f_1) \mapsto H(f_0, f_1) := (v, w)$ *be a mapping such that for all* $f_0, f_1 \in \mathcal{C}^K_{\mathrm{c},N}(b)$ *with* $\mathcal{M}_+(f_0) \cap \mathcal{M}_+(f_1) = \emptyset$*,* $H(f_0, f_1)$ *characterizes a separating hyperplane for* f_0 *and* f_1*. Then,* H *is* not *Turing computable.*

Despite the previously discussed obstacles in characterizing the achievable stability and performance of classifier algorithms, it might, for some applications, be sufficient to compute separating hyperplanes without any requirements on the resulting margin of error – i.e., it may be arbitrarily small – provided that the hyperplane does separate the classification regions. That is, while an small amount of noise in a new data may lead to an erroneous classification, noiseless data will still be classified correctly with certainty. However, Theorem 5 implies that this problem is unsolvable on digital hardware platforms as well.

IX. Conclusion and Additional Remarks

Throughout this article, we have applied the framework of Turing machines to investigate support-vector machines with regards to a mathematically rigorous formalism of computability for digital hardware. Our results show that in many ways, the theory of support-vector machines is untraceable by digital algorithms, since the relevant mathematical objects – that is, the optimum distance $d_{\inf}$, lower bounds on the optimum distance $d_{\inf}$, support-set points, or separating hyperplanes – are either linked to the classification regions by functions that are not Turing computable, or even are uncomputable objects themselves. In particular, the theory is *fundamentally* untraceable on digital hardware, as a mathematical consequence of Turing's theory. Its infeasibility does *not* emerge from our ignorance of suitable machine-learning algorithm, training data of low quality, or insufficient information about the classification regions.

In the context of machine learning, the problem of computing the pseudo-inverse of a matrices with respect to a strict formalism of computability has recently been considered [20]. It was shown that the mapping $\boldsymbol{A} \mapsto \boldsymbol{A}^{\dagger} := \boldsymbol{A}^{*}(\boldsymbol{A}\boldsymbol{A}^{*})^{-1}$ is not Turing computable, i.e., there exists no algorithm that computes $\boldsymbol{A}^{\dagger}$ based on a description of $\boldsymbol{A}$. However, it was also shown that if $\boldsymbol{A}$ is itself computable, i.e., its entries are computable numbers, then $\boldsymbol{A}^{\dagger}$ is computable as well. Each entry of $\boldsymbol{A}^{\dagger}$ can thus, in principle, be approximated up to arbitrary precision, but the procedure on how to compute such an approximation cannot be determined from a description of $\boldsymbol{A}$. The Π_1-completeness of $d_{\inf}$ is a strictly stronger result. For a suitable choice of f_0 and f_1, the value $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ is uncomputable, i.e., there does not exist an algorithm that approximates the number $d_{\inf}(\mathcal{M}_+(f_0), \mathcal{M}_+(f_1))$ up to arbitrary precision at all. In other words, the target values of $d_{\inf}$ are computationally ill-behaved even though the objects in the domain of $d_{\inf}$ are computationally well-behaved.

Turing's theory of computation is linked directly to integrity assessment. Per definition, integrity refers to the guarantee that a technological system remains within it's prescribed margin of operation. If this margin of operation is controlled by an algorithm on the basis of digital hardware, it is a necessary criterion that the state and dynamics of the system's physical components can be adequately "translated" into the digital realm, i.e., the relevant quantities must be computable. If Turing's theory prohibits such a translation, the integrity of the system as a whole – i.e., its physical components, its digital control hardware, and the control algorithm realized by the latter – is compromised, since the control hardware cannot "understand" the state of the system's physical parts.

With regards to near-future technological systems, the fact that relevant problems in the field automated decision-making can only be "solved" in a heuristic manner on digital hardware needs to be considered. As discussed in the introduction, the next generation of communication systems, as standardized by 6G, relies on the ability to reliably solve mathematical problems by intelligent algorithms in an unprecedented manner. At the same time, these systems can be expected to increasingly affect sensitive human goods, making them potentially hazardous in case of operation outside the intended margin. Accordingly, strict adherence to integrity requirements is fundamentally necessary, with the integrity assessment performed on a mathematically sound and rigorous basis. The authors conclude that in order to provide such a well-grounded technology assessment, a development of the understanding of fundamental limitations of digital hardware with respect to future engineering problems is critically needed. Further, in more general terms, an broader understanding of the relations between theoretical criteria for technology assessment, models of hardware platforms employing different, not necessarily digital, mechanisms for computing, and real-world engineering problems is required.

References

[1] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6g networks," *IEEE Network*, vol. 34, no. 6, pp. 272–280, 2020.

[2] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2017.

[3] G. Fettweis and H. Boche, "6G: The personal tactile internet - and open questions for information theory," *IEEE BITS the Information Theory Magazine*, vol. 1, no. 1, pp. 71–82, 2021.

[4] ——, "On 6G and trustworthiness," *Communications of the ACM*, vol. 65, no. 4, pp. 48–49, 2022.

[5] H. Boche, Y. N. Böck, and U. J. Mönich, "On the arithmetic complexity of the bandwidth of bandlimited signals," *IEEE Transactions on Information Theory*, vol. 69, no. 1, pp. 682–702, 2022.

[6] H. Boche and U. J. Mönich, "Algorithmic computability of the signal bandwidth," *IEEE Transactions on Information Theory*, vol. 67, no. 4, pp. 2450–2471, 2021.

[7] N. V. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[8] ——, *The Nature of Statistical Learning Theory*. Springer, 2000.

[9] T. Xu, T. S. Brisimi, T. Wang, W. Dai, and I. C. Paschalidis, "A joint sparse clustering and classification approach with applications to hospitalization prediction," in *2016 IEEE Conference on Decision and Control (CDC)*, 2016.

[10] Y. Chen, N. Sohani, and H. Peng, "Modelling of uncertain reactive human driving behavior: a classification approach," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018.

[11] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[12] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. s2-42, no. 1, pp. 230–265, 1937.

[13] ——, "On computable numbers, with an application to the Entscheidungsproblem. A correction," *Proceedings of the London Mathematical Society*, vol. s2-43, no. 1, pp. 544–546, 1938.

[14] R. I. Soare, *Recursively Enumerable Sets and Degrees*. Springer, 1987.

[15] M. B. Pour-El and J. I. Richards, *Computability in Analysis and Physics*. Springer, 1989.

[16] K. Weihrauch, *Computable Analysis: An Introduction*. Springer, 2000.

[17] S. C. Kleene, "General recursive functions of natural numbers," *Mathematische Annalen*, vol. 112, no. 1, pp. 727–742, 1936.

[18] A. M. Turing, "Computability and λ-definability," *Journal of Symbolic Logic*, vol. 2, no. 4, pp. 153–163, 1937.

[19] X. Zheng and K. Weihrauch, "The arithmetical hierarchy of real numbers," *Mathematical Logic Quarterly*, vol. 47, no. 1, pp. 51–65, 2001.

[20] A. Fono, H. Boche, and G. Kutyniok, "Limitations of deep learning for inverse problems on digital hardware," in *8th International Conference on Computation Harmonic Analysis*, 2022.