A Parameterized Theory of PAC Learning

Cornelius Brand,[1] **Robert Ganian,**[1] **Kirill Simonov,**[2]

[1]Algorithms and Complexity Group, TU Wien, Austria, [2]Chair for Algorithm Engineering, Hasso Plattner Institute, Germany
{cbrand, rganian}@ac.tuwien.ac.at, kirill.simonov@hpi.de

Abstract

Probably Approximately Correct (i.e., PAC) learning is a core concept of sample complexity theory, and efficient PAC learnability is often seen as a natural counterpart to the class P in classical computational complexity. But while the nascent theory of parameterized complexity has allowed us to push beyond the P-NP "dichotomy" in classical computational complexity and identify the exact boundaries of tractability for numerous problems, there is no analogue in the domain of sample complexity that could push beyond efficient PAC learnability.

As our core contribution, we fill this gap by developing a theory of parameterized PAC learning which allows us to shed new light on several recent PAC learning results that incorporated elements of parameterized complexity. Within the theory, we identify not one but two notions of fixed-parameter learnability that both form distinct counterparts to the class FPT—the core concept at the center of the parameterized complexity paradigm—and develop the machinery required to exclude fixed-parameter learnability. We then showcase the applications of this theory to identify refined boundaries of tractability for CNF and DNF learning as well as for a range of learning problems on graphs.

1 Introduction

While a number of different models for sample complexity have by now been considered in the literature, the fundamental concept of Probably Approximately Correct (i.e., PAC) learning (Valiant 1984) remains a core pillar which can theoretically capture and explain the success of learning algorithms in a broad range of contexts. Intuitively, in the classical proper PAC setting we ask whether it is possible to identify (or "learn") a hypothesis from a specified hypothesis space after drawing a certain number of labeled samples[1] according to an unknown distribution. As the best case scenario, one would wish to have an algorithm that can learn a hypothesis with arbitrarily high precision and with arbitrarily high probability after spending at most polynomial time and using at most polynomially-many samples. These are called *efficient* PAC-learning algorithms, and the class of learning problems admitting such algorithms forms a natural counterpart to the class P in the classical complexity theory of computational problems.

In spite of this parallel, our understanding of the sample complexity of learning problems remains significantly behind the vast amounts of knowledge we have by now gathered about the time complexity of computational problems. Indeed, while understanding the distinction between NP-hard and polynomially tractable classes of instances for computational problems remains an important research direction, over the past two decades focus has gradually shifted towards a more fine-grained analysis of the boundaries of tractability for such problems. In particular, *parameterized complexity* (Cygan et al. 2015; Downey and Fellows 2013) has emerged as a prominent paradigm that yields a much deeper understanding of the limits of tractability than classical complexity theory, and today we see research exploring the parameterized complexity of relevant problems appear in a broad range of venues spanning almost all areas of computer science. The main idea behind the parameterized paradigm is to investigate the complexity of problems not only with respect to the input size n, but also based on a numerical parameter k that captures some property of the input; the central notion is then that of *fixed-parameter tractability* (FPT), which means that the problem of interest can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f.

Given its ubiquitous applications, it is natural to ask whether (and how) the principles of parameterized complexity analysis can also be used to expand our understanding of the foundations of sample complexity beyond what may be possible through the lens of efficient PAC-learnability. In fact, there already are a handful of papers (Li and Liang 2018; Arvind, Köbler, and Lindner 2009; Alekhnovich et al. 2008; Gärtner and Garriga 2007) that have hinted at the potential of this novel field by identifying learning algorithms that seem to act as intuitive counterparts to the fixed-parameter algorithms emblematic of parameterized complexity. However, the theoretical foundations of a general parameterized extension to the PAC-learning framework have yet to be formalized and developed, stifling further development of this unique bridge between the two research communities.

Contribution. As our first conceptual contribution, we lay down the foundations of a general parameterized extension to the PAC-learning framework. Crucially, when defining the analogue of fixed-parameter tractability in the sample

[1]Formal definitions are provided in the Preliminaries.

complexity setting, we show that there are two natural definitions which need to be considered: a parameterized learning problem is

1. fpt-*time learnable* if a suitable PAC hypothesis can be learned from polynomially-many samples using a fixed-parameter algorithm, and
2. fpt-*sample learnable* if a suitable PAC hypothesis can be learned using a fixed-parameter algorithm.

Essentially, in the latter case the total number of samples used need not be polynomial, but of course remains upper-bounded by the (fixed-parameter) running time of the algorithm. An important feature of the framework is that parameters are allowed to depend not only on the sought-after concept (as was considered in prior work (Arvind, Köbler, and Lindner 2009)), but also on properties of the distribution. This can be seen as analogy to how one utilizes parameters in the classical parameterized complexity paradigm: they can capture the properties we require on the output (such as the size of a solution), but also restrictions we place on the input (such as structural parameters of an input graph).

The parameterized PAC-learning framework proposed in this paper also includes a machinery that can be used to obtain lower bounds which exclude the existence of FPT-time and fpt-sample learning algorithms; this is done by building a bridge to the well-studied W-hierarchy in parameterized complexity theory. Moreover, we provide the sample-complexity analogues to the complexity class XP; similarly as in the fixed-parameter case, it is necessary to distinguish whether we restrict only the running time or the number of samples.

After laying down the foundations, we turn our attention to two major problem settings with the aim of illustrating how these notions can expand our knowledge of sample complexity. First, we study the boundaries of learnability for the fundamental problem of learning DNF and CNF formulas, which are among the most classical questions in PAC learning theory.

On one hand, k-CNF and k-DNF formulas are known to be efficiently PAC-learnable for every fixed k, albeit the running time of the known algorithms have the form $\mathcal{O}(n^k)$. As our first result, we show that while both of these learning problems are xp-time learnable, under well-established complexity assumptions they are neither fpt-time nor fpt-sample learnable when parameterized by k. On the other hand, for each $p > 1$ it is well-known that p-term DNF and p-clause CNF formulas are not efficiently PAC learnable (Pitt and Valiant 1988; Alekhnovich et al. 2008) unless $\mathsf{P} = \mathsf{NP}$, and here we ask the question whether a suitable parameterization can be used to overcome this barrier to tractability. We show that while learning p-term DNF and p-clause CNF formulas remains intractable under the most natural parameterizations of the target concept, one can learn such formulas by an fpt-time algorithm by exploiting a parameterization of the distribution. In particular, it is not difficult to observe that the learning problem is tractable when each sample contains at most one variable set to `True`, and we show that it is possible to lift this to a non-trivial fpt-time learning algorithm when parameterized by the number of variables which are exempt from this restriction (i.e., can always contain any value).

While CNF and DNF learning is fundamental for PAC learning, in the last part of our study we turn towards a setting which has been extensively studied in both the sample complexity and parameterized complexity paradigms: graph problems. Indeed, fundamental graph problems such as vertex cover and subgraph detection served as a testbed for the development of parameterized complexity theory, and are also heavily studied in machine learning and sample complexity contexts (Nguyen and Maehara 2020; Abasi and Bshouty 2019; Abasi, Bshouty, and Mazzawi 2018; Damaschke and Muhammad 2010; Alon and Asodi 2005; Choi and Kim 2010). Here, we obtain a meta-theorem showing that a large class of graph problems is fpt-time learnable: in particular, we provide an fpt-time learning algorithm for finding a vertex deletion set to $\mathcal{H}$, where $\mathcal{H}$ is an arbitrary graph class that can be characterized by a finite class of forbidden induced subgraphs. As one special case, this captures the problem of learning hidden vertex covers (Damaschke and Muhammad 2010). We conclude by excluding a similar result for classes $\mathcal{H}$ characterized by forbidden minors—hence, for instance hidden feedback vertex sets are neither fpt-time nor fpt-sample learnable when parameterized by their size.

Due to space constraints, proof details are provided in the supplementary material.

2 Preliminaries

As basic notation and terminology, we set $\{0,1\}^* = \bigcup_{m\in\mathbb{N}}\{0,1\}^m$. A *distribution* on $\{0,1\}^n$ is a mapping $\mathcal{D} : \{0,1\}^n \to [0,1]$ such that $\sum_{x\in\{0,1\}^n} \mathcal{D}(x) = 1$, and the *support* of $\mathcal{D}$ is the set $\operatorname{supp}\mathcal{D} = \{x \mid \mathcal{D}(x) > 0\}$. We first recall the basics of both PAC-learning and parameterized complexity theory.

PAC-Learning

Let us begin by formalizing the classical learning theory considered in this article (Valiant 1984; Mohri, Rostamizadeh, and Talwalkar 2012).

Definition 2.1. A *concept* is an arbitrary Boolean function $c : \{0,1\}^n \to \{0,1\}$. An assignment $x \in \{0,1\}^n$ is called a *positive sample* for c if $c(x) = 1$, and a *negative sample* otherwise. A *concept class* $\mathcal{C}$ is a set of concepts. For every $m \in \mathbb{N}$, we write $\mathcal{C}_m = \mathcal{C} \cap \mathcal{B}_m$, where $\mathcal{B}_m$ is the set of all m-ary Boolean functions.

Definition 2.2. Let $\mathcal{C}$ be a concept class. A surjective mapping $\rho : \{0,1\}^* \to \mathcal{C}$ is called a *representation scheme* of $\mathcal{C}$. We call each r with $\rho(r) = c$ a *representation* of concept c.

In plain terms, while concepts may be arbitrary functions, representations are what make these "tangible" and are what we typically expect as the output of learning algorithms.

Definition 2.3. A *learning problem* is a pair $(\mathcal{C}, \rho)$, where $\mathcal{C}$ is a concept class and ρ is a representation scheme for $\mathcal{C}$.

Definition 2.4. A *learning algorithm* for a learning problem $(\mathcal{C}, \rho)$ is a randomized algorithm such that:

1. It obtains the values n, ε, δ as inputs, where n is an integer and $0 < \varepsilon, \delta \leq 1$ are rational numbers.

2. It has access to a hidden representation r^* of some concept $c^* = \rho(r^*)$ and a hidden distribution $\mathcal{D}_n$ on $\{0,1\}^n$ through an oracle that returns *labeled samples* $(x, c^*(x))$, where $x \in \{0,1\}^n$ is drawn at random from $\mathcal{D}_n$.
3. The output of the algorithm is a representation of some concept, called its *hypothesis*.

Remark 2.5. Clearly, the algorithm can infer the value of n from the samples that the oracle returns. Still, n is included in the input for the sake of being explicit. We use $s = |r^*|$ to denote the size of the hidden representation.

Definition 2.6. Let $\mathcal{A}$ be a learning algorithm. Fix a hidden hypothesis c^* and a distribution on $\{0,1\}^n$. Let h be a hypothesis output by $\mathcal{A}$ and $c = \rho(h)$ be the concept h represents. We define

$$\mathrm{err}_h = \mathbb{P}_{x \sim \mathcal{D}_n}(c(x) \neq c^*(x))$$

as the probability of the hypothesis and the hidden concept disagreeing on a sample drawn from $\mathcal{D}_n$, the so-called *generalization error* of h under $\mathcal{D}_n$.

The algorithm $\mathcal{A}$ is called *probably approximately correct (PAC)* if it outputs a hypothesis h such that $\mathrm{err}_h \leq \varepsilon$ with probability at least $1 - \delta$.

Usually, learning problems in this framework are regarded as tractable if they are PAC-learnable within polynomial time bounds. More precisely, we say that a learning problem L is *efficiently* PAC-learnable if there is a PAC algorithm for L that runs in time polynomial in $n, s, 1/\varepsilon$ and $1/\delta$.

Parameterized Complexity

To extend the above concepts from learning theory to the parameterized setting, we now introduce the parts of parameterized complexity theory that will become important later. We follow the excellent exposition of the textbook by Cygan et al. (2015) and define the following central notions:

Definition 2.7. A *parameterized problem* is a language $L \subseteq \{0,1\}^* \times \mathbb{N}$. For a pair $(x,k) \in \{0,1\}^* \times \mathbb{N}$, k is called the *parameter* of the instance (x,k). The encoding length of (x,k) is called the *size* of the instance.

Remark 2.8. The definition of parameterized decision problems can easily be extended to search problems, where we are additionally required to output a witness in case our algorithm outputs "yes."

Definition 2.9. A parameterized problem L is *fixed-parameter tractable* if there exists an algorithm $\mathcal{A}$, a computable non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ and a polynomially bounded non-decreasing function $p(\cdot,\cdot) : \mathbb{N}^2 \to \mathbb{N}$ such that $\mathcal{A}$ correctly decides for an input (x,k) whether or not $(x,k) \in L$ holds in time $f(k) \cdot p(n,k)$. We denote the class of fixed-parameter tractable problems by FPT.

Remark 2.10. While fixed-parameter tractability lies at the heart of parameterized complexity theory, the class XP defined below captures a weaker notion of tractability that is still desirable for parameterized problems which are not believed to be in FPT.

Definition 2.11. A parameterized problem L is in the complexity class XP if there exists an algorithm $\mathcal{A}$, a computable non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ and a polynomially bounded non-decreasing function $p(\cdot,\cdot) : \mathbb{N}^2 \to \mathbb{N}$ such that $\mathcal{A}$ correctly decides for an input (x,k) whether or not $(x,k) \in L$ holds in time $p(n,k)^{f(k)}$.

Remark 2.12. Less formally, XP is the class of problems that are solvable in polynomial time for a constant parameter value. The difference to FPT is that for XP, we allow this polynomial (and in particular, its degree) to depend on k, while it is fixed for all k in the definition of FPT.

We also assume basic familiarity with the complexity classes $\mathsf{W}[1]$ and $\mathsf{W}[2]$ (Downey and Fellows 2013). It is a well-established conjecture that these are strictly larger than FPT, and hence establishing $\mathsf{W}[1]$- or $\mathsf{W}[2]$-hardness for a problem essentially rules out its fixed-parameter tractability.

3 Parameterized PAC-Learning

As the first step towards defining a theory of parameterized PAC learning, we need to consider the parameters that will be used. In the computational setting we associated a parameter with each instance, but in the learning setting this notion does not exist—instead, a learning algorithm needs to deal with a hidden concept representation and a hidden distribution, and we allow both of these to be tied to parameterizations.

Definition 3.1 (Parameterization of Representations)**.** Let $\{\mathcal{R}_k\}_{k \in \mathbb{N}}$ with $\mathcal{R}_k \subseteq \{0,1\}^*$ be a mapping assigning a set of representations to every natural number k such that for every $r \in \{0,1\}^*$, there is some k such that $r \in \mathcal{R}_k$, that is, $\bigcup_k \mathcal{R}_k = \{0,1\}^*$. We call $\{\mathcal{R}_k\}_{k \in \mathbb{N}}$ a *parameterization of representations*. Given a parameterization of representations $\{\mathcal{R}_k\}_{k \in \mathbb{N}}$, we associate a value $\kappa_{\mathcal{R}}(r)$ to single representations $r \in \{0,1\}^*$ by defining

$$\kappa_{\mathcal{R}}(r) = \min\{k \,:\, r \in \mathcal{R}_k\}.$$

Remark 3.2. In line with the usual notion of parameterizations, we will assume $\kappa_{\mathcal{R}}$ to be a computable function.

Example. Let ρ be the representation scheme taking a (binary representation of a) k-term DNF formula to its underlying Boolean function. If we are interested in learning k-term DNFs, we probably want to consider a k'-term DNF a k-term DNF for $k' \leq k$, too. Therefore, we let $\mathcal{R}_k$ be the set of all k'-term DNFs for $k' \leq k$. The associated mapping $\kappa_{\mathcal{R}}$ then maps every k-term DNF to the value of k.

Definition 3.3 (Parameterization of Distributions and Samples)**.** Let λ be a mapping assigning a natural number to every distribution on $\{0,1\}^n$ for each n, such that for every two distributions $\mathcal{D}, \mathcal{D}'$ on $\{0,1\}^n$, if $\operatorname{supp}\mathcal{D} \subseteq \operatorname{supp}\mathcal{D}'$, then $\lambda(\mathcal{D}) \leq \lambda(\mathcal{D}')$. In this case, we call λ a *parameterization of distributions*.

We extend every parameterization of distributions λ to subsets $X \subseteq \{0,1\}^n$ by defining its corresponding *parameterization of sample sets* via:

$$\lambda(X) = \min_{\mathcal{D} \,:\, X = \operatorname{supp}\mathcal{D}} \lambda(\mathcal{D}). \tag{3.1}$$

Remark 3.4. Note that the definition implies that, equivalently, $\lambda(X) = \min_{\mathcal{D} \,:\, X \subseteq \operatorname{supp}\mathcal{D}} \lambda(\mathcal{D})$; in other words, $\lambda(X)$ is the lowest parameter that can be obtained from a distribution that has X as its support.

All parameterizations considered in our exposition depend solely on the support; however, we do want to explicitly also allow more expressive parameterizations that depend on the distribution. To build a theory allowing such parameterizations, it is necessary to impose an additional technical condition which ensures that the distributions minimizing the parameter values for λ are "well-behaved".

Definition 3.5. We say that a distribution $\mathcal{D}^*$ is called *typical for X under λ* if $\mathcal{D}^*$ attains the minimum in Eq. (3.1), that is, $\lambda(X) = \lambda(\mathcal{D}^*)$ and $\operatorname{supp} \mathcal{D}^* = X$.

Definition 3.6 (L-Sampleable Parameterizations)**.** Let $\mathcal{A}$ be a randomized algorithm that receives input $X \subseteq \{0,1\}^n$ and outputs a concept c in time $L(n, |X|, \lambda(X))$ for some non-decreasing function $L(\cdot,\cdot,\cdot)$. We say that λ is *L-sampleable* if the following holds true for the random variable C that corresponds to the output of $\mathcal{A}$ over all random bits used by $\mathcal{A}$ to output c: There is some distribution $\mathcal{D}^*$ that is typical for X under λ such that $C \sim \mathcal{D}^*$ holds, i.e., C has the same distribution as $\mathcal{D}^*$.

Remark 3.7. Every parameterization λ that depends only on the support X is linear-time-sampleable, since the uniform distribution on X will be typical for X under λ. More generally, we believe that every "reasonable" parameterization is polynomial-time-sampleable.

Example. For instance, a valid choice of λ is the mapping that selects the largest number k such that there is a concept x in the support of $\mathcal{D}$ that has k non-zero entries. This parameter can obviously be computed on sets of concepts, as demanded by the condition, and is linear-time sampleable (in the size of the input, $n \cdot t$).

Definition 3.8 (Parameterized Learning Problems)**.** A *parameterized learning problem* is a learning problem $(\mathcal{C}, \rho)$ together with a pair $(\{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$, called its *parameters*, where $\{\mathcal{R}_k\}_{k\in\mathbb{N}}$ is a parameterization of representations and λ is a parameterization of distributions.

At this point, a note on the relation between the definitions presented above and the formalism developed by Arvind, Köbler, and Lindner (2009) is in order. While the latter is capable of expressing learning problems such as k-juntas, where k is the parameter, it cannot account for properties of the sample space. For example, one might be interested in learning Boolean formulas from samples that have a limited number of k variables set to true. Our framework captures this by including the distribution into the parameterization. This can be likened on a conceptual level to the difference that exists in ordinary parameterized complexity theory between parameterizations by solution properties versus parameterizing by properties of the input instance.

Definition 3.9 (Parameterized Learning Algorithm)**.** A *parameterized learning algorithm* for a parameterized learning problem $(\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$ is a learning algorithm for $(\mathcal{C}, \rho)$ in the sense of Definition 2.4. In addition to n, ε, δ, a parameterized learning algorithm obtains two inputs k and ℓ, which are promised to satisfy $k = \kappa_{\mathcal{R}}(r^*)$ as well as $\ell = \lambda(\mathcal{D}_n)$, and the algorithm is required to always output a hypothesis h satisfying $\kappa(h) \in \mathcal{R}_k$.

Remark 3.10. By requiring the hidden hypothesis and the output hypothesis to adhere to the same representation scheme, we limit ourselves to the setting of *proper* learning. In principle, nothing speaks against extending our framework also to the improper case, as is done in Arvind, Köbler, and Lindner (2009) for their formalization. Since all our examples speak about proper learning, we restrict our definitions to this case.

Remark 3.11. As readers acquainted with parameterized complexity theory may find noteworthy, parameterized learning problems as defined here depend on two parameters, as opposed to a single parameter. For parameterized algorithms, it is customary to combine multiple parameters k, ℓ into one via defining a new parameter such as $k + \ell$ or $\max\{k, \ell\}$. Indeed, this leads to less heavy notation, while sacrificing nothing in terms of expressive power of the obtained theory.

We shall see towards the end of this section that for the purposes of this article, it is more convenient to make do with two separate parameters, in order to establish a meaningful link between ordinary parameterized algorithms and parameterized learning algorithms.

Let $\operatorname{poly}(\cdot)$ denote the set of functions that can be bounded by non-decreasing polynomial functions in their arguments. Furthermore, $\operatorname{fpt}(x_1, \ldots, x_t; k_1, \ldots, k_t)$ and $\operatorname{xp}(x_1, \ldots, x_t; k_1, \ldots, k_t)$ denote those functions bounded by $f(k_1, \ldots, k_t) \cdot p(x_1, \ldots, x_t)$ and $p(x_1, \ldots, x_t)^{f(k_1,\ldots,k_t)}$, respectively, for any non-decreasing computable function f in $k_1, \ldots, k_t$ and $p \in \operatorname{poly}(x_1, \ldots, x_t)$.

Definition 3.12 ((T,S)-PAC Learnability)**.** Let $T(n, s, 1/\varepsilon, 1/\delta, k, \ell), S(n, s, 1/\varepsilon, 1/\delta, k, \ell)$ be any two functions taking on integer values, and non-decreasing in all of their arguments.

A parameterized learning problem $\mathcal{L} = (\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$ is *(T,S)-PAC learnable* if there is a PAC learning algorithm for $\mathcal{L}$ that runs in time $\mathcal{O}(T(n, s, 1/\varepsilon, 1/\delta, k, \ell))$ and queries the oracle at most $\mathcal{O}(S(n, s, 1/\varepsilon, 1/\delta, k, \ell))$ times.

We denote the set of parameterized learning problems that are (T,S)-PAC learnable by $\mathsf{PAC}[T,S]$. This is extended to sets of functions $\mathbf{S}, \mathbf{T}$ through setting $\mathsf{PAC}[T,S] = \bigcup_{S\in\mathbf{S}, T\in\mathbf{T}} \mathsf{PAC}[T,S]$.

Definition 3.13. We define the following complexity classes:

$$\mathsf{FPT\text{-}PAC}_{\mathsf{time}} = \mathsf{PAC}[\operatorname{fpt}, \operatorname{poly}], \tag{3.2}$$

$$\mathsf{FPT\text{-}PAC} = \mathsf{PAC}[\operatorname{fpt}, \operatorname{fpt}], \tag{3.3}$$

$$\mathsf{XP\text{-}PAC}_{\mathsf{time}} = \mathsf{PAC}[\operatorname{xp}, \operatorname{poly}], \tag{3.4}$$

$$\mathsf{XP\text{-}PAC} = \mathsf{PAC}[\operatorname{xp}, \operatorname{xp}], \tag{3.5}$$

where we fixed

$$\begin{aligned} \operatorname{poly} &= \operatorname{poly}(n, s, 1/\varepsilon, 1/\delta, k, \ell), \\ \operatorname{fpt} &= \operatorname{fpt}(n, s, 1/\varepsilon, 1/\delta; k, \ell), \\ \operatorname{xp} &= \operatorname{xp}(n, s, 1/\varepsilon, 1/\delta; k, \ell). \end{aligned}$$

Remark 3.14. In addition to the complexity classes just defined, there is a fifth class that may be considered here: $\mathsf{PAC}[\operatorname{xp}, \operatorname{fpt}]$. However, this class does not play a role in any of the examples presented in this work, and hence we leave its exploration to future works.

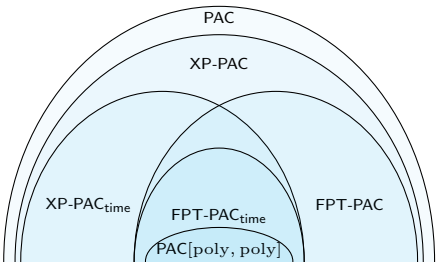

Figure 1: A schematic view of the parameterized learning classes defined in Definition 3.13.

Figure 1 provides an overview of these complexity classes and their relationships. As does XP, the class XP-PAC in the learning setting contains precisely those parameterized learning problems which become efficiently PAC-learnable whenever the parameters are fixed to an arbitrary constant.

Example. A problem that fits into the class XP-PAC is learning of k-CNF and k-DNF formulas, parameterized by k. On the other hand, the upcoming Theorems 5.6 and 6.2 furnish the class FPT-PAC$_{\text{time}}$. An example for the class FPT-PAC in spirit can be found in Li and Liang (2018). Finally, an example of a learning problem in XP-PAC$_{\text{time}}$ is provided in Observation 6.5.

Consistency Checking: A Link between Theories

With the basic complexity classes in place, we turn to establishing links between the newly developed theory and the by now well-established parameterized complexity paradigm for decision problems. Crucially, these links will allow us to exploit algorithmic upper and lower bounds from the latter also in the learning setting.

Definition 3.15 (Parameterized Consistency Checking)**.** With every parameterized learning problem $\mathcal{L} = (\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$ and every function $f(n,t,k,\ell)$ we associate a parameterized search problem f-CONSISTENCY($\mathcal{L}$) as follows:

1. Its input is a list of labeled samples $((x_1,a_1),\ldots,(x_t,a_t))$, with $x_i \in \{0,1\}^n$ for all i and x_i pairwise distinct, as well as $a_i \in \{0,1\}$.
2. Its parameters are $k \in \mathbb{N}$, given as part of the input, and $\ell = \lambda(\{x_1,\ldots,x_t\})$.
3. The task is to decide whether there is a representation r that has $\kappa(r) \in \mathcal{R}_k$ with $|r| \leq f(n,t,k,\ell)$ and for the concept $c = \rho(r)$, it holds that $c(x_i) = a_i$ for all i.

We call this parameterized problem the *parameterized consistency checking problem* associated with $\mathcal{L}$ and f.

Ignoring all parameters in Definition 3.15 gives the usual notion of consistency checking problems (Pitt and Valiant 1988). The function f is required to provide an explicit upper-bound on the sought-after representation; indeed, while each learning instance comes with a hidden representation of a certain size, this is not the case with the consistency checking problem and we need to allow the running time bounds to take the size of the representation into consideration.

It is well-known that, under the assumption that the hypothesis space is not too large, there is an equivalence between a learning problem being PAC-learnable and the corresponding decision problem being solvable in randomized polynomial time. We now observe that a similar equivalence holds also in the parameterized setting.

Lemma 3.16. *Let $\mathcal{L} = (\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$ be a parameterized learning problem, and let $f(n,t,k,\ell)$ be a function. If $\mathcal{L}$ is (T,S)-PAC learnable and λ is L-sampleable, then there exists a randomized algorithm that, upon input labeled samples $\{(x_1,a_1),\ldots,(x_t,a_t)\}$ and parameters k,ℓ with $\lambda(\{x_1,\ldots,x_t\}) = \ell$, returns a consistent hypothesis $h \in \mathcal{R}_k$ of size $|h| \leq s = f(n,t,k,\ell)$ with probability at least $1-\delta$ if it exists, and runs in time $\mathcal{O}(T(n,s,t,\delta,k,\ell) + L(n,t,\ell)\cdot S(n,s,t,\delta,k,\ell))$.*

Theorem 3.17. *Let $\mathcal{L} = (\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$ be a parameterized learning problem, and let $f(n,t,k,\ell)$ be a function.*

If $\mathcal{L}$ is in FPT-PAC*, $f \in \mathrm{fpt}(n,t;k,\ell)$ and λ is S-sampleable for some $S \in \mathrm{fpt}(n,t;\ell)$, then the parameterized consistency checking problem f-CONSISTENCY($\mathcal{L}$) associated with $\mathcal{L}$ and f is in* FPT*.*

Similarly, if $\mathcal{L}$ is in XP-PAC*, $f \in \mathrm{xp}(n,t;k,\ell)$ and λ is S-sampleable for some $S \in \mathrm{xp}(n,t;\ell)$, then the parameterized consistency checking problem f-CONSISTENCY($\mathcal{L}$) associated with $\mathcal{L}$ and f is in* XP*.*

Lemma 3.18. *Let $\mathcal{L} = (\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$ be a parameterized learning problem, and let $\mathcal{H}_{n,k} = \mathcal{R}_k \cap \rho^{-1}(\mathcal{C}_n)$ be the set of all representations under ρ in $\mathcal{R}_k$ of concepts in $\mathcal{C}_n$. Suppose there is a deterministic algorithm running in time $T(n,t,\delta,k,\ell)$ for the consistency checking problem $\log|\mathcal{H}_{n,k}|$-CONSISTENCY($\mathcal{L}$). Then, $\mathcal{L}$ is (T',S)-PAC learnable, where*

$$S(n,s,1/\varepsilon,1/\delta,k,\ell) = \frac{1}{\varepsilon}(\log|\mathcal{H}_{n,k}| + \frac{1}{\delta})),$$

$$T'(n,s,1/\varepsilon,1/\delta,k,\ell) = T(n,s,\frac{1}{\varepsilon}(\log|\mathcal{H}_{n,k}| + \frac{1}{\delta}),k,\ell).$$

Theorem 3.19. *Let $\mathcal{L} = (\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$, and denote the set of representations of $\mathcal{C}_n$ in $\mathcal{R}_k$ under ρ as $\mathcal{H}_{n,k}$.*

If the parameterized consistency checking problem $\log|\mathcal{H}_{n,k}|$-CONSISTENCY($\mathcal{L}$) is in FPT *and $\log|\mathcal{H}_{n,k}| \in \mathrm{fpt}(n;k)$, then $\mathcal{L}$ is in* FPT-PAC$_{\text{time}}$*.*

Similarly, if the parameterized consistency checking problem $\log|\mathcal{H}_{n,k}|$-CONSISTENCY($\mathcal{L}$) is in XP *and $\log|\mathcal{H}_{n,k}| \in \mathrm{xp}(n;k)$, then $\mathcal{L}$ is in* XP-PAC$_{\text{time}}$*.*

The significance of the previous theorems lies in transferring parameterized algorithmic upper and lower bounds for consistency checking into upper and lower bounds for parameterized learning problems, respectively.

To wit, Theorem 3.17 allows us to conclude from the fact that a parameterized consistency checking problem is efficiently solvable by a parameterized algorithm that also the parameterized learning problem it belongs to is efficiently solvable. This will be exploited in Theorems 5.6, 6.2.

On the other hand, Theorem 3.19 tells us that an efficient algorithm for a parameterized learning problem implies an

efficient algorithm for the corresponding paramerized consistency checking problem. Turning this around, we see that lower bounds on consistency checking imply lower bounds for learning. This will be exploited in Theorems 5.1, 6.4.

4 Previous Work on Parameterized Learning

After having introduced this new framework, we dedicate a separate section to how it fits in with the few papers that have considered parameterized approaches to learning theory, and how it differs from them.

Proceeding chronologically, the earliest link between parameterized complexity theory and learning theory was established already by the pioneers of paramerized complexity (Downey, Evans, and Fellows 1993). While they also establish a parameterized learning model and link learning complexity with well-studied graph problems, in contrast to our work, they study exact learning by extended equivalence queries. Gärtner and Garriga (2007) study the complexity of learning directed cuts in various learning models, including PAC learning. While their respective result falls under the standard poly-time PAC learning, it is notable that the improved learning bound they show is in terms of a structural parameter of the graph. Better-than-trivial exponential running-time bounds for learning k-term DNF formulas have been studied, e.g., $n^{\tilde{\mathcal{O}}(\sqrt{n \log k})}$ (Alekhnovich et al. 2008). We already compared our framework to the earlier work by Arvind, Köbler, and Lindner (2009) in Section 3, which only identified the class FPT-PAC.

The more recent work of Li and Liang (2018) on parameterized algorithms for learning mixtures of linear regressions is not, strictly speaking, within our framework of parameterized PAC learning, since their error parameter is the distance to the hidden concept rather than the probability of mislabelling under any distribution. However, their sample complexity bound is FPT, turning into infinite when the parameters are not bounded, and it is also a notable example of a PAC-learning result that holds only for a certain family of distributions (an approach which is also covered by our framework; see the intuition behind Definition 3.6). Finally, van Bergerem, Grohe, and Ritzert (2022) consider parameterized complexity of learning first-order logic; they show hardness results via parameterized reductions as well as FPT-time learning via consistency checking, which aligns well with our framework.

5 Parameterized DNF and CNF Learning

For the most of this section, we consider the problem of learning a hidden DNF formula under various parameterizations, which are formally defined throughout the section. We first observe that the problem of learning a CNF formula is equivalent to learning a DNF formula under any parameterization that is preserved under negation of the formula: indeed, by negating the input to a DNF-learning algorithm and then negating the output formula we obtain a CNF-learning algorithm, and vice versa. Thus our results hold for learning CNF formulas as well, which we do not show explicitly below.

Let LEARNING DNF be the learning problem $(\mathcal{C}, \rho)$ where $\mathcal{C}$ is the set of boolean functions corresponding to formulas in disjunctive normal form and ρ is the straightforward representation of the terms of the formula. To define a fundamental variant of the problem where the number of terms in the target formula is bounded, let LEARNING k-TERM DNF be the parameterized learning problem $(\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$ where $\mathcal{C}$ and ρ are as above, κ maps the representation of the formula to the number of terms in it, and λ is any constant parameterization. Alas, it is well-known that the consistency checking problem for learning even 2-term DNF is NP-hard (Alekhnovich et al. 2008), and thus by a non-parameterized variant of Theorem 3.17, LEARNING k-TERM DNF is not in XP-PAC$_{\mathsf{time}}$ unless P = NP. Thus, in this parameterization one should not expect any tractability results.

Another natural way to parameterize LEARNING DNF would be to bound the maximum length of a term. Specifically, let LEARNING k-DNF be the parameterized learning problem $(\mathcal{C}, \rho, \{\mathcal{R}_k\}_{k\in\mathbb{N}}, \lambda)$ where $\mathcal{C}$ and ρ are as in LEARNING DNF, $\kappa = \kappa_{\mathcal{R}}$ maps the representation of the formula to the maximum length of a term in it, and λ is trivial. It is well-known that, for every fixed k, LEARNING k-DNF is efficiently PAC-learnable by a brute-force argument and hence the problem is immediately in XP-PAC$_{\mathsf{time}}$ parameterized by k (Mohri, Rostamizadeh, and Talwalkar 2012, Example 2.9). On the other hand, we show that under standard parameterized complexity assumptions this learning result is tight, extending the hardness reduction of Arvind et al. (Arvind, Köbler, and Lindner 2009) for k-juntas and –monomials.

Theorem 5.1. *Assuming* W[2] ≠ FPT, LEARNING k-DNF *and* LEARNING k-CNF *are not in* FPT-PAC$_{\mathsf{time}}$.

The above covers the two most natural parameterizations of the hypothesis class — the number of terms and the maximum size of a term in the target formula — however, we did not yet touch parameterizations of the sample space. As a simple step, observe that solving the consistency problem is easy when each assignment assigns at most one variable to "true"; a one-term DNF can always be produced in this case. On the other hand, the previously-known coloring reduction (Pitt and Valiant 1988) shows that even when each assignment has at most two variables assigned to "true", the problem becomes NP-hard. In the following, we analyze the tractability of learning k-term DNF and k-clause CNF "between" these two cases, and show the following positive result based on parameterizing by a notion of "backdoor to triviality" (Gaspers and Szeider 2012; Semenov et al. 2018).

Theorem 5.2. LEARNING k-CLAUSE CNF *and* LEARNING k-TERM DNF *are both* FPT-PAC$_{\mathsf{time}}$ *parameterized by* $k+s$*, where* s *is the minimum size of a set* S *of variables such that the support of the distribution satisfies the following:*

- *each assignment assigns at most one variable outside of* S *to "false" (respectively "true"), and*
- *each of the remaining variables, at most one assignment assigns it to "false" (respectively "true").*

Proof Sketch. By Theorem 3.19, it is enough to provide a fixed-parameter algorithm for the consistency checking problem with $f = \mathrm{fpt}(n; k + s)$; in particular, it suffices to argue that we can bound the length of all possible hypotheses by such an f. The main technique used in the proof is that of

kernelization (Cygan et al. 2015), i.e., preprocessing of the instance of the consistency checking problem.

We begin by partitioning the set of labelled samples into equivalence classes, where two samples are equivalent if and only if they agree on S. By a series of claims, one can show that if an equivalence class contains more than $k+2$ samples, one can be safely deleted without changing the instance. □

6 Learning on Graphs

Let $H_1, \ldots, H_p$ be a fixed family of graphs. Let $\mathcal{H}$ be a class of graphs that do not contain any of $H_1, \ldots, H_p$ as an induced subgraph. In a classical $\mathcal{H}$-VERTEX DELETION problem the task is, given a graph G and a parameter k, to determine whether there exist a subset of vertices $S \subset V(G)$ of size at most k such that $G - S$ belongs to $\mathcal{H}$. Produced in the standard fashion, the consistency version of this problem receives as input a sequence of graphs $G_1, \ldots, G_t$ over the same vertex set V, together with the sequence of labels $\lambda_1, \ldots, \lambda_t$, and the task is to find a subset $S \subset V$ of size at most k such that $G_i - S \in \mathcal{H}$ if and only if $\lambda_i = 1$. Let us call the respective learning problem LEARNING $\mathcal{H}$-DELETION SET.

In particular, when the forbidden family consists of a single graph K_2, $\mathcal{H}$ is a class of empty graphs, and $\mathcal{H}$-VERTEX DELETION is equivalent to VERTEX COVER. If the family is P_3, the problem becomes CLUSTER VERTEX DELETION — find a subset of vertices to delete so that the graph turns into a cluster graph, i.e., a disjoint union of cliques. LEARNING $\mathcal{H}$-DELETION SET thus generalizes both LEARNING VERTEX COVER and LEARNING CLUSTER DELETION SET, where the task is to learn a hidden vertex cover and deletion set to a cluster graph, respectively.

Let us tie this explicitly to the formal framework of parameterized learning developed above. In this context, we interpret an element $x \in \{0,1\}^n$ as the adjacency matrix of a graph on N vertices, where $n = N^2$. A concept $c : \{0,1\}^n \to \{0,1\}$ is represented by a subset S of the N vertices. The value $c(x)$ indicates whether or not S is an $\mathcal{H}$-Deletion set for the graph with adjacency matrix x. As noted, the representation scheme of c takes the subset S to c as just described. Hence, $\mathcal{C}_n$ corresponds to all $\mathcal{H}$-Deletion sets on N-vertex graphs. We parameterize by taking $\mathcal{R}_k$ to be the set of all vertex subsets of size at most k, and let the distributions carry the trivial constant parameterization $\lambda(\mathcal{D}) = 1$.

Note that, for graph problems, the length of hypotheses is always naturally bounded linearly in n, so we will omit explicit references to f for consistency checking.

Next, we show that the well-known fpt-time algorithm (Cygan et al. 2015) for $\mathcal{H}$-VERTEX DELETION extends to the respective consistency problem, implying that the wide class of problems characterized by finite family of forbidden induced subgraphs is fpt-time learnable.

Lemma 6.1. *Let $\mathcal{H}$ be a class of graphs forbidding induced $H_1, \ldots, H_p$. Then, the parameterized consistency checking problem associated with* LEARNING $\mathcal{H}$-DELETION SET *is fixed-parameter tractable.*

Theorem 6.2. *Let $\mathcal{H}$ be a class of graphs forbidding induced $H_1, \ldots, H_p$.* LEARNING $\mathcal{H}$-DELETION SET *is* fpt*-time PAC-learnable parameterized by the size of the deletion set.*

In contrast, we now show that another well-studied family of deletion problems does most likely not admit a positive result similar to the above. Namely, for a graph H, let $\mathcal{H}$ be a class of graphs that are H-minor-free. In particular, when $H = K_3$, $\mathcal{H}$-VERTEX DELETION is equivalent to FEEDBACK VERTEX SET. This problem is well-known to admit fpt-time algorithms when parameterized by the size of the vertex set to delete (Cygan et al. 2015), however we show that the consistency checking problem associated with LEARNING FEEDBACK VERTEX SET is W[2]-hard. That is, unless the two parameterized complexity classes FPT and W[2] coincide, the consistency problem is not in FPT. The consequence FPT = W[2] is a parameterized analogue to P = NP, and considered highly unlikely.

Lemma 6.3. *The consistency checking problem for* LEARNING FEEDBACK VERTEX SET *is* W[2]*-hard, even with only yes-instances in the input.*

Theorem 6.4. *Unless* FPT = W[2]*, the learning problem* LEARNING FEEDBACK VERTEX SET *is not in* FPT-PAC*.*

On the other hand, the consistency checking problem for LEARNING FEEDBACK VERTEX SET is trivially in XP. Since $\log |\mathcal{H}_{n,k}|$ is trivially upper-bounded by $\mathcal{O}(n)$, by Theorem 3.19 we immediately obtain:

Observation 6.5. LEARNING FEEDBACK VERTEX SET *is in* XP-PAC$_{\mathsf{time}}$.

The same argument can be made for learning many other graph structures, such as dominating and independent sets.

7 Conclusion

Over the last two decades, the parameterized complexity paradigm has arguably revolutionized our understanding of computational problems throughout computer science. We firmly believe that applying a more fine-grained, parameterized lens on learning problems can similarly reveal a wealth of research questions and potential breakthroughs that have remained hidden up to now. This article provides researchers with the tools and concepts they need to begin a rigorous exploration of this novel research direction, while also laying the foundations of a bridge that will connect the research communities surrounding learning theory and time complexity. It is perhaps worth noting that these communities have so far remained fairly isolated from each other, with their few interactions occurring at venues dedicated to AI research.

Unlike in typical complexity-theoretic papers, we view the greatest contribution of this article to be the theory-building part, i.e., Section 3. Indeed, while the specific algorithms and lower bounds that we use to showcase the theory are non-trivial and interesting in their own right, ensuring that the theoretical foundations fit together, are broad enough to capture potential future parameterizations, but also can support the crucial link to the parameterized consistency checking problem was a highly challenging task.

The introduced parameterized PAC learning framework opens many avenues for future study. For instance, there are interesting and non-trivial examples of parameterized learning problems in the class PAC[xp, fpt]?

Acknowledgements

The authors acknowledge support from the Austrian Science Foundation (FWF, project Y 1329 START-Programm).

References

Abasi, H.; and Bshouty, N. H. 2019. On Learning Graphs with Edge-Detecting Queries. In Garivier, A.; and Kale, S., eds., *Algorithmic Learning Theory, ALT 2019, 22-24 March 2019, Chicago, Illinois, USA*, volume 98 of *Proceedings of Machine Learning Research*, 3–30. PMLR.

Abasi, H.; Bshouty, N. H.; and Mazzawi, H. 2018. Non-adaptive learning of a hidden hypergraph. *Theor. Comput. Sci.*, 716: 15–27.

Alekhnovich, M.; Braverman, M.; Feldman, V.; Klivans, A. R.; and Pitassi, T. 2008. The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.*, 74(1): 16–34.

Alon, N.; and Asodi, V. 2005. Learning a Hidden Subgraph. *SIAM J. Discret. Math.*, 18(4): 697–712.

Arvind, V.; Köbler, J.; and Lindner, W. 2009. Parameterized learnability of juntas. *Theor. Comput. Sci.*, 410(47-49): 4928–4936.

Choi, S.; and Kim, J. H. 2010. Optimal query complexity bounds for finding graphs. *Artif. Intell.*, 174(9-10): 551–569.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer. ISBN 978-3-319-21274-6.

Damaschke, P.; and Muhammad, A. S. 2010. Competitive Group Testing and Learning Hidden Vertex Covers with Minimum Adaptivity. *Discret. Math. Algorithms Appl.*, 2(3): 291–312.

Downey, R. G.; Evans, P. A.; and Fellows, M. R. 1993. Parameterized Learning Complexity. In Pitt, L., ed., *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993*, 51–57. ACM.

Downey, R. G.; and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. Springer. ISBN 978-1-4612-6798-0.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer. ISBN 978-1-4471-5558-4.

Gärtner, T.; and Garriga, G. C. 2007. The Cost of Learning Directed Cuts. In Kok, J. N.; Koronacki, J.; de Mántaras, R. L.; Matwin, S.; Mladenic, D.; and Skowron, A., eds., *Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007, Proceedings*, volume 4701 of *Lecture Notes in Computer Science*, 152–163. Springer.

Gaspers, S.; and Szeider, S. 2012. Backdoors to Satisfaction. In Bodlaender, H. L.; Downey, R.; Fomin, F. V.; and Marx, D., eds., *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, 287–317. Springer.

Li, Y.; and Liang, Y. 2018. Learning Mixtures of Linear Regressions with Nearly Optimal Complexity. In Bubeck, S.; Perchet, V.; and Rigollet, P., eds., *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, volume 75 of *Proceedings of Machine Learning Research*, 1125–1144. PMLR.

Mohri, M.; Rostamizadeh, A.; and Talwalkar, A. 2012. *Foundations of Machine Learning*. Adaptive computation and machine learning. MIT Press. ISBN 978-0-262-01825-8.

Nguyen, H.; and Maehara, T. 2020. Graph Homomorphism Convolution. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, 7306–7316. PMLR.

Pitt, L.; and Valiant, L. G. 1988. Computational limitations on learning from examples. *J. ACM*, 35(4): 965–984.

Semenov, A. A.; Zaikin, O.; Otpuschennikov, I. V.; Kochemazov, S.; and Ignatiev, A. 2018. On Cryptographic Attacks Using Backdoors for SAT. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 6641–6648. AAAI Press.

Valiant, L. G. 1984. A Theory of the Learnable. *Commun. ACM*, 27(11): 1134–1142.

van Bergerem, S.; Grohe, M.; and Ritzert, M. 2022. On the Parameterized Complexity of Learning First-Order Logic. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '22, 337–346. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392600.