Learning Implicitly with Imprecise Data in PAC Semantics

Alexander Philipp Rader

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2020

Abstract

Data in the real world is often imprecise, such as information coming from sensors. For learning systems that take point-valued data as input, we have to discard the imprecision information and assign each variable an exact value. To alleviate this issue, we propose a learning system that can deal with imprecise information directly by representing variables as intervals. To achieve that, we extend a framework known as Probably Approximately Correct (PAC) semantics, which learns implicitly, i.e. without creating an explicit model, using examples. We formalise the notion of imprecise data and prove that our extension keeps the existing polynomial-time complexity guarantees. Furthermore, we implement the PAC semantics framework for the first time and present usage scenarios using real-life data. We show that the framework works as expected, but some assumptions can be difficult to satisfy in real life.

Acknowledgements

Many thanks to my supervisor Dr Vaishak Belle for providing guidance and help throughout the project, as well as Dr Brendan Juba for always answering the numerous questions I had. I also want to thank Ionela (Gini) Mocanu for discussing and supporting the development of the project in our weekly meetings.

Table of Contents

Chapter 1

Introduction

The most popular machine learning (ML) methods today, such as neural networks, belong to the connectionist paradigm of artificial intelligence. They represent knowledge using distributed networks of nodes with weighted connections. This stands in contrast to the symbolic approach, where knowledge is represented using a highly structured set of symbols, for example propositional logic. In recent years, many fields have been using variations of the neural network with great success, like the transformer in natural language understanding [21] or the convolutional neural network in image processing. [15] There has been a shift away from symbolism and towards connectionist systems, due to their flexibility and performance. [22] Nevertheless, it is still important to analyse and improve alternative models, since they possess unique advantages that can be useful for specific domains.

In this dissertation, we will be extending a symbolic system, called Probably Approximately Correct (PAC) semantics, which uses a subset of first-order logic to represent knowledge. It is part of the learning to reason framework [12], in which an agent learns through interacting with the world and can use logical inference to answer a query. We will propose changes that allow it to directly handle imprecise data. As we will see, the unique features of symbolic representations will allow us to formalise this notion intuitively and make complexity guarantees about it.

1.1 Motivation

The symbolic approach and PAC semantics specifically have many desirable aspects to them. The main advantages for our use-cases are:

- **Powerful knowledge representation:** We represent data using a subset of first-order logic, specifically linear arithmetic in satisfiability modulo theories (SMT), see section 2.2. This allows us to express logical and mathematical relations directly.
- **Implicit learning:** A major criticism of symbolism is the need for manual model construction and feature engineering. [22] However, in the PAC semantics framework, a query is answered using examples from the outside world. The

algorithm does not have to produce an explicit representation of the knowledge and learning is done implicitly. We still have the choice to hand-craft explicit background knowledge, but this model does not have to answer the query by itself. More on that in section 2.1.

- **Partial data:** The way we encode examples allows them to be partially obstructed. This adds robustness, since our model can still answer queries with incomplete data, which often occurs in the real world. This process is formally described in section 3.4.
- **Imprecise data:** In addition to partial data, this dissertation adds the ability to process imprecise data in section 3.3. That means, even if we do not know a value exactly, we can represent it as an interval and still learn from it.
- **Confidence and validity:** As the name suggests, the PAC framework does not give definite answers, but ”probably approximately correct” ones. This means, that you can specify the probability of a query being true and how confident we want to be that our estimate is correct. These notions are formalised in section 3.2.
- **Efficient reasoning:** While first-order logic in general is in NP, the subset of linear arithmetic allows for powerful representations that can be evaluated in polynomial time. [3]
- **Explainability:** Symbolic system built on logic are inherently more explainable than neural networks, which are often black boxes. [8] These transform their input using several layers and learn weights that are very difficult to interpret for humans. Logical formulas, on the other hand, are much more intuitive than representations like vectors. Moreover, queries are answered using inference techniques, which humans can trace and understand.

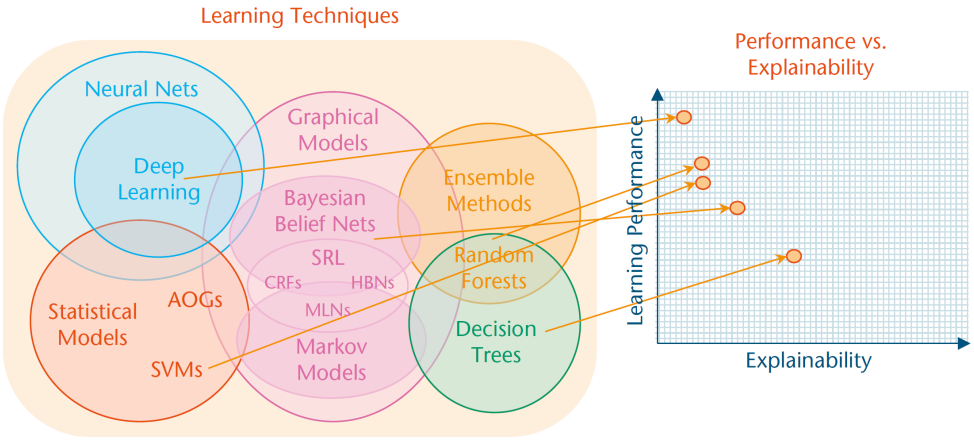

Figure 1.1: Learning performance vs explainability trade-off from [8]

However, connectionist methods methods have an ace up their sleeve: performance. They produce state-of-the-art results for many problems and their structure allows for easy parallelisation using GPUs. [22] The exceptional results make up for their opaque

nature. In fact, there seems to be a trade-off between learning performance and explainability, as [8] point out. They showcase this dichotomy for various ML methods in their paper using a notional graph, which can be seen in Figure 1.1. Decision trees, which provide human-readable decisions, might have better explainability, but worse performance than deep learning methods.

One could tackle this problem by taking high-performing systems and making them more explainable, moving the point to the right. [8] What we do in this dissertation, however, is to take an already explainable system and increase its capabilities, i.e. moving the point upwards. Our proposed extensions will allow the framework to deal with imprecise data, like inaccuracies from sensors. To keep the learning performance high, we prove that the polynomial running-time guarantees still hold. Moreover, we implement the algorithm, which to our knowledge has not been done before, to demonstrate possible applications on real-life data. With these contributions, we hope to showcase the features that PAC and the symbolic paradigm bring and improve upon them.

1.2 Contributions

The first steps in integrating PAC semantics and SMT have been made by [17]. This work builds upon this concept further in the following ways:

- Extending the PAC + SMT framework to handle imprecise data in the form of intervals, rather than just assignments. We formalise this notion by defining a blurring process. (Section 3.3)
- Proving that the complexity for the extended framework stays polynomial-time. (Section 3.6)
- Removing the need for restriction closure, which simplifies many proofs. (Section 3.4.1)
- Implementing the PAC framework in Python and analysing its performance, advantages and drawbacks. (Chapter 4)

Chapter 2

Background

This chapter gives an overview of the framework and concepts that will be used in the next chapter. It will go over all necessary preliminaries in PAC and SMT and set the dissertation in context to other published work over the years.

2.1 Probably Approximately Correct (PAC) semantics

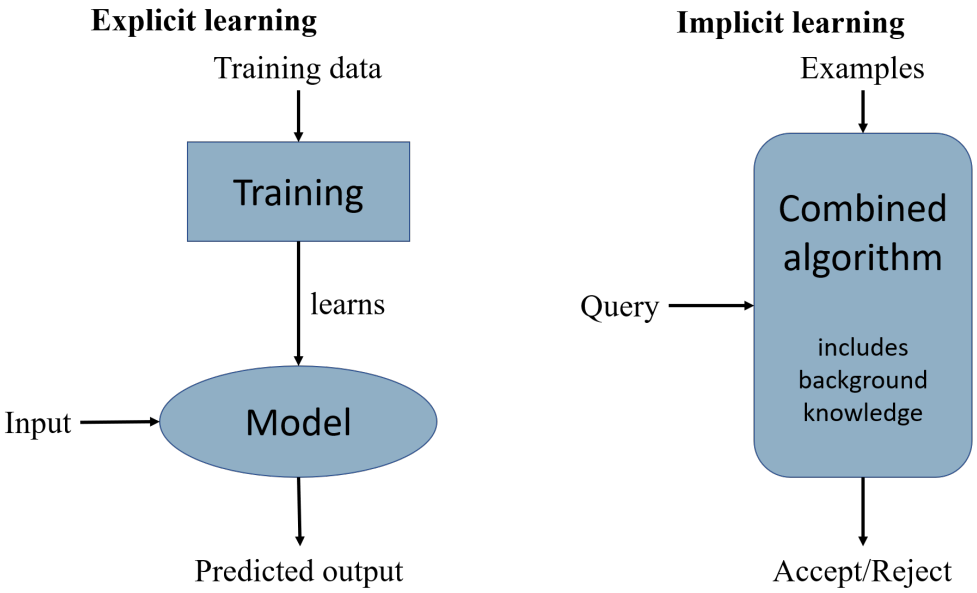

Figure 2.1: Explicit compared to implicit learning

The main difference between PAC semantics and other traditional logic and machine learning algorithms is that PAC semantics is an implicit learning framework. As shown in Figure 2.1, it does not create an explicit model. Instead, a query is answered using both background knowledge and examples. The background knowledge is the only explicit representation, but is not required to be able to answer the query on its own. Hence, it does not have to model every aspect of the data. The algorithm mainly

uses examples to check whether the query is accepted. Implicit learning for PAC was introduced in [20] and adapted to a decision task in [10]. In [17] the framework was combined with SMT, which we take as a basis for this dissertation. We will use the structure and terminology from that paper, which we introduce now.

There are five main components: the query α, the knowledge base Δ, the examples $\{\rho^{(1)}, \rho^{(2)}, \ldots, \rho^{(m)}\}$, validity ε and confidence δ.

- **Query α:** A formula that the user puts into the algorithm and wants to be answered.
- **Knowledge base Δ:** Knowledge we have about the world. Δ is assumed to always be true.
- **Examples $\{\rho^{(1)}, \rho^{(2)}, \ldots, \rho^{(m)}\}$:** Observations from the outside world that help answer the query.
- **Validity ε** : Determines how valid the query has to be in order to be accepted.
- **Confidence δ:** Specifies how well the validity seen from examples generalises to the true underlying validity. The more confident we want to be, the more examples we have to use.

With these components defined, we can use them to construct the decision algorithm from Figure 2.1. On a high level, the algorithm takes in examples and a query, which it then answers by returning either ”Accept” or ”Reject”. It reaches this conclusion by looping through each example and checking whether the query is entailed by the combination of the knowledge base and the current example. The proportion of times the query is accepted makes up the validity. E.g., if the user specifies a validity of 0.7, then at least 70% of the examples have to entail the query to return ”Accept”. The more examples the algorithm can check, the closer the estimated validity comes to the true underlying validity. This concept is captured in the confidence variable δ. We will go through a detailed example in section 3.1.

What makes PAC more robust is the fact that the examples do not have to be complete. The algorithm can still check entailment of the query with partial information. In this dissertation, we will improve robustness further by allowing examples to be intervals rather than assignments, which allows for imprecise measurements. The language in which we represent knowledge will be SMT, which we define now.

2.2 Satisfiability Modulo Theories (SMT)

Satisfiability (SAT) is the problem of deciding whether there exists an assignment of truth values to variables (propositional symbols) such that a propositional logical formula φ is true. Satisfiability modulo theories (SMT) is a generalization of SAT for deciding satisfiability with respect to some background theory. We will now give a formal definition adapted from [3]:

2.2.1 Syntax

A signature is a set of predicate symbols $\mathcal{P}$ and function symbols $\mathcal{F}$. A term t is an expression that can be one of the following: c, $f(t_1,\ldots,t_n)$, where $c \in \mathcal{F}$ with arity 0 and $f \in \mathcal{F}$ with arity $n > 0$. A formula φ is an expression that can be one of the following:

- An atomic formula (or atom) of the form A, $p(t_1,\ldots,t_n)$, $t_1 = t_2$, $\bot$ or $\top$.
- $\neg\varphi_1$, $\varphi_1 \to \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$, $\varphi_1 \vee \varphi_2$ or $\varphi_1 \wedge \varphi_2$.

A literal is an atomic formula or the negation of one, denoted by l. A clause is a disjunction $l_1 \vee \cdots \vee l_n$ of literals. In this work, we focus on the background theory of linear real arithmetic (LRA). The signature $\Sigma_{\mathcal{R}}$ is $(0, 1, +-, \leq)$ and the symbols are interpreted as they are in the usual way over the reals. The reason we do that is because this particular fragment is decidable in polynomial-time, unlike non-linear theories. [3]. This signature is enough to represent any formula in LRA, however, for convenience, we will extend it to include function symbols of the form $\{<, >\leq, \geq, =, \neq\}$.

2.2.2 Semantics

The meaning of a formula is conveyed using a truth value from the set $\{\textbf{true}, \textbf{false}\}$ by means of first-order models. A model $\mathcal{A}$ consisting of:

- a non-empty set A, the universe of the model,
- a mapping $(\_)^{\mathcal{A}}$ assigning
  - to each constant symbol $a \in \mathcal{F}$ an element $a^{\mathcal{A}} \in A$,
  - to each function symbol $f \in \mathcal{F}$ of arity $n > 0$ a total function $f^{\mathcal{A}} : A^n \to A$,
  - to each propositional symbol $B \in \mathcal{P}$ an element $B^{\mathcal{A}} \in \{\textbf{true}, \textbf{false}\}$,
  - to each $p \in \Sigma^P$ of arity $n > 0$ a total function $p^{\mathcal{A}} : A^n \to \{\textbf{true}, \textbf{false}\}$.

2.2.3 Entailment

We say that a formula α is entailed by a formula φ, iff for all assignments that make φ true, α is true as well. We denote this by writing $\varphi \models \alpha$. We can answer entailment using a satisfiability procedure by checking that $\varphi \wedge \neg\alpha$ is unsatisfiable. In linear real arithmetic SMT, specialised theory solvers that check satisfiability procedure exist and can run in polynomial time. [3]

2.3 Related Work

The concept of PAC has existed for decades in different variations. Here, we give an overview of the most important works in the fields and how this dissertation stands in relation to them. Broadly speaking, there are two categories: PAC learning, which creates an explicit model, and PAC semantics, which models knowledge implicitly.

2.3.1 PAC learning

With [19], Valiant introduced the concept of PAC learning. The aim of the paper was to describe the process of learning precisely and study it from a computational viewpoint. Learning was defined as knowledge acquisition in the absence of explicit programming. Instead, a concept was learned using access to positive examples and an oracle. Positive examples would be presented according to a probability distribution, while oracles tell the learning machine whether the data positively exemplifies the concept. As a representation, the paper chose propositional logic with Boolean functions. The conclusions drawn were that there are classes of functions that can be learned in polynomial time, such as conjunctive normal form formulas with a bounded number of literals in each clause. However, it failed to find polynomial-time procedures for many non-trivial classes of functions, such as general CNF expressions, which set serious limits on feasibility. These limitations are the main reasons why many later works, including this dissertation, will follow the implicit PAC semantics paradigm.

[16] enhanced the PAC learning framework to work with partial data. The examples were assumed to be drawn from a probability distribution over binary vectors and then mapped using a masking process to an observation over ternary attributes from $\{0,1,*\}$. No noisy data was allowed, meaning that any unmasked value in an observation had the same value as the example it came from. The system could only sense observations, not the underlying examples. The paper showed that the PAC learnable class of monotone formulas was also learnable from incomplete information. It is related to our work in that we will use the same masking process.

2.3.2 PAC semantics

[12] achieved a breakthrough in the field by introducing the learning to reason framework. They proposed to side-step the issue of learning intractable explicit representations by looking at examples every time a query was put in. A system learned by accessing the world and then used this knowledge to reason about entailment of a query. This framework allowed for efficient solutions even when learning and reasoning separately were not tractable. Examples include the reasoning problem $W \models \alpha$, where α is a $\log n$CNF and W is some CNF formula or a Boolean formula with a polynomial-sized DNF. That paper introduced the fundamental learning structure that we will be using.

Valiant used the PAC framework to create a robust logic in [20]. There, knowledge was represented using rules. This allowed a system to have rules as an input and use examples to further refine its knowledge. The combination of background knowledge and examples was similar to the approach in this paper. It showed that there was a sound and polynomial time inference procedure and that there were polynomial time algorithms for learning the rules from examples. The logic described was also able to cope with partial knowledge, but using a method slightly different than the one we will adopt. There, the logic still just had the two values $\{0,1\}$, and encoded unspecified values using a second predicate for each one, which stated whether the value was specified.

[10] established an implicit learning procedure in PAC for a decision problem in propo-

sitional logic. The framework checked entailment of a query using background knowledge and examples. These examples were obscured using a masking process through partial assignments. The inclusion of partial data made the framework much less trivial than in [12], because not every value was given an assignment anymore. The paper therefore introduced multiple concepts like restriction-closure and witnessing to prove in which cases the limited decision problem was still efficiently solvable with partial data. We will adapt any concepts from that paper for our purposes, particularly the form of the proofs of polynomial-time learnability. But unlike in that paper, which uses propositional logic, our domain will be linear real arithmetic.

[17] expanded the work of [10] to standard fragments of SMT rather than propositional logic. The domain of the background knowledge and the query has been adjusted to SMT, while the examples were still kept as assignments of variables to values from the set of integers. The paper showed the polynomial time complexity for difference logic and linear arithmetic fragments, while non-linear arithmetic was doubly exponential but still sound and complete. We will directly expand upon this paper by considering examples that are intervals rather than assignments.

Other research using the PAC semantics framework has been published recently, such as [4] which deals with combining PAC and first-order logic. Specifically, they proposed a framework to robustly learn universally quantified clauses over a countably infinite domain. No restrictions were posed on concepts like clause length or predicate arity, while maintaining positive results. One might argue that their results automatically include this work, since we are working with a subset of first-order logic. However, we are handling a very specific theory of it and aim to prove polynomial-time learnability, which is not possible for first-order logic in general.

What we have noticed with the papers in this field is that they were all theoretical. They formalised learning concepts and proved their complexity, but never implemented any of them. However, many of their additions and improvements were justified in terms of real-world use cases, such as sourcing examples from the outside world or dealing with incomplete data. With this dissertation, we aim to fill this gap and provide a working implementation in addition to providing rigorous definitions and proofs.

Chapter 3

Theory

We are developing a framework that builds upon the work of [17]. In fact, we are doing exactly what was proposed in the future work section of the conclusion of that paper: extending the framework, so that the examples are intervals rather than assignments. This is useful for two reasons:

- It allows for greater representation power. Often, values cannot be captured precisely as on number, but only up to an interval. Note that we can still specify numbers as a single value with a pair of inequalities. For example, $x = 5$ can be represented as $x \leq 5 \wedge x \geq 5$.

- It makes a concept called restriction closure redundant, which simplifies many proofs. This will be discussed further in section 3.4.1.

Since this extension changes the structure of the input formulas, every step of the algorithm is affected. Therefore, it is necessary to prove which classes of functions remain PAC learnable in polynomial time for the new framework. In this chapter, we detail the exact structure of the framework, go over key concepts like validity and partial observability, state the problem formally and prove polynomial-time learnability.

3.1 An example

To illustrate how the PAC framework arrives at a decision, we will look at a very simplified example. Consider a fitness watch that can measure the heart rate and blood oxygen level of the wearer. Since its sensors are not terribly accurate, it saves the measurements as intervals every few seconds. Moreover, the blood oxygen sensor needs to be very close to the skin of the wearer, or else the measurements are unusable. One feature of the watch is the calculation of a stress level. It calculates stress using the formula

$$stress = heart\_rate - 5 \times (oxygen - 90) \tag{3.1}$$

where heart_rate is is measured in beats per minute and oxygen represents the percentage of oxygen saturation in the blood.[1] If the stress level exceeds the threshold of 50 for, say, 60% of a certain number of measurements, the watch will alert the user and recommend breathing exercises. We will show how our PAC's unique features can be used to arrive at that decision.

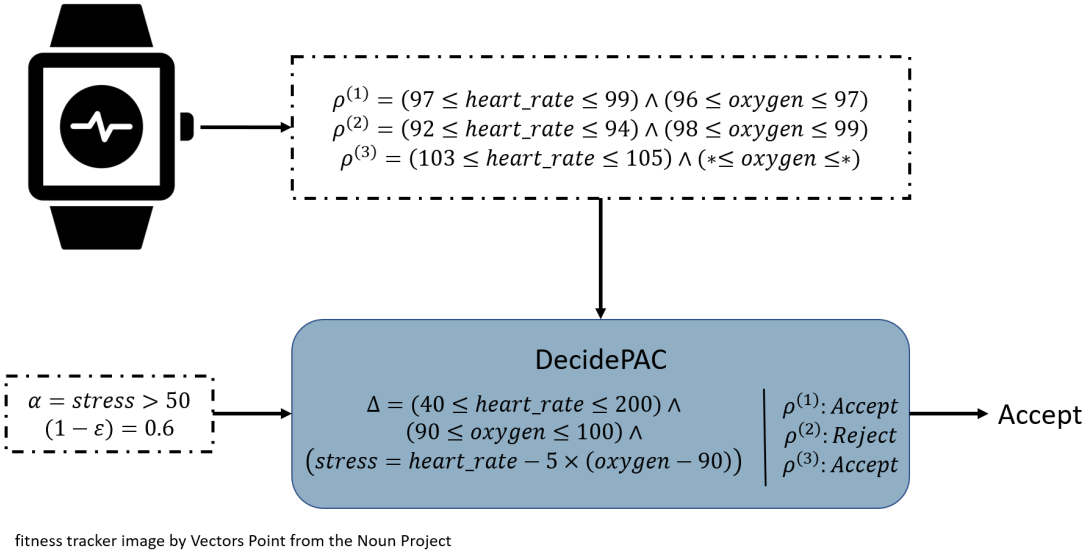

Figure 3.1: High-level example of the framework

A specific example is illustrated in Figure 3.1. First, notice how the DecidePAC algorithm has some explicit background knowledge, Δ, inside it. In this particular case, Δ includes bounds for the heart rate and oxygen saturation, as well as the formula for calculating stress. The first two represent domain knowledge, under the assumption that the stress tester works for people with normal values. The bound for possible oxygen saturation values can be set between 90% and 100%, as anything lower is unnatural and a sign of hypoxemia. [5] Similarly, the heart rates can be bounded by the values 40 and 200. [1] The third value represents the stress calculation set by the designers of the system, which has to be explicitly built into the system. In this toy example, it is a simple linear combination of the two values. A real system would have many more parameters and more complicated functions.

While an explicit model exists, it alone cannot answer the query $\alpha = stress > 50$. Depending on the values of the heart rate and oxygen within the boundaries set in Δ, the stress value might be below or above 50. We need to add knowledge in the form of examples, which in this case the sensor readings of the fitness watch provide. Since the watch might not be tight enough on the wrist, no guarantees can be made about all the sensor readings being complete. In this example, the values for oxygen are missing. As we will show when walking through the process, though, PAC semantics can deal with incomplete data.

The way in which the PAC framework accepts or rejects the query, is by entailment

[1] This formula was chosen for demonstration purposes and of course does not in any way reflect a real calculation of stress level.

of the knowledge base and the examples. It loops through each example and checks entailment.

$$\Delta \wedge \rho^{(n)} \models \alpha \tag{3.2}$$

In our example, α is entailed by $\Delta \wedge \rho^{(1)}$, since any values that make the left side true, yield a value bigger than 50 for stress. Hence, $\rho^{(1)}$ is accepted. For $\rho^{(2)}$, some allowed values make the query wrong. For example, $\{heart\_rate = 92, oxygen = 99\}$ creates $stress = 92 - 5 \times (99 - 90) = 47$. Thus, $\rho^{(2)}$ is rejected. $\rho^{(3)}$ showcases how our framework deals with incomplete data. Since we are working with SMT formulas, unknown values can simply be left out. The last entailment looks as follows:[2]

$$\begin{aligned}(40 \leq heart\_rate \leq 200) \wedge (90 \leq oxygen \leq 100) \wedge \\ stress = heart\_rate - 5 \times (oxygen - 90) \wedge \\ (103 \leq heart\_rate \leq 105) \models stress > 50\end{aligned} \tag{3.3}$$

Even though we do not know the exact interval for oxygen, the broad one from the knowledge base suffices to make the entailment always true. Even for the lowest possible heart rate of 103 and the highest possible oxygen level of 100, the stress level is above 50. Thus, $\rho^{(3)}$ can be accepted despite missing data.

Two out of the three examples are accepted, now the question remains if the whole decision procedure should return accept or reject. This is where the validity comes into play. In addition to the query, a validity of $(1-\varepsilon) = 0.6$ is specified. This means that the query is accepted if at least 60% of the examples are accepted. This is the case, hence DecidePAC returns accept. In the next section, we will formalise this notion of validity and also introduce the concept of confidence.

3.2 Validity and confidence

To start with, we formalise the concept $(1-\varepsilon)$-validity. For that, we use the definition from [17]:

Definition 3.2.1 ($(1-\varepsilon)$-valid)**.** Given a joint distribution D over Σ^n, we say that a Boolean function b is $(1-\varepsilon)$-valid if $\Pr_{\boldsymbol{\rho} \in D}[b(\boldsymbol{\rho}) = 1] \geq 1-\varepsilon$. If $\varepsilon = 0$, we say b is perfectly valid.

Σ^n represents the domain over the reals $\mathbb{R}$. Here, we draw *assignments* $\boldsymbol{\rho}$ from a distribution (note the bold symbol for assignments). We turn these to *intervals* ρ (non-bold) using a separate process in section 3.3. Essentially, these points $\boldsymbol{\rho}$ represent the true underlying values, which we might not be able to sense directly.

In our case, we are interested in the $(1-\varepsilon)$-validity of the query α, given the distribution of the world D. In addition to the examples $\boldsymbol{\rho}$, we also allow access to a knowledge base Δ that allows us to encode explicit information without having to access D. In order for Δ to help to accurately answer α, it has to be consistent with D. Otherwise, it

[2] For conciseness, we are showing intervals as $1 \leq x \leq 3$, even though in proper SMT form it would look like $x \geq 1 \wedge x \leq 3$.

could contradict an example $\boldsymbol{\rho}$. In other words, we assume the knowledge base Δ is perfectly valid with respect to D.

If we knew the underlying distribution D, we could find out exactly how valid a query is. However, this is unrealistic, since we are unlikely to know everything about the world in a real-life scenario. Therefore, in the PAC framework, we only have access to *examples* $\boldsymbol{\rho}$ from D. We have to estimate the actual underlying validity by drawing m examples, calculating the validity (0 or 1) for each one and summing up the results. Since we are dealing with the sum of m independently chosen Bernoulli random variables, we can use Hoeffding's inequality to provide a guarantee that the estimate is close to the real value. [9]

Theorem 1 (Hoeffding's inequality)**.** Let $X_1, \ldots, X_m$ be independent random variables taking values in $[0,1]$. Then for any $\gamma > 0$,

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m} X_i \geq \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} X_i\right] + \gamma\right] \leq e^{-2m\gamma^2}.$$

We can see that we can adjust how close we are to the actual validity with γ. Moreover, the upper bound for the probability that the estimate does indeed only differ by γ depends on the number of examples m as well. We will call this probability our confidence δ. If we give δ and γ as an input, we can calculate how many examples we need for our approximation.

Claim 3.2.1 (Lower bound for examples)**.** The number of examples m needed to reach a margin of deviation γ and confidence δ is $m \geq \frac{1}{2\gamma^2} \ln \frac{1}{\delta}$.

Proof. The confidence δ can be bounded by Hoeffding's inequality.

$$e^{-2m\gamma^2} \leq \delta$$

Using algebraic manipulations we arrive at our claim with the following steps:

$$\begin{aligned}
-2m\gamma^2 &\leq \ln\delta \\
m\gamma^2 &\geq -\frac{1}{2}\ln\delta \\
m\gamma^2 &\geq \frac{1}{2}(0 - \ln\delta) \\
m\gamma^2 &\geq \frac{1}{2}(\ln(1) - \ln\delta) \\
m\gamma^2 &\geq \frac{1}{2}\ln\frac{1}{\delta} \\
m &\geq \frac{1}{2\gamma^2}\ln\frac{1}{\delta}
\end{aligned}$$

□

To sum up, we approximate the $(1-\varepsilon)$-validity of query α given distribution D by drawing m examples $\boldsymbol{\rho}$. The number m varies depending on how close we want to get to the true validity, given by γ, and how confident we want to be, given by δ.

3.3 Modelling imprecision

In many scenarios, however, we do not have access to the true values of examples. Instead, we use sensors to estimate them, which leads to imprecisons. A camera sensor, for example, does not know the exact amount of light in a scene at a time. Instead, it measures the light hitting it over its exposure time and averages it. The PAC + SMT framework has hitherto not taken such imprecisions into account, so we will introduce a formal process in this section. We assume that the true assignments $\boldsymbol{\rho}$ come from the distribution D, but then are blurred and turned into intervals.

Definition 3.3.1 (Blurring process)**.** A *blur* is a function $B : \Sigma^n \to \Sigma^{2n}$, with the property that for any $\boldsymbol{\rho} \in \Sigma^n$, $B(\boldsymbol{\rho})$ is *consistent* with $\boldsymbol{\rho}$, i.e., for each $\boldsymbol{\rho}_i$, $B(\boldsymbol{\rho})_{2i-1} \leq \boldsymbol{\rho}_i \leq B(\boldsymbol{\rho})_{2i}$. We refer to elements of the set Σ^{2n} as *intervals*, where two numbers symbolise the lower and upper bound. A *blurring process* $\mathbf{B}$ is a blur-valued random variable (i.e. a random function). We denote the distribution over intervals obtained by applying the masking process as $\mathbf{B}(D)$.

With this definition, we can formally turn assignments into intervals. If an assignment $\boldsymbol{\rho}$ has n variables, $B(\boldsymbol{\rho})$ will have $2n$, where the entries $2i-1$ and $2i$ represent the lower and upper bound of the interval respectively. We will denote the the transformation $\mathbf{B}(\boldsymbol{\rho})$ as ρ with the non-bold symbol. The consistency property makes sure that the true value from the assignment lies within the interval. From now on, we will only work with intervals ρ. In case we do have exact information about a variable, we can keep the lower and upper bound as the same value.

3.4 Partial observability

Sometimes, the framework might not have access to all variables in an example, such as the fitness tracker example from section 3.1, which does not always get blood oxygen readings. We will formalise the notion of partial data using a masking process. The examples $\rho^{(1)}, \ldots, \rho^{(m)}$ do not come directly from D, but from $M(D)$, where M is a mask. Values from D can be masked by giving them a value of $*$, but the actual value cannot be changed. This means that data remains noiseless, but partially obscured. We modify the definition for a masking process from [17], since we now have intervals instead of assignments as examples.

Definition 3.4.1 (Masking process)**.** A *mask* is a function $M : \Sigma^{2n} \to \{\Sigma \cup \{*\}\}^{2n}$, with the property that for any $\rho \in \Sigma^{2n}$, $M(\rho)$ is *consistent* with ρ, i.e., whenever $M(\rho)_i \neq *$ then $M(\rho)_i = \rho_i$. We refer to elements of the set $(\Sigma \cup \{*\})^{2n}$ as *partial intervals*. A *masking process* $\mathbf{M}$ is a mask-valued random variable. We denote the distribution over partial intervals obtained by applying the masking process as $\mathbf{M}(D)$.

The $*$ value is of course not part of the universe in the theory of linear arithmetic. Hence, by the time entailment is decided, those values have to be dealt with. The way we do it is by defining a function $\downarrow$ that turns the set of numbers ρ into an SMT formula.

Definition 3.4.2 ($\downarrow$)**.** For a set of partial intervals $\rho = \{\rho_1, \rho_2, \ldots, \rho_{2n}\}$, we define $\rho\downarrow$

as the following SMT formula: $\rho \downarrow = x_1 \geq \rho_1 \wedge x_2 \leq \rho_2, \ldots, x_n \geq \rho_{2n-1} \wedge x_n \leq \rho_{2n}$ for all ρ_i that are defined. Any ρ_i that have the value $*$ are left out.

Say we have $\rho = \{0 \leq x \leq 1, 2 \leq y \leq *, * \leq z \leq *\}$, then $\rho \downarrow = (x \geq 0) \wedge (x \leq 1) \wedge (y \geq 2)$.

3.4.1 A note on restriction closure

This simple and intuitive way of dealing with partial data poses one of the big advantages over using assignments. In [10] and [17], partial assignments made it necessary to define the concept of restriction closure, which we can bypass completely. Since examples were assignments, having partial examples meant that some variables were not given values. They needed to show that if $\Delta \models \alpha$, then for the restriction closed form, denoted by $|_\rho$, $\Delta|_\rho \models \alpha|_\rho$. In essence, this means that if a decision procedure exists for a formula, the same decision procedure works for any partial assignments of α and δ.

However, we are not using assignments as examples anymore, but intervals, meaning that we want to show that $\Delta \wedge \rho \downarrow \models \alpha$. Since we don't restrict any variables in Δ or α, masking does not change the values of any variables, but only makes ρ shorter. The decision procedure thus says the same, making restriction closure obsolete.

Providing guarantees and finding decision procedure that adhere to restriction closure to SMT was focus of [17]. In this paper, we have gotten rid of it, which that makes the process of proving properties about our framework easier. It might seem like a more obvious solution, but is in fact a step forward from what was done in earlier work.

3.5 Witnessing

Before we can prove implicit learnability, we have to adjust the definition of witnessing. In [10], witnessing meant that the formula evaluates to true given the partial assignments of its variables. Here, we have partial intervals, which means that we have multiple choices of assignments for each variable. We therefore define witnessing in terms of entailment.

Definition 3.5.1 (Witnessed formulas)**.** We define a formula to be witnessed to evaluate to true given partial intervals as follows: A formula φ is witnessed true under partial intervals ρ if $(\rho \downarrow) \models \varphi$, i.e. φ is true under every assignment possible under ρ.

Witnessing is an important part of the proof in the next section. It makes sure that a formula can still be evaluated to true, despite missing values and intervals.

3.6 Polynomial-time implicit learnability

Now we will prove three major theorems, which we adapt from [17]:

1. SMT formulas for linear arithmetic with intervals as examples can be learned implicitly.

2. Given a polynomial-time satisfiability procedure for linear arithmetic, implicit learnability runs in polynomial time.

3. There exists a polynomial-time satisfiability procedure for linear arithmetic.

By proving all three theorems, we will have shown that our extended framework has maintained polynomial-time implicit learnability.

Algorithm 1: DecidePAC

input : Algorithm A, formula α, variables $\varepsilon, \delta, \gamma \in (0,1)$, list of partial intervals $\{\rho^{(1)}, \rho^{(2)}, \ldots, \rho^{(m)}\}$, knowledge base Δ

output: Accept if there exists a proof of α from $\Delta \wedge I$ and examples $\{\rho^{(1)}, \rho^{(2)}, \ldots, \rho^{(m)}\}$ on $M(D)$ with probability at least $(1-\varepsilon+\gamma)$
Reject if $\Delta \models \alpha$ is not $(1-\varepsilon-\gamma)$-valid under D

$B \leftarrow \lfloor \varepsilon \times m \rfloor, FAILED \leftarrow 0$
foreach *i in m* **do**
  if $A(\alpha, \rho^{(i)}, \Delta)$ *returns UNSAT* **then**
    Increment *FAILED*
  if *FAILED* $> B$ **then**
    return *Reject*
return *Accept*

Theorem 2 (Implicit learning)**.** Let Δ be a conjunction of constraints representing the knowledge base and an input query α. We draw at random $m = \frac{1}{2\gamma^2} ln \frac{1}{\delta}$ partial intervals $\{\rho^{(1)}, \rho^{(2)}, ..., \rho^{(m)}\}$ from $\mathbf{M}(D)$ for the distribution D and a masking process $\mathbf{M}$. Suppose that we have a sound decision procedure A. There exists an algorithm using A such that, with probability $1-\delta$,

- if $(\Delta \Rightarrow \alpha)$ is not $(1-\varepsilon-\gamma)$-valid with respect to the distribution D, the algorithm returns Reject; and
- if there exists some KB I such that $\Delta \wedge I \models \alpha$ and I is witnessed true with probability at least $(1-\varepsilon+\gamma)$ on $\mathbf{M}(D)$, then Algorithm 1 returns Accept.

Proof. We modify the proof from [10] for our framework. Suppose we run algorithm 1 on $m = \frac{1}{2\gamma^2} ln \frac{1}{\delta}$ examples drawn from D.

For the first point, note that if $\Delta \Rightarrow \alpha$ is not $(1-\varepsilon-\gamma)$-valid with respect to D, an interpretation sampled from D must satisfy Δ and falsify α with a probability of at least $\varepsilon+\gamma$. Now, if we mask some of the variables from this particular interpretation by sampling a ρ from $\mathbf{M}(D)$, then $\Delta \wedge \rho \downarrow \not\models \alpha$. This is because the original interpretation with all intervals specified already did not entail α. By removing some of the intervals, the values that falsify α are still included and the entailment still does not hold. Thus, for any $\rho^{(i)}$ from $\mathbf{M}(D)$, A rejects with probability at least $\varepsilon+\gamma$. Algorithm 1 calls A for m examples as specified at the beginning of the proof. We know from 3.2.1 that m examples suffice to estimate the validity up to γ with a probability of $(1-\delta)$.

For the second point, suppose that there exists some I such that $\Delta \wedge I \models \alpha$ and I is witnessed true with probability at least $(1-\varepsilon+\gamma)$ on $\mathbf{M}(D)$. I represents the implicit

knowledge base, which consists of the partial intervals and any derivations made in the entailment proof. Hence, for each ρ drawn from $\mathbf{M}(D)$, A will accept the entailment with a probability of at least $(1-\varepsilon+\gamma)$. Again, claim 3.2.1 ensures that γ and $(1-\delta)$ are correct. □

The key assumption that our proof makes is that the implicit knowledge base I is witnessed true with a probability with at least $(1-\varepsilon+\gamma)$ on the partial intervals. To reiterate, I takes the form of the derivation steps to decide entailment of the query from the knowledge base and examples. To be a valid derivation, each step has to be true, of course, which is why we require witnessing. What this theorem essentially states, is that implicit learnability on partial intervals is only possible if the values are masked in such a way that entailment can still be decided with a high probability. Arbitrary masking is therefore not allowed. This makes sense, since if we for example mask every single value, then the examples would give us no new information. In our implementation in section 4.4.1 we will demonstrate that arbitrary masking does indeed not uphold a good estimate of the true validity.

We have shown that our extended framework is still implicitly learnable given a sound decision procedure A. Since such decision procedure exist entailment of SMT formulas in linear arithmetic [17], implicit learning is indeed possible. Note that the witnessing assumption was made in [17] already, so the results remain the same for our extension. Now we show that the implicit learnability stays polynomial-time.

Theorem 3 (Polynomial-time learnability)**.** Given an input of $m = \frac{1}{2\gamma^2} ln \frac{1}{\delta}$, algorithm 1 runs in time $O(T(A)\frac{1}{2\gamma^2}\ln\frac{1}{\delta})$, where $T(A)$ is the running time of decision procedure A.

Proof. Line 1 runs in constant time. The foreach loop runs $m = \frac{1}{2\gamma^2} ln \frac{1}{\delta}$ times. In each iteration of the loop, procedure A is called once and some constant work is being done. Thus, the complexity is $O(1) + O(\frac{1}{2\gamma^2}\ln\frac{1}{\delta}T(A) + O(1)) = O(T(A)\frac{1}{2\gamma^2}\ln\frac{1}{\delta})$. □

Whether that makes the algorithm polynomial-time depends on the running time of A. For satisfiability procedures in linear arithmetic, such a procedure exists. This was proven in [17], which we will present below for completeness.

Theorem 4. Let $\delta, \gamma, m, \varepsilon, \Delta, \alpha, \rho^{(i)}$ and $\mathbf{M}$ be as introduced above. Suppose we solve a linear program using the procedure from [6] which runs in time $O(n^{\omega+o(1)}\log(\frac{n}{\delta}))$, where n is the number of variables in the program and ω is the current matrix multiplication exponent ($\omega \approx 2.373$). Then Algorithm 1 using this decision procedure returns Accept or Reject in time $O(n^{\omega+o(1)}\log(\frac{n}{\delta})\frac{1}{2\gamma^2}\log\frac{1}{\delta})$.

Proof. We use the algorithm of Cohen et al. for solving a system of intervals, which runs in time $O(n^{\omega+o(1)}\log(\frac{n}{\delta}))$. For every iteration of the main algorithm, the system of intervals is evaluated in conjunction with some observed set of intervals ρ, which is logically equivalent to evaluating whether $KB \wedge \rho\downarrow \models \alpha$. To decide entailment, we need determine whether the set of intervals obtained from that statement all together offer a feasible solution. Every iteration costs the time for checking feasibility, bounded by $O(n^{\omega+o(1)}\log(\frac{n}{\delta}))$. The total number of iterations is $\frac{1}{2\gamma^2}\log\frac{1}{\delta}$, corresponding to the

number of samples drawn, hence the total time bound is $O(n^{\omega+o(1)} \log(\frac{n}{\delta}) \frac{1}{2\gamma^2} \log \frac{1}{\delta})$. □

3.7 Discussion

To sum up, polynomial-time learnability is preserved under our extension for linear arithmetic. If we increase the theory to non-linear arithmetic, the decision procedure is decidable [18], but unfortunately double exponential in the worst case. [23] Nevertheless, linear arithmetic is much stronger than propositional logic and in this work we have extended expressivity further by allowing examples to be intervals. Thus, the results are positive and we have achieved the main objective of the dissertation.

It might seem odd, however, that we now allow examples to be less precise and can still guarantee that the estimated validity is arbitrarily close to the underlying validity, just as in [17]. Even the number of examples needed for a required confidence has stayed the same, despite those examples now being less exact. While our guarantees might not have changed, it has in fact become more difficult to estimate the validity of the query correctly. The crucial detail lies in the definition of witnessing. Recall that we only give guarantees about implicit learning in theorem 2 if there exists some KB I that is witnessed true with high probability. Our extension has made it harder for this assumption to hold.

To illustrate this, imagine a scenario where the query is only true if the constraint $\varphi = x_1 - x_2 > 0$ is witnessed true under the partial intervals. For the assignments $\boldsymbol{\rho} = \{x_1 = 2, x_2 = 1\}$, the formula would be witnessed true, as that is what it evaluates to. However, let us say these readings are inexact and the values can lie anywhere between $\rho = \{1.5 \leq x_1 \leq 2.5, 0.5 \leq x_2 \leq 1.5\}$. Now φ is not witnessed true anymore, since the values $x_1 = 1.5, x_2 = 1.5$ falsify the formula. Even though φ holds for all but one pair of assignments in the interval, it is not witnessed true anymore.

In that regard, our notion of witnessing is rather strict. One could argue that it might be too strict and we should accept the query, say, when I is true for the majority of possible values within the intervals. However, with a looser definition we would not be able to give any implicit learning guarantees anymore. We do not know the underlying distribution, so even if there is only one falsifying pair in our example above, it might occur 99% of the time. The purpose of theorem 2 is to give guarantees for cases where it is clear whether the query is valid or not given the intervals. Since intervals are meant to model measurement imprecision, they shouldn't be too big. In edge cases such as above, we cannot give any guarantees precisely because they are edge cases.

Whether such assumptions are helpful in the real world can really only be determined by using the framework in practice. Therefore, we have implemented it and will look at potential use-cases of PAC on a real dataset in the next chapter.

Chapter 4

Implementation

We have achieved the original goal of the project by proving polynomial-time learnability in linear arithmetic for our extended framework. PAC was originally introduced as a theoretical framework to formalise learning and prove feasibility bounds. [19]. However, much of [17] and this dissertation has extended the framework with real-life problems in mind. In [17], it was combined with SMT, which has many practical applications in areas such as software verification, type inference, program analysis, scheduling and graph problems. [7] In this dissertation, we motivated our extensions with the ability to model imprecise data, which could come from sensors. Therefore, it is natural to try and actually implement our framework. To the best of our knowledge, no implementation exists for PAC, and of course no implementation exists for our specific framework, since we introduced it here. This chapter will look at the challenges of creating an implementation, explore ways to use it in real-life settings and uncover which assumptions made in the theoretical framework are difficult to satisfy practically.

4.1 Why use PAC in the real world?

Different aspects of our framework have qualities that could prove to be helpful in a real-world setting:

- **Polynomial time complexity:** If all the conditions are met (consistent background knowledge, high probability of witnessing, etc.), the algorithm can answer queries in polynomial-time, even if creating a model is intractable.
- **Incomplete data:** The masking process is modelling the idea of missing data. Real-world data can be incomplete for many reasons, such as human error, sensor failures or difficulty of collecting all data points. PAC can not only deal with missing data, but can still give guarantees about learnability. We will explore the effect of missing data in section 4.4.
- **Imprecise data:** The main contribution of this dissertation is dealing with imprecise data, i.e. allows intervals as examples. Moreover, since the domain is SMT, all data, including background knowledge, can be expressed in terms of

linear arithmetic. This allows for compact and logical modelling of data by turning datapoints into inequalities, as we will do in section 4.4.

- **Approximately correct answers:** Often, one cannot be 100% sure of the answer and the framework can deal with that. Using confidence and validity, the programmer can adjust how uncertain the system is allowed to be.
- **Using as much data as needed:** The number of examples the algorithm uses can be adjusted with the confidence and γ parameters.. Therefore, it does not necessarily need to query the whole dataset, which might save time. The user can adjust the trade-off between running time and precision.
- **No explicit model is created:** Each query is answered in real-time using examples instead of modelling all the data. This is a useful properties in situations where the examples change over time and thus the model would have to be continually adapted.

4.2 The code

We implemented the framework in Python using the Z3 theorem prover[1] for SMT formulas. The code is accessible at GitHub.[2]. For instructions on how to get the code running on your machine, please refer to appendix A. In the next sections, we will use this code to demonstrate use-cases and analyse various aspects of implementing PAC.

4.3 Finding a suitable dataset

To analyse the usefulness of PAC in a real-life scenario, we wanted to implement it on a real dataset.

First, we looked at the Satisfiability Modulo Theories Library (SMT-LIB). [2] It contains many benchmarks on numerous theories, including the one we are covering: quantifier-free linear real arithmetic. [3] However, these are benchmarks for SMT solver implementations, which means they were created to test the efficiency and correctness of solvers using complicated SMT formulas. Most of contain dozens or hundreds of variables, which are then used in a single, very long formula to check satisfiability. There is no way to easily split up this formula into a knowledge base and a query. Moreover, we would have to synthetically create examples, which would defeat the purpose of finding real-life applications.

The dataset needed a structure that fit well with PAC, i.e. contains background knowledge that is always true, examples we can source from and a helpful query. Such datasets definitely exist, however, their data is not in SMT form. The problem is that our framework combines two separate fields: PAC and SMT. This has not been done before [17], which came out in 2020. Thus, no dataset combining both fields exist.

[1] `https://github.com/Z3Prover/z3`

[2] `https://github.com/APRader/pac-smt-arithmetic`

[3] They can be found at `http://smtlib.cs.uiowa.edu/benchmarks.shtml`

SMT datasets do not fit well with PAC, and datasets that work with PAC do not contain SMT formulas.

We did not want to use synthetic data either, since the purpose of this section is to gauge the feasibility of real-life use of our framework. To resolve this predicament, we decided to use an existing machine learning dataset and convert it into a suitable form. The dataset we chose is the Kaggle electric motor temperature dataset[4], which was created from [14] and [13]. It includes almost a million lines of measurements of various motor component sensor data from multiple sessions. Each session is given a *profile_id*. The *profile_id*s of 65 and 72 denote the test set, the rest are training data.

The reasons we chose this dataset as an example to demonstrate the applicability of our framework are:

- There are nice queries you can represent in SMT regarding motor component and temperature measurements. For example, if you want to know whether the the temperature drop-off from the inside of the motor (pm) and the outside (ambient) is always larger than, say, 2 degrees, you can express that as a query as follows:

$$pm - ambient > 2$$

- The whole dataset consists of datapoints measured every 0.5 seconds. Thus, they lend themselves to be put in terms of intervals. This can drastically reduce the size of the dataset, by combining many rows into one expression.
- The task uses real-time sensor data, which is what the PAC framework is based on.
- PAC can deal with incomplete data, which is useful for this task in case a sensor breaks during motor use.
- You can specify confidence and validity, depending on the importance of knowing which components might be overheating and running-time constraints. This allows fine-grained control of the procedure.

It is a machine learning dataset containing datapoints, with the goal of predicting values from the test set. Specifically, the target features are rotor temperature (”pm”) and stator temperatures (”stator_*”), since they are not easily measurable. Of course, the PAC decision framework is about answering queries, not predicting values. That is why we will create our own tasks and analyses in 4.4. However, in 4.5 we will attempt to adapt the framework to a more traditional prediction task and discuss the challenges.

4.4 Usage scenario

There are many possible latent variables one could calculate with the 12 variables present in the dataset. Let us focus on one scenario: Say one gets current (i_d and i_q) and voltage (u_d and u_q) readings from the motor. These readings are inexact, so the values come as intervals and sometimes are missing. Throughout the use of the motor,

[4] https://www.kaggle.com/wkirgsn/electric-motor-temperature

the task is to check that the effective power lies above a certain threshold. Since power is given by voltage*current, we can represent this query as an SMT formula:

$$\alpha = i\_d \times u\_d + i\_q \times u\_q > threshold$$

With the query defined, we still needed to create the knowledge base and examples from the dataset. One major constraint of the framework is that the knowledge base and examples come from the same distribution. Otherwise, they could contradict each other when we are putting them in a conjunction. The way we solved it is to make the knowledge base very broad and then sample examples from the same data. The dataset is divided into many sessions, each given a unique profile id. We treat each measuring session separately and create a knowledge base Δ from them. Δ is created by taking the minimum and maximum value of each variable and turning them into intervals. Similarly, we create the examples from the same set by turning every 10 rows into an interval.

The above example uses a non-linear formula, which Z3 can principally handle. However, unlike linear arithmetic, the time complexity is doubly exponential [3]. To see how much of an effect it has in practical use, we ran the algorithm for 25000 instances on both the non-linear formula above and a linear version, where we replaced the multiplications with additions. The results showed that the decision procedure could process 885 queries on average for the linear version, but only 78 on the non-linear version per second, which is more than 10 times less. This was run on a laptop and not under any controlled environment, but still gives us a feel of how much the choice of domain matters. If a function is viable to compute depends on the use case. If a system only needs to process 10 examples per second, then a non-linear function can still be feasible.

4.4.1 Validity vs masking

An interesting question we can analyse with the implementation is how random masking affects validity. In theorem 2, we only guarantee the estimated validity of masked intervals to be accurate if the derivation proof for the query is still witnessed true with a probability of $(1-\varepsilon+\gamma)$. This means that the values being masked should not hinder the ability of the query being answered. Random masking does not obey this assumption. We ran the above query on the dataset with different probabilities of random masking. Figure 4.1 shows the relation between validity and random masking of the query on 10 different sessions.[5]

The validity values for 0 masking represent an accurate estimate of the true validity, since correct instances are always witnessed true trivially. It is quite apparent that the validity is underestimated the more we mask. This decline is quite rapid, especially considering that we only use 4 out of the 12 variables in the query. Thus, masking any other variables doesn't affect its validity and a 10% masking rate only affects 3.33% of the necessary variables.

[5]For the threshold we used the value 0, since all the variables are standardised and centred at 0. Of course, the query does not make sense semantically anymore for standardised values, however we are looking at the effects of masking.

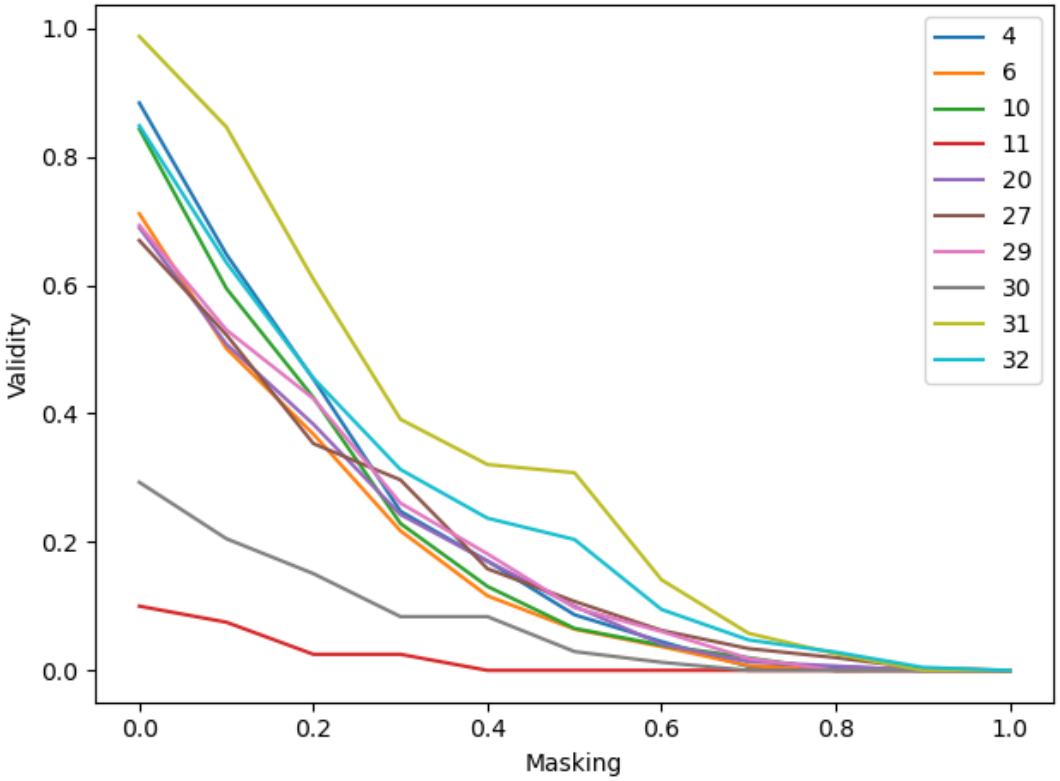

Figure 4.1: Validity of query compared to probability of masking per session

The rapid decline in estimation accuracy shows how important adhering to the witnessing assumption is. The framework is not accurate for noisy data loss, since even small amount of random masking can have a big effect. Instead, masking should represent small, relatively predictable and non-essential missing data.

4.4.2 Discussion

One might ask why the system could not just calculate those latent variables like power directly. While this might be sufficient in many cases, using SMT allows for the data to be in interval form and missing values, provided all assumptions are met.

However, this example task is indeed very arbitrary. Beyond looking at the running time and the validity vs masking trade-off, there is not much to measure. And even the running time mostly depends on the efficiency of the Z3 solver, which is not something we implemented. The fundamental problem is that there are no benchmarks or baselines for this PAC + SMT system. There are not even other implementations to compare it with. Of course, the implementation is arguably useful in its own right. It is a tool for showing potential use-cases and for other people to integrate into their systems. However, for the implementation to be truly useful, we would either have to use it on a real system, like an actual motor, which was beyond the scope of this dissertation.

Another issue is that the dataset is built for a typical machine learning task. It consists

of training and test data and the goal is to predict variables in the test set. We wanted to see if we could adapt PAC for such a prediction task, as that would give us baselines and performance measures to test again. Our process is described in the next section.

4.5 Adapting PAC for prediction

The aim of the data set is to predict whether certain components of the motor are overheating. Specifically, to predict the values *stator_yoke*, *stator_winding*, *stator_tooth* and *pm*. These are temperatures of motor components that cannot be easily measured during use of the motor. All the other values can be measured during use, and thus can be utilised to predict those target features. Our idea was to use the training set as examples and the test set as background knowledge. We could then answer a query, such as if the motor component is overheating, by looking at similar examples to our current observation and checking in how many of those examples the component was overheating. In more detail, we followed the following steps:

1. Formulate the query α that asks if the target features are overheating.
2. Transform the dataset into SMT expressions.
   (a) Take an observation Δ of the current motor state and use that as background knowledge. In our case those will be the rows of the test set, excluding the target features.
   (b) Transform the rows of the training set into examples $\rho^{(1)}, \ldots, \rho^{(n)}$, which include the target features.
3. Choose m examples where $m \leq n$ that have non-target features that match with the observation Δ.
4. Check entailment of $\Delta \wedge \rho^{(i)} \models \alpha$.

We will illustrate the process using a toy example. Assume we have the datasets shown in Tables 4.1 and 4.2, where stator_yoke is the target feature.

4.5.1 Formulating the query

Assume we can observe the values ambient and coolant and want to predict if stator_yoke is overheating, as we cannot measure that value. Say that stator_yoke is overheating if it exceeds 100. Our query thus is $\alpha = stator\_yoke > 100$.

4.5.2 Transforming the data sets

We want to turn the training and test data into intervals. A natural way to do this is to take n rows and storing the interval of values. n can be varied depending on how much one wants to compress the data. Let $n = 3$ for the training set. Our examples are then defined as follows:

$$\rho^{(1)} = ambient \geq 45 \wedge ambient \leq 47 \wedge coolant \geq 4 \wedge coolant \leq 5 \\ \wedge stator\_yoke \geq 101 \wedge stator\_yoke \leq 104$$

	ambient	coolant	stator_yoke
1	45	5	101
2	46	5	102
3	47	4	104
4	42	3	95
5	42	2	94
6	43	2	95
7	46	5	103
8	44	5	103
9	47	4	104
10	45	7	99
11	48	4	98
12	46	5	99

Table 4.1: Training set

	ambient	coolant	stator_yoke
1	45	5	102
2	46	4	103

Table 4.2: Test set

$$\rho^{(2)} = ambient \geq 42 \wedge ambient \leq 43 \wedge coolant \geq 2 \wedge coolant \leq 3 \\ \wedge stator\_yoke \geq 94 \wedge stator\_yoke \leq 95$$

$$\rho^{(3)} = ambient \geq 44 \wedge ambient \leq 47 \wedge coolant \geq 4 \wedge coolant \leq 5 \\ \wedge stator\_yoke \geq 103 \wedge stator\_yoke \leq 104$$

$$\rho^{(4)} = ambient \geq 45 \wedge ambient \leq 48 \wedge coolant \geq 4 \wedge coolant \leq 7 \\ \wedge stator\_yoke \geq 98 \wedge stator\_yoke \leq 99$$

The test set represents our current measurements of the running motor. We do not know stator_yoke, but will predict whether it is overheating (i.e. > 100) using PAC and then compare it with the actual value from the test set (looking at Table 4.2, it is in fact overheating). Let $n = 2$ for the test set and we get

$$\Delta = ambient \geq 45 \wedge ambient \leq 46 \wedge coolant \geq 4 \wedge coolant \leq 5$$

4.5.3 Choosing proper examples

We have four examples from our training set which we can use to predict whether our motor is overheating. However, not all of these reflect the state our motor is currently in. The training set consists of measurements from various sessions in which the motor was in different states. We only want to use those examples, where the values ambient and coolant are similar. Then, checking whether stator_yoke was overheating makes sense, since the conditions are comparable. In terms of SMT, this can be checked with the expression

$$\beta^{(i)} = \Delta \models (\rho^{(i)} - stator\_yoke)$$

If β is true, then the observation Δ is within the example $\rho^{(i)}$ and the example is suitable. Of, course for this to be practical, the example interval $\rho^{(i)}$ should be larger than the observation Δ. One way to make this happen statistically, is to convert more rows from the training set (e.g. $n = 20$) into intervals and only take a few (e.g. $n = 3$) measurements from observations (in our case the test set). Now, let us see which examples

are appropriate:

$$\beta^{(1)} = ambient \geq 45 \wedge ambient \leq 46 \wedge coolant \geq 4 \wedge coolant \leq 5 \models ambient \geq 45 \wedge ambient \leq 47 \wedge coolant \geq 4 \wedge coolant \leq 5 = \top$$

$$\beta^{(2)} = ambient \geq 45 \wedge ambient \leq 46 \wedge coolant \geq 4 \wedge coolant \leq 5 \models ambient \geq 42 \wedge ambient \leq 43 \wedge coolant \geq 2 \wedge coolant \leq 3 = \bot$$

$$\beta^{(3)} = ambient \geq 45 \wedge ambient \leq 46 \wedge coolant \geq 4 \wedge coolant \leq 5 \models ambient \geq 44 \wedge ambient \leq 47 \wedge coolant \geq 4 \wedge coolant \leq 5 = \top$$

$$\beta^{(4)} = ambient \geq 45 \wedge ambient \leq 46 \wedge coolant \geq 4 \wedge coolant \leq 5 \models ambient \geq 45 \wedge ambient \leq 48 \wedge coolant \geq 4 \wedge coolant \leq 7 = \top$$

Thus, we can use examples 1,3 and 4.

4.5.3.1 Checking entailment

Now we run the Decide-PAC algorithm with examples $\rho^{(1)}, \rho^{(3)}, \rho^{(4)}$ and check entailment.

$$stator\_yoke \geq 101 \wedge stator\_yoke \leq 104 \models stator\_yoke > 100 \Rightarrow ACCEPT$$

$$stator\_yoke \geq 103 \wedge stator\_yoke \leq 104 \models stator\_yoke > 100 \Rightarrow ACCEPT$$

$$stator\_yoke \geq 98 \wedge stator\_yoke \leq 99 \models stator\_yoke > 100 \Rightarrow REJECT$$

In this case, we would get 0.67-validity, since stator_yoke was only overheating in 2 out of 3 examples.

4.5.4 Discussion

This implementation is quite a departure from the original PAC algorithm. It uses the observation as the background knowledge and the training set as examples. Moreover, the examples are selected in terms of their similarity to the observation to fulfill the requirement that the background knowledge is consistent with the examples.

When we implemented the algorithm in practice, we ran into a big problem, which is illustrated in Figure 4.2. We ran 1000 observations through the matching process, seeing with how many out of 50,000 examples each observation lies. As is visible in the bar chart, the vast majority of observations do not match with any examples. The problem is that the test points come from a different underlying distribution than the training points. The observations that we were trying to match with examples are atypical, which is why no matching ones are found. In general, finding boxes that

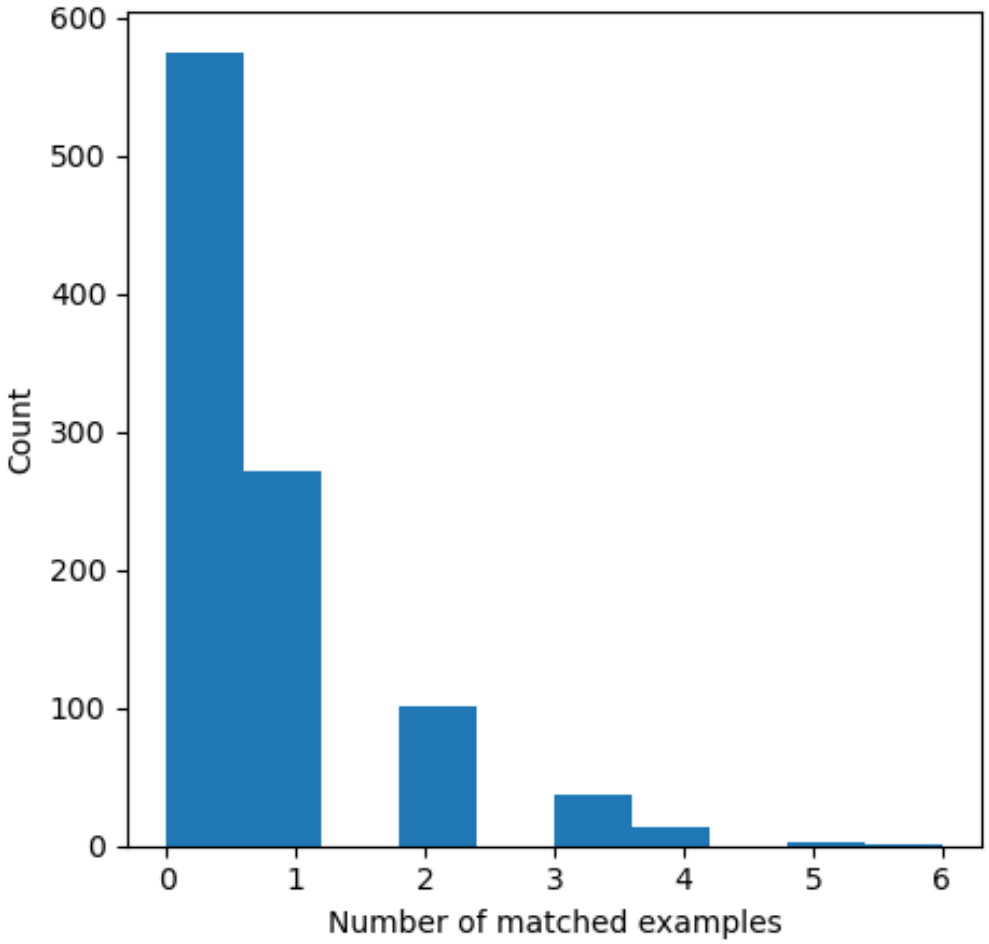

Figure 4.2: How many observations matched with how many out of 50,000 examples

contain all points is a well-known and computationally hard reference class problem. [11]

The adapted framework did not generalise well to unseen data at all. The probability of all 12 variables lying in the same interval as an example is extremely low. A better approach might be to finding the k closest examples, like the k-nearest-neighbour algorithm. However, at that point the assumption that the examples and background knowledge come from the same distribution would be violated. An adjustment like that would create a completely different framework, which has little to do with PAC semantics. For this reason, we decided not to go further with the idea.

With the prediction adaptation we tried to fit a square peg into a round hole. PAC semantics is not meant to predict data from a test set. It is a framework for implicitly learning from examples to answer a query. In our pursuit to find a task that allows us to compare PAC semantics with other machine learning frameworks, we had to change it drastically and then ended up in a dead end.

If there is a take-away from our implementation endeavours, it is that making a hitherto purely theoretical framework viable for practical applications requires a large community effort. You need to program and test it, find real-life applications to use it on,

create datasets with suitable format, make baselines that allow for comparing different methodologies and so on. With this implementation and our use case ideas we hop to have set a first step into a long process before such a framework can become useful in real life.

Chapter 5

Conclusion

We presented an extension to the PAC semantics and SMT framework from [17] that added the ability to handle imprecise data. We formalised this notion by defining a blurring process and adapting existing definitions such as witnessing. Despite the enhanced representation of examples, we have proven that polynomial-time implicit learnability is still preserved in linear real arithmetic. This is a nice result, since we have managed to expand the expressivity of the framework without increasing its running time. However, these results rely on some assumptions about how exactly data is masked and blurred, which have to be satisfied.

To test how well these assumptions hold up for a real dataset, we implemented the PAC semantics framework, which has not been done before. We showed that the validity estimate of queries very quickly becomes inaccurate if random masking is applied. This implies that care has to be taken in ensuring that incomplete data does not affect the entailment of the query. Moreover, the restriction that background knowledge and examples have to come from the same distribution makes it difficult to adapt the framework to unseen or atypical data.

While a lot of work has been done in the field to rigorously define and prove guarantees about various variations of the PAC framework, there is a lack of practical applications available. Without datasets and tasks with baselines, it has proven very difficult to gain value out of an implementation. There are many unique features in the PAC semantics framework, such as implicit learning, the ability to handle partial and now also imprecise data or high explainability. These could prove to be very useful for a variety of applications, but their potential is still largely untapped. With this work, we have both improved the theoretical aspects of the framework, as well as provided starting points for practical applications. It is important to move a field forward

Bibliography

[1] American Heart Association. Know your target heart rates for exercise, losing weight and health. https://www.heart.org/en/healthy-living/fitness/fitness-basics/target-heart-rates, 2020.

[2] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The satisfiability modulo theories library (smt-lib). www.SMT-LIB.org, 2016.

[3] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. *Satisfiability modulo theories*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 1 edition, 2009.

[4] Vaishak Belle and Brendan Juba. Implicitly learning to reason in first-order logic. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3381–3391. Curran Associates, Inc., 2019.

[5] Mayo Clinic. Hypoxemia (low blood oxygen). https://www.mayoclinic.org/symptoms/hypoxemia/basics/definition/sym-20050930, 2020.

[6] Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *CoRR*, abs/1810.07896, 2018.

[7] Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: Introduction and applications. *Commun. ACM*, 54(9):69–77, September 2011.

[8] David Gunning and David Aha. Darpa's explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, Jun. 2019.

[9] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[10] Brendan Juba. Learning implicitly in reasoning in pac-semantics. *CoRR*, abs/1209.0056, 2012.

[11] Brendan Juba and Hengxuan Li. More accurate learning of k-dnf reference classes. In *AAAI 2020*, 2020.

[12] Roni Khardon and Dan Roth. Learning to reason. *J. ACM*, 44(5):697–725, September 1997.

[13] W. Kirchgässner, O. Wallscheid, and J. Böcker. Deep residual convolutional and recurrent neural networks for temperature estimation in permanent magnet

synchronous motors. In *2019 IEEE International Electric Machines Drives Conference (IEMDC)*, pages 1439–1446, 2019.

[14] W. Kirchgässner, O. Wallscheid, and J. Böcker. Empirical evaluation of exponentially weighted moving averages for simple linear thermal modeling of permanent magnet synchronous machines. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pages 318–323, 2019.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[16] Loizos Michael. Partial observability and learnability. *Artificial Intelligence*, 174(11):639 – 669, 2010.

[17] Ionela Mocanu, Vaishak Belle, and Brendan Juba. Polynomial-time implicit learnability in smt. In *Proceedings to the 24th European Conference on Artificial Intelligence (ECAI 2020)*, 1 2020.

[18] Alfred Tarski. A decision method for elementary algebra and geometry. *Journal of Symbolic Logic*, 17(3):207–207, 1952.

[19] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[20] Leslie G. Valiant. Robust logics. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 642–651, New York, NY, USA, 1999. ACM.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[22] Jieshu Wang. Symbolism vs. connectionism: A closing gap in artificial intelligence. `http://wangjieshu.com/2017/12/23/symbol-vs-connectionism-a-closing-gap-in-artificial-intelligence/`, 2017.

[23] Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1):3–27, 1988.

Appendix A

Implementation setup

The implementation can be found on
`https://github.com/APRader/pac-smt-arithmetic`
and works under Python 3.8. In order to run it, you have to install the z3-solver package, for example by running `pip install z3-solver`.

The file `pac.py` contains general-use functions for PAC semantics, such as the decision procedure and masking. In the file `motor_util.py` you can find utility functions specifically for the motor temperature dataset. You have to download the dataset yourself (under `https://www.kaggle.com/wkirgsn/electric-motor-temperature`) and save it under the name `pmsm_temperature_data.csv`. Lastly, our masking vs validity experiments can be reproduced using the file `implementation.py`.