UNIVERSITÄT ZU LÜBECK

From the Institute for Theoretical Computer Science
of the University of Lübeck
Director: Prof. Dr. Rüdiger Reischuk

# Algorithmic Learning of DNF Formulas, Finite Automata, and Distributions

**Dissertation**

for Fulfillment of Requirements

for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences

Submitted by

**Matthias Lutter**
from Winsen (Luhe)

Lübeck, June 18, 2019

| | |
|---|---|
| First referee: | Prof. Dr. Rüdiger Reischuk |
| Second referee: | Prof. Dr. Hans Simon |
| Date of oral examination: | October 25, 2019 |
| Approved for printing. | Lübeck, October 28, 2019 |

# Abstract

In this thesis, two open problems regarding learning DNF formulas and regular languages, respectively, are solved. Further, new results concerning the identification of areas exceeding a specific density in probability distributions, called density levels, is presented.

First, efficient and proper learning of DNF formulas is studied. For $k$-term DNF formulas, a learning algorithm that learns from positive examples alone is presented, where the examples are drawn at random according to an unknown $q$-bounded distribution, a generalization of the uniform distribution. This solves an open problem addressed by Pitt and Valiant [J. ACM, 1988]. Afterwards, a negative result concerning learnability of poly($d$)-term DNF is presented.

Second, learning regular languages represented by a specific type of finite automata, namely residual alternating automata, is considered. An alternating automaton can provide a very succinct representation of a regular language, while residuality adds a natural meaning to the automaton's states that may simplify its later analysis. It is shown that the $\texttt{AL}^\star$ algorithm designed by Angluin, Eisenstat, and Fisman [Proc. IJCAI, 2015] sometimes returns non-residual alternating automata (AFAs), which disproves a conjecture postulated by the authors. This issue is fixed by the design of a a new algorithm $\texttt{AL}^{\star\star}$ that always returns residual AFAs. This algorithm fulfills the same asymptotic complexity bounds as $\texttt{AL}^\star$. Afterwards, the succinctness of different finite automata is investigated.

Last but not least, the estimation of density levels via algorithmic learning is investigated within the framework of Ben-David and Lindenbaum [JCSS, 1997] – a framework that transforms the spirit of PAC learning to density levels. At first, the relation between learning density levels and the estimation of the entire probability distribution function is clarified. Then it is shown that algorithms that maximize the empirical excess mass can be applied directly to learn density levels in this framework. Until now, it seems that this framework has not been helpful to design learning algorithms. Empirical excess mass maximization has been introduced by Müller and Sawitzki [JASA, 1991] and Hartigan [JASA, 1987] independently. It has been used widely to estimate density levels without such a framework. The result presented in this thesis implies that empirical excess mass maximization algorithms are successful learners without the necessity to prove this property for each algorithm separately.

# Zusammenfassung

In dieser Arbeit werden zwei offene Probleme bezüglich des Lernens von DNF-Formeln beziehungsweise von regulären Sprachen gelöst. Des Weiteren werden neue Forschungsergebnisse bei der Identifikation von Bereichen bestimmter Mindestdichte in Wahrscheinlichkeitsverteilungen präsentiert, den Dichteleveln.

Zuerst wird effizientes und sogenanntes „proper“ Lernen von $k$-term-DNF-Formeln untersucht. Im Zuge dessen wird ein Lernalgorithmus vorgestellt, der einzig aus positiven Beispielen lernt, wobei die Beispiele zufällig gemäß einer unbekannten $q$-beschränkten Verteilung gezogen werden. Diese Klasse von Wahrscheinlichkeitsverteilungen stellt eine Verallgemeinerung der Gleichverteilung dar. Mit dem vorgestellten Verfahren wird ein offenes Problem von Pitt und Valiant [J. ACM, 1988] gelöst. Anschließend wird ein negatives Ergebnis für das Lernen von poly($d$)-term-DNF vorgestellt.

Als Zweites wird das Lernen von regulären Sprachen, repräsentiert durch residuale alternierende Automaten, betrachtet. Alternierung ermöglicht die kompakte Beschreibung regulärer Sprachen, während Residualität den Automatenzuständen eine natürliche Bedeutung verleiht, die die spätere Analyse des Automaten vereinfacht. Es wird gezeigt, dass der $\texttt{AL}^\star$-Algorithmus, entworfen von Angluin, Eisenstat, und Fisman [Proc. IJCAI, 2015], vereinzelt nicht residuale alternierende Automaten (AFAs) ausgibt. Dies widerlegt eine Vermutung der Autoren. Folglich wird der neue Algorithmus $\texttt{AL}^{\star\star}$ entwickelt, der dieses Problem behebt und beweisbar stets residuale AFAs erzeugt. Dabei werden die gleichen asymptotischen Komplexitätsschranken eingehalten wie von $\texttt{AL}^\star$. Anschließend wird die Kompaktheit verschiedener Typen von endlichen Automaten untersucht.

Schlussendlich wird das Abschätzen von Dichteleveln mittels algorithmischen Lernens im Framework von Ben-David und Lindenbaum [JCSS, 1997] untersucht. Dieses Framework überträgt das PAC-Lernen auf das Lernen von Dichteleveln. Zunächst wird jedoch der Zusammenhang zwischen dem Lernen von Dichteleveln und der vollständigen Abschätzung der Wahrscheinlichkeitsverteilungsfunktion abgeklärt. Darauffolgend wird gezeigt, dass jeder Algorithmus, der die „empirische Überschussmasse“ maximiert, direkt im Framework zum Lernen von Dichteleveln angewendet werden kann. Bisher erschien das Framework von Ben-David und Lindenbaum wenig hilfreich beim Entwurf von Lernalgorithmen. Die Maximierung der empirischen Überschussmasse wurde unabhängig voneinander von Müller und Sawitzki [JASA, 1991] sowie von Hartigan [JASA, 1987] eingeführt. Sie wurde bereits außerhalb des Frameworks genutzt, um Dichtelevel abzuschätzen. Das hier vorgestellte Ergebnis impliziert, dass derartige Lernverfahren erfolgreich sind, ohne dies jedes Mal einzeln zeigen zu müssen.

# Contents

# 1 Introduction and Motivation

Machine learning is the algorithmic generalization of data in order to predict specific attributes of future data. The amount of data a learning algorithm can access before it needs to formulate a *hypothesis* is bounded to prevent it from trivial reproduction. Learning is not always possible, e. g. if the data is completely random. In the case of random data, only rote learning is possible, i. e. obtaining and memorizing all data.

Consider the model of *PAC learning*. It has been introduced by Valiant [61] and is kind of the standard model in *computational learning theory*. In this model, the learner shall identify an unknown subset – called the *concept* – of the *learning domain*. The learner receives a sample consisting of random examples from the learning domain according to an unknown arbitrary distribution. The examples are labeled by their membership of the concept. Learning algorithms are designed under an assumption about which concepts may occur at all. The set of these concepts is called the *concept class*. For example, a concept may be the set of binary strings satisfying a specific monomial. Then, the concept class may be the set of all concepts represented by a monomial. Whenever we talk about learning specific Boolean functions such as monomials, disjunctive normal form (DNF) formulas, and so on, we consider the concept class that consists of the concepts containing the satisfying assignments of such a formula each. Ideally, the learner identifies the target concept exactly. This may not always be possible, thus a slightly diverging hypothesis, which even may not be part of the concept class, could be returned. If the hypothesis generated by the learner always is guaranteed to be part of the concept class, learning is called *proper*.

Blumer et al. [14] have shown that the number of examples required for learning in the PAC model is linear in the complexity of the concept class, measured by the *VC dimension*. However, even for concept classes with small VC dimension, the learning problem may not be solvable within a reasonable time complexity. E. g., this is the case for proper learning of $k$-term DNF formulas – DNF formulas that consist of at most $k$ monomials – for $k \geq 2$, unless $\mathcal{RP} = \mathcal{NP}$ [51].

In computational learning theory, it is studied which concepts can be learned in which learning models at all and within which computational time complexity. This lays the foundation of practical machine learning methods and helps to understand how and why they work. Recently, there has been great progress in practice, e. g. see AlphaZero [58]. Most of the recent advances have been achieved without a well-founded theoretical basis. This may work for a certain time, but for further targeted development it is necessary to deeply understand the functional principles. Therefore, research in learning theory is important

especially to close the gap between theory and practice. In this thesis, we will consider three different problems from computational learning theory, regarding learnability of DNF formulas, finite automata, and probability distributions.

As already mentioned above, in the PAC model $k$-term DNF formulas cannot be learned properly and efficiently from arbitrary distributions of examples unless $\mathcal{RP} = \mathcal{NP}$. Pitt and Valiant [51] have formulated the question whether the situation changes if the set of possible distributions is restricted. Subsequent research indeed shows that if the family of probability distributions is restricted, the situation changes dramatically. Three notable results are as follows.

- Proper and efficient learning of $k$-term DNF formulas becomes possible over $q$-bounded distributions [34].
- However, depending on the application the learner may only have access to positive examples (satisfying assignments). The problem of learning $k$-term DNF formulas from satisfying assignments alone has been solved partially by Sakai and Maruoka [54] for the uniform distribution. The authors have constructed an efficient algorithm for learning the more restricted *monotone* DNF (MDNF) formulas – DNF formulas without negated variables – properly and from satisfying assignments alone. Moreover, the algorithm presented by the authors stays efficient even for $\log(d)$-term MDNF formulas, where $d$ denotes the number of different variables that may occur in the formula.
- More recently, De et al. [21] managed to learn general DNF formulas efficiently from uniformly distributed satisfying assignments alone, but not properly.

In Chapter 3 of this thesis, the problem of learning $k$-term DNF formulas properly and efficiently from satisfying assignments only over $q$-bounded distributions is solved. Also a negative result for unbounded DNF formulas in the considered learning model is presented.

Let us introduce the second learning problem studied in this thesis: learning regular languages by finite-state automata (FA). In 1987, Angluin [5] has introduced the $\mathtt{L}^\star$ algorithm that learns the minimal deterministic FA (DFA) for an unknown regular target language $L$ by asking *membership* and *equivalence queries*. While the answer to a membership query tells the learner whether a specific word is in $L$, an equivalence query is answered with a counterexample if the hypothesis presented within the query is not equivalent to the target language $L$. Since the minimal DFA may be quite large, some effort has been expended to learn more succinct automata like non-deterministic ones (NFA) or even alternating FAs (AFA). A first attempt to generate an NFA hypothesis was due to Yokomori [66]. However, in some cases the automata generated by his algorithm may be even larger than the equivalent minimal DFA [42]. The breakthrough for learning NFAs was the work of Bollig et al. [15], using a property called *residuality* that had been found by Denis et al. [26] before. Bollig et al. [15] presented the algorithm $\mathtt{NL}^\star$ for learning residual NFAs (RNFA), which may be more compact than DFAs. Alternating finite automata (AFA) may

allow an even more compact representation than NFAs while the membership problem can still be solved efficiently [17]. Angluin et al. [7] have constructed the learning algorithm $\mathtt{AL}^\star$ for AFAs more recently. They conjectured that their algorithm always outputs residual alternating automata automata (RAFAs), but they left this question open since they were not able to give a proof. Residuality is desirable, because it makes NFAs and AFAs more natural such that they may be easier to analyze.

The conjecture of Angluin et al. is disproved by a carefully designed counterexample in Chapter 4 of this thesis. Afterwards, a modified algorithm $\mathtt{AL}^{\star\star}$ is constructed and proven to always output RAFAs, solving the open problem.

Finally, we consider distribution learning. The learner receives unlabeled examples drawn at random according to an unknown distribution. The goal is to estimate the density function of the target distribution or at least a specific property of the distribution. Ben-David and Lindenbaum [11] have introduced a framework for learning high-density areas of probability distributions. The learner shall return a hypothesis $h$ such that the density function of the unknown target distribution exceeds a specific value for almost all elements $x \in h$. The authors mention applications like detection of accident-prone drivers, marketing analysis, or pattern recognition. However, we are aware of only one concrete application of this framework, see Ernst [28]. That may be because it seems hard to implement tests for the sufficient conditions introduced by Ben-David and Lindenbaum. Also it seems difficult to prove the correctness of such tests if non-trivial concept classes are considered. In Chapter 5, it is shown that the maximization of the *empirical excess mass* is an alternative sufficient condition for a successful learning algorithm in the framework. The empirical excess mass is a measure that has been introduced by Hartigan [36] and Müller and Sawitzki [48] independently. It has been studied in statistical distribution estimation, but has not been used in computational learning theory yet. Combining the framework of Ben-David and Lindenbaum with the empirical excess mass maximization yields applicability of already-known algorithms from statistical distribution estimation in the framework.

The results presented in Chapter 3 originate from common work with Maciej Liśkiewicz and Rüdiger Reischuk and have been published in [29]. Among other things, the author of this thesis has especially contributed the fundamental solution strategies and drafts of the proofs. Regarding Chapter 4, the results presented there have been published together with Sebastian Berndt, Maciej Liśkiewicz, and Rüdiger Reischuk before in [12]. Here, again among other things, the author of this thesis has contributed the construction of a complex software package to examine learning algorithms for finite automata, the non-residual counterexample for $\mathtt{AL}^\star$, and the method to ensure residuality of $\mathtt{AL}^{\star\star}$. Finally, the results presented in Chapter 5 originate from common work with Rüdiger Reischuk. The author of this thesis has especially contributed the fundamental solution strategies and drafts of the proofs.

# 2 Learning Models

In this chapter, an overview of models commonly used in the computational learning theory literature is given. The presentation focuses on PAC(-like) models (e. g. PAC, CN, SQ, ...) and perfect learning. Before a precise definition of each model is given, a general setting is presented that abstracts the different learning models to their similarities.

When the general setting has been defined, the main differences of the specific models will become visible. They differ in their (error) parameters and the error parameters' ranges as well as in the oracle the learner can access. For all learning models, their parameters and the parameters' range turn out to be a natural consequence of what the oracles allow. Thus, PAC-like learning is separated from perfect learning by the error parameters' range. Nevertheless, it seems natural to classify the learning models by their oracles, which then imply the parameters and parameters' restrictions.

Besides the pure models, there exist variants that can be applied to them. E. g., one can choose a specific family of underlying distributions, require efficient computability, or *proper* learning. These variants and the power of different models with and without variations will be discussed at the end of this chapter.

## 2.1 General Setting

Let $\mathcal{X} = \{\mathcal{X}_d : d \in \mathbb{N}\}$ denote the learning domain, a family of sets $\mathcal{X}_d$, where $\mathcal{X}_d$ contains elements with a representation of length $d$. In the following, $d$ will always denote the length of representation, also called *dimension* of $\mathcal{X}_d$.

A concept of dimension $d$ is a subset $c \subseteq \mathcal{X}_d$. We overload the notation with the characteristic function $c : \mathcal{X}_d \to \{0, 1\}$. Let $\mathcal{C}_d \subseteq 2^{\mathcal{X}_d}$ be a set of concepts of dimension $d$. A *concept class* $\mathcal{C} = \{\mathcal{C}_d : d \in \mathbb{N}\}$ is a family of sets $\mathcal{C}_d$.

For example, the concept class of monomials is a family of sets of all concepts that can be represented by a monomial. I. e., a concept $c$ of dimension $d$ is in $\mathcal{C}_d$ if there exists a monomial $M$ over $\mathcal{X}_d = \{0, 1\}^d$ such that $M = c$.

Let $\mathcal{D}_d$ denote a set of distributions over $\mathcal{X}_d$ and $\mathcal{D} := \{\mathcal{D}_d : d \in \mathbb{N}\}$. Further, let $\mathcal{H}_d$ be a set of hypotheses $h : \mathcal{X}_d \to \{0, 1\}$ with dimension $d$. The corresponding family of hypotheses is $\mathcal{H} = \{\mathcal{H}_d : d \in \mathbb{N}\}$. For a concept $c \in \mathcal{C}_d$ and a probability distribution $D \in \mathcal{D}$, let $O$ denote an oracle that provides some kind of information about $c$ and $D$.

**Definition 2.1** (Learning Algorithm)**.** *In the general setting, a* learning algorithm $\mathcal{A}$ *with hypothesis space* $\mathcal{H} = \{\mathcal{H}_d : d \in \mathbb{N}\}$ *receives parameters* $d$, $\varepsilon$, *and* $\delta$ *and generates a hypothesis* $h \in \mathcal{H}_d$. *For that purpose,* $\mathcal{A}$ *may ask its oracle* $O$ *at most a polynomial number of queries with respect to* $d$. *The number of queries asked is called* query complexity. $\mathcal{A}$ *is called* efficient *if its running time is also polynomially bounded. In the setting of* proper *learning,* $\mathcal{H}_d \subseteq \mathcal{C}_d$ *is required for every* $d \in \mathbb{N}$.

A learning algorithm may access several oracles. Formally, a set of oracles $O_1, \ldots, O_k$ can be combined to a single oracle $O_1 + \cdots + O_k$ that on a query $(i, Q)$ returns $O_i(Q)$.

**Definition 2.2** (Successful Learning Algorithm)**.** *Given certain ranges* $R_\varepsilon$ *and* $R_\delta$, *a* learning algorithm $\mathcal{A}$ *is successful for a concept class* $\mathcal{C}$ *over a class of distributions* $\mathcal{D}$ *if for every dimension* $d \in \mathbb{N}$, *every concept* $c \in \mathcal{C}_d$, *every probability distribution* $D \in \mathcal{D}_d$, *and every error parameters* $\varepsilon \in R_\varepsilon, \delta \in R_\delta$ *with probability at least* $1 - \delta$ *the error of the hypothesis* $h$ *output by* $\mathcal{A}$, *defined by*

$$\mathsf{error}(h) := \Pr_{x \in_D \mathcal{X}_d}[h(x) \neq c(x)] \ ,$$

*is less than* $\varepsilon$.

Typically $R_\varepsilon = R_\delta = (0, 1)$, but the situation $R_\varepsilon = R_\delta = \{0\}$, which is called *perfect learning*, is also considered. Learning *weakly without false positives* denotes that the condition $\mathsf{error}(h) < \varepsilon$ is supplemented by the additional requirement $h \subseteq c$. Learning *strongly without false positives* requires that the hypothesis must always satisfy $h \subseteq c$, independent of the confidence parameter $\delta$.

**Definition 2.3** (Learnability)**.** *A concept class* $\mathcal{C}$ *is (efficiently) learnable with respect to* $\mathcal{D}$ *if an (efficient) learning algorithm for* $\mathcal{C}$ *over* $\mathcal{D}$ *exists.*

*A concept class* $\mathcal{C}$ *is learnable in the* distribution-free setting *if* $\mathcal{C}$ *is learnable with respect to the family of all probability distributions.*

## 2.2 Learning from Examples

Let us organize the great quantity of different settings used in learning theory by the oracles that are available to the learner. In this section, different types of non-interactive oracles are considered. These oracles just provide some randomly chosen element each time they are queried. In the next section, interactive oracles that respond to queries will be described.

For learning models that use only non-interactive oracles, $\varepsilon$ and $\delta$ are restricted to positive values, because learning with error probability 0 is impossible. The query complexity of learning algorithms in these models must be bounded polynomially with respect to $d$, $1/\varepsilon$, and $1/\delta$. We begin with the following brief

definition of the $EX(D)$ oracle. This oracle provides random elements of $\mathcal{X}_d$ with respect to the hidden distribution $D \in \mathcal{D}_d$.

### 2.2.1 Learning from Labeled Examples

The probably approximately correct (PAC) learning has been introduced by Valiant [61]. The essence of this model are the confidence and error parameters $\delta > 0$ ("probably") and $\varepsilon > 0$ ("approximately"). The PAC model provides the basis for the other example-based learning models considered later in this section as well as for SQ learning (see Section 2.3.2). Strictly speaking, distribution-free learning is required in the PAC model and the models derived from PAC learning. We will specify the family of distributions $\mathcal{D}$ together with the learning model in each case. Thus, the following PAC definition is more about the oracle than the distributions.

**Definition 2.4.** *For a concept $c \in \mathcal{C}_d$ and a probability distribution $D \in \mathcal{D}_d$, let $PAC(c, D)$ denote the following oracle. When given a query to $PAC(c, D)$, it returns a pair $(x, c(x))$, where $x$ is drawn from $\mathcal{X}_d$ according to $D$ independently at random.*

*A learning algorithm $\mathcal{A}$ with a PAC oracle is called a PAC learner.*

### 2.2.2 Learning from Positive Examples Alone

For applications like steganography or machine learning, it is often difficult to obtain meaningful natural counterexamples, while positive examples can be gathered easily. E. g. for learning in steganography, innocuous digital images can be collected easily. However, it is not clear how suspicious images are distributed in general and how they can be obtained independent of a specific stegosystem [45, 29]. The PAC model can be modified as follows to cover the setting of *positive examples only.*

**Definition 2.5.** *For a concept $c \in \mathcal{C}_d$ and a probability distribution $D \in \mathcal{D}_d$, let $POS(c, D) = EX(D_c)$ be the sampling oracle where $D_c$ denotes the conditional probability distribution of $D$ over the set $c$, i. e. $D_c(x) = D(x)/D(c)$ if $x \in c$, and otherwise $D_c(x) = 0$, where $D(c) = \sum_{y \in c} D(y)$.*

*A learning algorithm $\mathcal{A}$ with a POS oracle is called a POS learner.*

### 2.2.3 Learning from Positive and Unlabeled Examples

Learning from positive examples alone is a hard requirement, especially in the distribution-free setting. In this setting, successful learning from a *POS* oracle alone implies that the learner must not output a hypothesis that contains any false positives. The reason is that every single false-positive could have an arbitrary large probability according to the unknown distribution over the

learning domain. Information about the distribution of elements outside the concept class may help the learner who can make a two-sided error in this setting. For a details discussion see Denis [24].

Denis has suggested to provide distribution information by an additional *EX* oracle. Note that learning with access to *POS* and *EX* oracles is not an artificial setting. Denis mentions applications in marketing analysis or medicine. In addition, applications are in steganalysis or intrusion detection, where positive examples (stegotexts or network attacks, resp.) are available among a large set of unclassified examples (set of images found on the internet or the entire network traffic, respectively).

**Definition 2.6.** *A learning algorithm $\mathcal{A}$ with a POS+EX oracle is called a POS+EX learner.*

### 2.2.4 Learning from Noisy Examples

Examples obtained in practice may sometimes be labeled incorrectly, e. g. when the examples are classified by humans. A learning algorithm that assumes every example to be labeled correctly might probably output a bad hypothesis or run into a deadlock. The learning model with classification noise (CN) has been introduced by Angluin [8].

In the CN model, the noise is stochastically independent of the examples. However, the noise rate $\eta$ is not known to the learning algorithm, but an upper bound $\eta_0 < 0.5$. The query complexity (respectively time complexity) of (efficient) learning algorithms in the noisy models must be bounded by a polynomial with respect to $d$, $1/\varepsilon$, $1/\delta$, and $1/(1/2 - \eta_0)$.

**Definition 2.7.** *For a concept $c \in \mathcal{C}_d$ and a probability distribution $D \in \mathcal{D}_d$, let $CN_\eta(c, D)$ denote an oracle defined as follows. Every time the $CN_\eta(c, D)$ oracle is queried, with probability $1 - \eta$ it returns $(x, c(x))$, and with probability $\eta$ the pair $(x, 1 - c(x))$. In both cases, $x$ is drawn from $\mathcal{X}_d$ independently at random according to $D$.*

*$\mathcal{A}$ is called a CN learner if for all $CN_\eta$ oracles with $\eta \leq \eta_0$, $\mathcal{A}$ is a learning algorithm with access to $CN_\eta$.*

The CN model assumes that the noise is constant. This strict requirement can be relaxed as follows. The constant partition classification noise (CPCN) model, introduced by Decatur [23], is a generalization of the CN model. The domain $\mathcal{X}_d$ is partitioned into a constant number $k$ of arbitrary regions $X_1, \ldots, X_k$, where each partition $X_i$ has its own noise rate $\eta_i \leq \eta_0$. The actual partitioning is not known to the learner. Ralaivola et al. [53] have shown that a concept class $\mathcal{C}$ can be learned (efficiently) in the CN model if and only if it can be learned (efficiently) in the CPCN model.

### 2.2.5 Learning High-density Regions of Distributions

This model has been introduced by Ben-David and Lindenbaum [11]. The goal is to identify regions where the unknown probability measure attains high values. The learner can access an *EX* oracle to perform this task.

This model has similarities to a noisy *POS* setting, where elements of the concept are sampled with higher probability than elements outside of the concept. The details of this model will be presented in Chapter 5.

## 2.3 Learning with Queries

Now let us consider interactive oracles that react to (adaptive) queries and provide specific information requested by the learner.

### 2.3.1 Learning from Membership Queries and Counterexamples

In this model, which has been introduced by Angluin [5], the learner can ask two different types of queries to its oracle:

- For a *membership query*, the oracle $MEMBER(c)$ is called with some element $x \in \mathcal{X}_d$ and returns $c(x)$.
- For an *equivalence query*, the oracle $EQUIV(c)$ is called with some hypothesis $h : \mathcal{X}_d \to \{0, 1\}$ and returns "yes" if $h = c$. Otherwise it returns a counterexample $x \in \mathcal{X}_d$ such that $h(x) \neq c(x)$.

The oracles provide exact information without randomness, hence in principal perfect learning can be achieved if there is no bound on the number of queries.

**Definition 2.8.** *$\mathcal{A}$ is called a learner from membership queries and counterexamples if $\mathcal{A}$ is a perfect learner with access to a MEMBER+EQUIV oracle.*

In Section 2.4.1, we will see that the *EQUIV* oracle can be simulated by a *PAC* oracle if errors are allowed. This leads to a variation of the MEMBER+EQUIV model towards the PAC+MEMBER model: the *MEMBER* oracle may help the PAC learner to remove uncertainty about elements he has not received by sampling. In this model, perfect learning is not required particularly with respect to the *PAC* oracle.

### 2.3.2 Learning from Statistical Queries

Learning from Statistical Queries (SQ) has been introduced by Kearns [38]. The motivation for his model is that every learning algorithm in the SQ model can be transformed to a CN learner and it is easier to design an algorithm and prove its correctness in the SQ model.

The learner has access to an SQ oracle $STAT(c, D)$ which he can ask the following type of queries. A statistical query $(\chi, \tau)$ consists of a set of labeled examples $\chi \subseteq \mathcal{X}_d \times \{0, 1\}$ and an accuracy parameter $0 < \tau < 1$. When $STAT(c, D)$ receives a statistical query $(\chi, \tau)$, it returns an estimation $\hat{P}$ of $P = \Pr_{x \in_D \mathcal{X}_d}[(x, c(x)) \in \chi]$, which is the probability that a $PAC(c, D)$ example is in $\chi$. The estimation satisfies $|\hat{P} - P| \leq \tau$. A set $\chi$ may be compactly represented, e. g. by an algorithm that computes a membership test. The length of representation and the computation time of membership testing must be bounded by a polynomial with respect to $d$ and $1/\varepsilon$.

The learner $\mathcal{A}$ needs to choose $\tau$ in order to adapt it to its required accuracy $\varepsilon$. To prevent $\mathcal{A}$ from choosing arbitrary small values, we require $1/\tau$ to be polynomial in $d$ and $1/\varepsilon$.

Contrary to learning from non-interactive oracles, an SQ learner does not have to deal with rare cases in which the relative frequencies notably deviate from the underlying distribution. The SQ oracle guarantees that the probability estimation always satisfies the requested accuracy $\tau$. Thus, the real probability of specific events is known to the learner with just a small deviation. Contrary to that, a *PAC* oracle may provide a sample whose relative probability differs massively from the real underlying probability distribution. In such a case, the learner has no chance to find an appropriate hypothesis and will fail consequently. Even though this case is rare, it makes learning impossible if the learner is not allowed to fail completely sometimes. The confidence parameter $\delta$ determines how often the learner is allowed to fail. A choice of $\delta > 0$ is necessary in PAC learning to enable learning ever. On contrary, an SQ learner does not have to deal with this problem, thus a confidence of $\delta = 0$ can be achieved – and consequently is required in this model. However, the approximation error $\tau > 0$ of $\hat{P}$ makes it impossible to achieve perfect learning, thus an error of $\varepsilon > 0$ must be allowed.

**Definition 2.9.** *$\mathcal{A}$ is called an SQ learner if $\mathcal{A}$ is a learning algorithm with an SQ oracle.*

In [25], Denis describes learning from positive statistical queries. In this model, the SQ oracle $STAT(c, D)$ is replaced by $POSSTAT{+}EXSTAT$, where $POSSTAT(c, D) := STAT(c, D_c)$ and $EXSTAT(c, D) := STAT(\mathcal{X}, D)$. While the *STAT* oracle returns estimates for the distribution of a *PAC* oracle, *POSSTAT* estimates a *POS* oracle and *EXSTAT* estimates an *EX* oracle. Thus, Denis' model is an SQ variant of learning from positive and unlabeled examples. More types of interactive oracles for learning with queries can be found in [6].

## 2.4 Relationship of Learning Models

In the previous sections, the following main attributes of learning models have been discussed: the oracles the learner may access, the distribution family, requirements regarding time efficiency, and proper learning. These attributes

can be combined in different ways and each combination may lead to a different set of learnable concept classes. For example, the class of $k$-term DNF formulas can be PAC learned in the distribution-free setting efficiently if and only if proper learning is not required [51].

### 2.4.1 The Power of Oracles

We denote the set of concept classes that are distribution-free learnable without requirements regarding efficiency or proper learning within a specific model by the name of the corresponding oracle. For example, $STAT = \{\mathcal{C} : \mathcal{C} \text{ is SQ learnable}\}$.

The following relationships of learning models are known:

$$\begin{aligned} POSSTAT{+}EXSTAT &\subseteq STAT \\ &\subsetneq STAT{+}EX \\ &\subseteq CPCN \\ &= CN \\ &\subseteq POS{+}EX \\ &\subseteq PAC \subseteq PAC{+}MEMBER\ . \end{aligned}$$

The relations $POSSTAT{+}EXSTAT \subseteq STAT \subseteq STAT{+}EX$ hold trivially as well as $PAC \subseteq PAC{+}MEMBER$.

Feldman [30] has shown that $STAT{+}EX$ is strictly more powerful than $STAT$ alone. For a special designed concept class $\text{Line}_p$, learning with the help of a $STAT{+}EX$ oracle is possible with polynomial sample complexity, while for a pure $STAT$ oracle every algorithm needs an exponential sample complexity in $d$ in order to bound the hypothesis's error to $\varepsilon$.

To show $STAT{+}EX \subseteq CN$, the $STAT{+}EX$ oracle can be simulated with the help of a $CN$ oracle as follows. For $STAT$ queries, use the technique of Kearns [38]. For $EX$ queries simply remove the labeling from the output of the $CN$ oracle.

Denis [24] has shown a method to simulate a $CPCN$ oracle using a $POS{+}EX$ oracle. The simulation of the $CPCN$ oracle answers a query as follows. With probability 2/3, it returns an example from $POS$ tagged with a positive label. On the other hand with probability 1/3, it returns an example from $EX$ tagged with a negative label. Applying this method, all examples that are not inside the concept are labeled correctly. Examples of the concept may be labeled incorrectly as negative, but only with probability $D(c)/3$. Note that the distribution of examples is changed by this simulation. However, this does not affect the learnability of a concept if the error parameter $\varepsilon$ is adjusted appropriately. This implies $CPCN \subseteq POS{+}EX$. For more details see [24].

For $POS{+}EX \subseteq PAC$, the $EX$ oracle can be simulated trivially by a $PAC$ oracle. To simulate the $POS$ oracle, the $PAC$ oracle is queries many times

until it returns a positive example. If positive examples are rare and no one is returned by the *PAC* oracle after polynomially many queries, a simulation of the *POS* oracle is not necessary. In this case, the empty hypothesis is a good approximation for the target concept, and can be returned by the learner. The relation $POS{+}EX \subseteq PAC$ has been observed originally by Denis [24].

In the following we look at some further relations of oracles that are not covered by the inequation above.

As already mentioned in Section 2.2.3, learning from a *POS* oracle alone may be difficult in the distribution-free setting. By continuing with this observation, the weakness of a pure *POS* oracle can be proven. The lack of information about the probability distribution of negative examples yields an information-theoretical difference between a *POS* oracle and a $POS{+}EX$ setting [24]. Thus, at least for proper learning, $POS \subsetneq POS{+}EX$. More detailed results for learning from a *POS* oracle alone can be found in [49, 57]

Under cryptographic assumptions and for efficient learning, it holds that $PAC \subsetneq PAC{+}MEMBER$. This follows from the learnability of the concept class of deterministic finite automata (DFAs). This class can be learned in the PAC model with an additional *MEMBER* oracle [5]. However, under cryptographic assumptions DFAs cannot be learned efficiently in the pure PAC model [41].

Finally, let us compare the *PAC* oracle with the *EQUIV* oracle. We have noted before that the *EQUIV* oracle may allow perfect learning for specific concept classes, while for non trivial concept classes this is impossible using a *PAC* oracle. However, the error requirements in the corresponding learning models take this into account. So when comparing the oracles, one may ask: "Can a perfect learning algorithm with an *EQUIV* oracle be transformed into a PAC algorithm with bounded error $\varepsilon, \delta > 0$?" Angluin [6] has shown that the answer is "yes". Every equivalence query can be simulated by a number of *PAC* queries depending on the error parameters. In the same work, she has also shown that the other direction is not true, at least if time efficiency is considered. Thus, $EQUIV \subseteq PAC$ and for efficient learning this inclusion is strict.

### 2.4.2 Distribution Families

In the previous subsections, we have seen that information about the distribution, even if they can only be obtained from an *EX* oracle, may help the learner as well for SQ learning ($STAT \subseteq STAT{+}EX$) as for learning from positive examples alone (for proper learning $POS \subsetneq POS{+}EX$). If the set of distributions $\mathcal{D}$ is restricted to specific families of distributions, this may obviate the need for the *EX* oracle, depending on the concrete distribution family. In this section, we discuss the situation for the uniform distribution. Since this family contains only one distribution for each dimension $d$ of the learning domain $\mathcal{X}_d$, in this setting the learner has full knowledge about the probability distribution. However, most results discussed in this section can be extended to more general distribution families like $q$-bounded distributions, product distributions, or

smooth distributions (for a definition of these distributions see e. g. [56], or Chapter 3 for $q$-bounded distributions).

Even when a learner has access to a *PAC* oracle, the distribution family matters for learnability of specific concept classes. An example is the class of $k$-term DNF again. For efficient and proper PAC learning, $k$-term DNF is not PAC learnable in the distribution-free setting [51], but it can be learned if the distribution family $\mathcal{D}$ is restricted to the uniform distribution [34].

### 2.4.3 Absolute and Relative Error

Let us consider learning from positive examples alone over the uniform distribution. In this setting, the definition of the error plays an important role, which we will discuss in the following. This matters especially when the hypothesis is used for sampling, as e. g. in grey-box steganography (see [45] for sampling from DNF formulas for grey-box steganography).

Let us assume that $D$ is the uniform distribution over $\mathcal{X}_d$ and the concept $c$ is very small such that $D(c) \leq \varepsilon$. In this case, the error measure given in Definition 2.2 (the absolute error) allows a region of false positives with up to the same probability weight as the concept itself. Assume that $h$ is such a hypothesis, i. e. $c \subseteq h$ and $D(h \setminus c) = \varepsilon \geq D(c)$. When sampling uniformly at random from the hypothesis, at most half of the examples are expected to lie inside the concept, because $D(h \setminus c) \geq D(h \cap c) = D(c)$. On the other hand, at least half of the examples drawn from $h$ uniformly at random are expected to lie outside of $c$.

One solution is to explicitly disallow false positives. However, this does not solve an other issue with small concepts: successful learning with the absolute error measure becomes trivial for small concepts, because the empty hypothesis can always be returned for such concepts, because it satisfies the error bound $\mathsf{error}(\emptyset) = D(c) \leq \varepsilon$.

The relative error measure, which has been used e. g. by Shvaytser [57] and De et al. [21], solves this issue. It is defined by

$$\mathsf{error}_{\mathsf{rel}}(h) := \Pr_{x \in_D \mathcal{X}_d}[h(x) \neq c(x)] \;/\; \Pr_{x \in_D \mathcal{X}_d}[c(x)] \;\; .$$

This measure is perfectly suited if learning from positive examples alone over the uniform distribution, because it solves the issues mentioned above. This has also been discussed by De et al. [21] before.

# 3 Learning *k*-term DNF Formulas From Satisfying Assignments

This chapter discusses learning of $k$-term DNF formulas over $\mathcal{X}_d = \{0,1\}^d$. A proper learning algorithm for $k$-term DNF is developed. Afterwards, a negative result of Kearns et al. [39] regarding learnability of $d$-term DNF from positive examples without false positives is strengthened.

The setting is as follows. Only positive examples, i. e. satisfying assignments of the $k$-term DNF formula that represents the unknown target concept $c$, are available to the learner via a *POS* oracle. These examples are distributed randomly according to an unknown $q$-bounded distribution, which is defined as follows.

**Definition 3.1.** *A distribution $D$ over $\mathcal{X}_d$ is called* $q$-bounded *if, for a rational number $q \geq 1$,*

$$\max_{a \in \mathcal{X}_d}\{D(a)\} \;\leq\; q \cdot \min_{a \in \mathcal{X}_d}\{D(a)\} \;.$$

With probability $1 - \delta$, the learner has to output a hypothesis $h$ in $k$-term DNF with a relative error of at most $\varepsilon$. In every case, the learning process must be efficient and the final hypothesis must satisfy the condition $h \subseteq c$, i. e. learning must be strongly without false positives.

Our learning algorithm is divided into two phases. In the first phase, a standard procedure is applied to learn a $k$-CNF representation of the target formula strongly without false positives. Afterwards, a set of maximal monomials that should cover most of the area of this $k$-CNF formula is constructed. Since the number of these monomials can be much greater than $k$, in a second phase with a separate sequence of positive examples, tests are performed to find a subset of size at most $k$ as final hypothesis.

An application for learning in this setting comes from grey-box steganography. In steganography, a sender wants to transmit a hidden message to a transceiver by embedding it into unsuspicious documents, called coverdocuments, such that an observer cannot determine whether an additional hidden message is transmitted. In the model of grey-box steganography, which has been introduced by Liśkiewicz et al. [45], the sender does not know the coverdocuments look like. However, he has access to a *POS*oracle that samples those coverdocuments. Hence the sender has to find a suitable hypothesis about these documents first. Afterwards, he has to generate stegodocuments in order to embed his secret message. To prevent detection by the adversary, the distribution of these stegodocuments must be indistinguishable from the distribution of coverdocuments. Liśkiewicz et al. have

shown that, given a DNF formula that describes a set of uniformly distributed coverdocuments exactly, the sender can efficiently generate stegodocuments that are indistinguishable from the coverdocuments. An equivalent CNF formula is not sufficient for this task, because it does not allow efficient sampling since the satisfiability problem for $k$-CNF formulas is $\mathcal{NP}$-hard if $k \geq 3$. Consequently, if the coverdocuments follow a uniform distribution over the satisfying assignments of an unknown $k$-term DNF formula, a learning algorithm for $k$-term DNFs in our setting yields a secure and efficient grey-box stegoencoder [45, 29].

Note that learning strongly without false positives is not mandatory to build a secure stegosystem as long as the relative error is small enough, i. e. such that it is very unlikely that the sender samples a stegodocument from its hypothesis that is not part of the concept representing the coverdocuments. However, the benefits of the stricter setting are as follows. Whenever the sender samples a false positive, an adversary that knows the true concept can detect the deviation, and consequently the use of a stegosystem, with absolutely certainty. Without false positives the adversary can never be absolutely sure about the use of a stegosystem. If the entire communication observed does not contain any non-coverdocument messages, the adversary can only make use of a probabilistic analysis. Thus, learning strongly without false positives is a nice property for grey-box steganography, because it completely prevents the (unlikely) risk of definite detection.

## 3.1 Related Work

For proper learning DNF formulas from a PAC oracle with two-sided error, the following results have been shown. The fastest learning algorithm for unrestricted DNF formulas in the distribution-free setting has been designed by Klivans and Servedio [43] with time complexity $2^{\tilde{\mathcal{O}}(d^{1/3})}$. Pitt and Valiant [51] have shown that efficient and proper learning of DNF formulas is impossible in the distribution-free setting, even for restricted $k$-term DNF formulas with $k \geq 2$, unless $\mathcal{RP} = \mathcal{NP}$. For $k = 1$ we actually consider monomials, which are learnable in this setting. For the uniform distribution, Verbeurgt [64] has given a proper PAC learning algorithm for unrestricted DNFs that runs in quasi-polynomial time, roughly $d^{\mathcal{O}(\log d)}$. It is open whether proper learning of unrestricted DNF formulas can be done efficiently over the uniform distribution. For $k$-term DNF formulas, this is possible due to Flammini et al. [34].

Let us consider known results for learning DNFs from positive examples alone without false positives and regarding the relative error measure. Since every POS learning algorithm satisfying the relative error bound is also a PAC learning algorithm, the negative result of Pitt and Valiant holds here as well, while monomials stay learnable. When the distribution family is restricted to $q$-bounded distributions, 2-term DNF formulas become learnable efficiently due to Flammini et al. [34]. Their algorithms uses the relative error measure and learns weakly without false positives. Thus, the result presented in the next

sections is an improvement even for 2-term DNFs since it guarantees learning strongly without false positives. Unrestricted DNF formulas, resp. $d$-term DNFs, cannot be learned weakly without false positives from a sub-exponential number of examples if only a *POS* oracle is used, regardless of the computation time and hypothesis space. This negative result even holds when the distribution family is restricted to the uniform distribution [39]. Since the uniform distribution is $q$-bounded for every $q \geq 1$, this is a negative result for $q$-bounded distributions as well. In Section 3.7 this result is strengthened even more. It is shown that no subfamily of $q$-bounded distributions allows learning of $d$-term DNF formulas in the setting described just here. This is a stronger result, because specific $q$-bounded distributions may set a higher probability to regions that are easy to learn and a lower probability to the the harder ones, which may reduce the relative error of the hypothesis. The results for proper and efficient learning of DNFs from a POS oracle without false positives are summarized in Table 3.1.

| concept class | distribution-free | uniform / $q$-bounded |
|---|---|---|
| 1-term DNF (monomials) | yes [61] | yes [61] |
| 2-term DNF | no [49, 51] | yes [34] |
| $k$-term DNF | no [49, 51] | yes (Theorem 3.5) |
| log-term DNF | no [51] | open |
| unrestricted DNF | no [51] | no [39] / no (Theorem 3.19) |

Table 3.1: Positive and negative results for efficient and proper learning of DNF formulas from positive examples with one-sided error over several distributions. The negative results of [51] (unless $\mathcal{RP} = \mathcal{NP}$) for $k$-term, with $k \geq 3$, log-term, and unrestricted DNFs even hold for learning with two-sided error from positive and negative examples.

If the formulas to be learned are restricted to monotone DNF (MDNF) formulas, Sakai and Maruoka [54] have presented an efficient algorithm that properly learns $\log(d)$-term MDNF formulas from satisfying assignments drawn randomly according the uniform distribution. Another notable result on learning (non-monotone) DNF formulas over the uniform distribution with the relative error measure $\mathsf{error}_{\mathsf{rel}}$ is due to De et al. [21]. They use only a *POS* oracle, but have to allow two-sided errors and do not learn properly. From a sample with size bounded by a polynomial in $d$, $k$, and $1/\varepsilon$, De et al. can construct a hypothesis that is suitable for sampling, where $k$ is the number of terms of the (unrestricted) target DNF formula. Their algorithm runs in time $d^{\mathcal{O}(k/\varepsilon)}$. Since the hypothesis is not a DNF formula, it is not known whether embedding of given hidden messages for steganography is possible with their sampler. The result of De et al. does not contradict Kearns et al. [39], because the negative result only holds for learning with one-sided error or in the distribution-free setting, respectively. We present the first efficient *proper* learning algorithm for $k$-term DNF formulas from positive samples alone – and even strongly without false positives.

## 3.2 On Monomials, *k*-term DNFs, and Other Formulas

The following terms an facts about Boolean formulas are needed to investigate the learnability of $k$-term DNF formulas in this chapter.

The number of Boolean variables $x_1, \ldots, x_d$ is always equal to the dimension $d$. $x_i^0 = \overline{x_i}$ denotes the negation of $x_i$ whereas $x_i^1 = x_i$. For a Boolean formula $\varphi$ let $\mathtt{sat}(\varphi) := \{a \in \{0,1\}^d : \varphi(a) = 1\}$ denote the set of assignments that satisfy $\varphi$, which will also be called the *support* of $\varphi$. For the algorithms described below it may happen that $\varphi$ becomes the empty formula $\lambda$. Depending on the surroundings, we define $\mathtt{sat}(\lambda) = \{0,1\}^d$ if $\lambda$ is considered as a special case of CNF formulas and $\mathtt{sat}(\lambda) = \emptyset$ in case of DNF formulas. A formula is called *unate* if for each variable $x_i$, it does not contain positive and negative occurrences of $x_i$.

Considering $\{0,1\}^d$ as an $d$-dimensional cube the support of a monomial $M$ over $x_1, \ldots, x_d$ will always be a subcube. We may assume that no monomial contains a literal twice or becomes *trivial* by containing a variable and its complement. Two monomials $M$ and $M'$ are considered *identical* if $\mathtt{sat}(M) = \mathtt{sat}(M')$ – that means they have the same set of literals. A monomial of length $\ell$, that means consisting of $\ell$ variables, has a support of size $2^{d-\ell}$.

A $k$-term DNF formula $\varphi$ is a disjunction of at most $k$ monomials. $\varphi$ is called *non-redundant* if it does not contain a monomial $M$ such that removing $M$ from $\varphi$ does not change $\mathtt{sat}(\varphi)$, in particular there are no identical or trivial monomials. Throughout this thesis only non-redundant DNF formulas are considered. The support of a $k$-term DNF is the union of at most $k$ subcubes.

For a $k$-term DNF $\varphi$ consisting of monomials $M_1, \ldots, M_k$ of length $\ell_1, \ldots, \ell_k$ the size of $\mathtt{sat}(\varphi)$ can be at most $2^d \sum_{i=1}^{k} 2^{-\ell_i}$. If a distribution $D$ is $q$-bounded, for every $a \in \{0,1\}^d$ the probability $D(a)$ has to be at least $q^{-1}\ 2^{-d}$ and at most $q\ 2^{-d}$. Thus, for $q$-bounded distributions $D$, $D(\mathtt{sat}(M_i)) \leq q\ 2^{-\ell_i}$.

**Definition 3.2.** *A monomial $M$ is* shorter *(with respect to its length) than a monomial $M'$ if $M$ consists of fewer literals than $M'$ – which by excluding trivial monomials is equivalent to $|\mathtt{sat}(M)| > |\mathtt{sat}(M')|$, i. e. $M$ has a* larger support *than $M'$.*

*On the other hand, a* proper submonomial *of $M$ is obtained by removing some literals from $M$, thus enlarging the support.*

Note that a proper submonomial of $M$ is always shorter and has a larger support, but a shorter monomial does not have to be a submonomial. The following notion plays an important role for the learner to be constructed.

**Definition 3.3.** *Let $\psi$ be a Boolean formula and $a \in \mathtt{sat}(\psi)$. A monomial $M$ is* $(\psi, a)$-maximal *if $a \in \mathtt{sat}(M)$ and $M$ is a* prime implicant *of $\psi$, i. e. $\mathtt{sat}(M) \subseteq \mathtt{sat}(\psi)$ and there is no proper submonomial of $M$ fulfilling this inclusion.*

*Let $\mathcal{S}(\psi, a)$ denote the set of all $(\psi, a)$-maximal monomials. For an integer $c > 0$ we call a subset $\mathcal{S}' \subseteq \mathcal{S}(\psi, a)$ a $(\psi, a, c)$-*set *if $|\mathcal{S}'| = \min\{c, |\mathcal{S}(\psi, a)|\}$, that means $\mathcal{S}'$ contains $c$ many maximal monomials or all monomials of $\mathcal{S}(\psi, a)$ if their number is fewer than $c$.*

A $k$-CNF formula $\psi$ is given by a conjunction of clauses each containing at most $k$ literals. We assume that no clause contains a literal more than once or a variable and its negation (a trivial clause). Also no clause should be empty which would make $\psi$ unsatisfiable. Similarly to DNF formulas, a CNF formula fulfilling these properties is called *non-redundant* and we will consider only such CNF formulas. The support of a clause with $k$ literals is the complement of a subcube of dimension $d - k$. This implies that the support of a $k$-CNF formula $\psi$ is the intersection of such complementary subcubes, which may become arbitrarily complex.

## 3.3 The *k*-term DNF Learner

Flammini et al. [34] have developed an algorithm for proper learning of $k$-term DNF formulas from positive and negative examples with two-sided error. In a first phase, their algorithm generates candidate monomials from positive examples alone. Every monomial of the unknown target DNF formula $\varphi$ is guaranteed to become a candidate monomial with high probability if the support of that monomial is large enough. In general, the number of candidate monomials may be much larger than $k$ and some of these candidates may satisfy a lot of assignments that are not satisfied by $\varphi$. Thus, in a second phase, combinations of at most $k$ candidate monomials are tested against a set of positive and negative examples in order to find a $k$-term DNF hypothesis. If such a combination fulfills a specific error bound then it becomes the output. Flammini et al. have shown that their method yields an approximately correct hypothesis with high probability.

In the following an alternative approach that can handle the lack of negative examples is described. In the second phase, monomials with false positives cannot be detected if only positive examples are available. Thus, in the first phase our approach generates only candidate monomials strongly without false positives. With this set of candidate monomials, our learner can properly learn a hypothesis from positive examples only. The hypothesis will not include false positives, but is still close to the real concept.

Our main challenge is to find appropriate candidate monomials. We start by constructing a $k$-CNF formula from a first sample. Then the main technical difficulty is to construct appropriate monomials from this CNF formula. This strategy in pseudo code is listed in Algorithm 1.

It has been shown how to learn a $k$-term DNF formula $\varphi$ strongly without false positives by using as hypothesis space $k$-CNF formulas [51, 61, 14]. The learner starts with the conjunction of all possible non-tautological clauses of length at

**Input:** number of variables $d$, distribution bound $q$, error parameter $\varepsilon$, confidence parameter $\delta$, oracle $POS(\texttt{sat}(\varphi), D)$ respecting the unknown target $k$-term DNF formula $\varphi$ and the unknown $q$-bounded distribution $D$
**Output:** $k$-term DNF hypothesis

1 $\varepsilon_1 \leftarrow \varepsilon\, q^{-2}\, k^{-1}\, 2^{-(3k+1)}$;
2 $n_1 \leftarrow \varepsilon_1^{-1}\, ((2d+1)^k + \ln(4/\delta))$;
3 get $n_1$ positive examples $E = (e_1, \ldots, e_{n_1})$ from $POS(\texttt{sat}(\varphi), D)$;
4 learn $k$-CNF formula $\psi$ using sample $E$;
5 $\mathcal{M} \leftarrow \emptyset$;
6 **for** $j \leftarrow 1$ **to** $n_1$ **do**
7 `// generate a` $(\psi, e_j, 2^k - 1)$`-set` $\mathcal{M}_j$
8 $\mathcal{M}_j \leftarrow$ `MaxMonomials`$(\psi, e_j, 2^k - 1)$;
9 $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_j$;
10 $n_2 \leftarrow 48\, \varepsilon^{-2}\, (k^2 \ln(2)\ + k \ln(n_1) + \ln(4/\delta))$;
11 get another $n_2$ positive examples $S = (s_1, \ldots, s_{n_2})$ from $POS(\texttt{sat}(\varphi), D)$;
12 **foreach** *subset* $W$ *of* $\mathcal{M}$ *of size at most* $k$ **do**
13 $\varphi_W \leftarrow \bigvee_{M \in W} M$;
14 **if** $\varphi_W$ *misclassifies fewer than* $3\varepsilon n_2/4$ *examples of* $S$ **then return** $\varphi_W$;
15 **return** $\bigvee_{M \in W} M$ for some $W \subseteq \mathcal{M}$ of size $k$; `// default output`

**Algorithm 1:** `Learn-`$k$`-Term-DNF`$(d, q, \varepsilon, \delta)$

most $k$, of which there are at most $(2d+1)^k$. Then clauses not satisfied by a positive example are deleted. This strategy is used by `Learn-`$k$`-Term-DNF` in Line 4. Figure 3.1 illustrates such a hypothesis $\psi$ for a target formula $\varphi$ consisting of three monomials.

**Fact 3.4.** *For every* $\varepsilon_1 > 0$*, using* $\varepsilon_1^{-1}\, ((2d+1)^k + \ln(4/\delta))$ *positive examples one can learn a* $k$*-CNF formula* $\psi$ *for* $\texttt{sat}(\varphi)$ *such that* $\mathsf{error}_{\mathsf{rel}}(\psi) < \varepsilon_1$ *with probability at least* $1 - \delta/4$.

We construct candidate monomials for $\varphi$ by extracting monomials from $\psi$. For $k \geq 3$ it is $\mathcal{NP}$-hard to find a single satisfying assignment for a $k$-CNF formula in general. But here we already know a number of satisfying assignments, namely the positive examples used to create $\psi$. Therefore, we can define a criterion for potential candidate monomials generated from $\psi$ and an example $a \in \texttt{sat}(\psi)$: $M$ has to be $(\psi, a)$-maximal (for an illustration see Fig. 3.1).

In Line 8 for every positive example $e_i$ used to learn the $k$-CNF formula $\psi$, the algorithm selects several $(\psi, e_i)$-maximal monomials, but at most $2^k - 1$ many. The algorithm `MaxMonomials` that computes these monomials will be explained in detail in the next section.

Finally, to construct the output, a $k$-term DNF formula $\varphi'$, the learner chooses a subset of $k$ monomials in $\mathcal{M} = \bigcup_j \mathcal{M}_j$, where $\mathcal{M}_j$ denotes the $(\psi, e_j, 2^k - 1)$-set

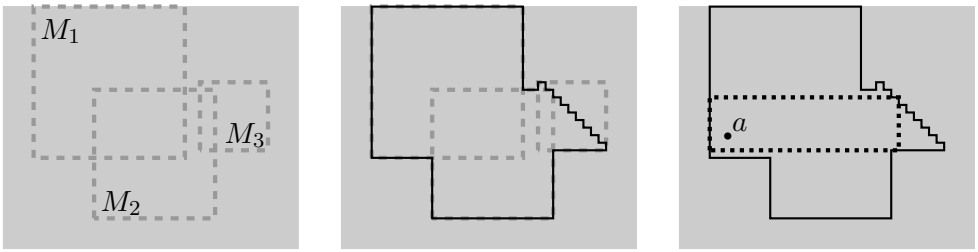


Figure 3.1: A running example for $k = 3$. The grey region illustrates the set $\{0,1\}^d$ of all possible assignments.
*Left:* For a target 3-term DNF $\varphi = M_1 \vee M_2 \vee M_3$, the support of each monomial is indicated as a dashed-line rectangle.
*Middle:* The support of the (hypothetical) 3-CNF formula $\psi$ that approximates $\varphi$ is surrounded by the solid black line. To compare, we show also the supports of the monomials $M_i$. On the right a scattered region (like a staircase) is generated that requires many rectangles to be covered.
*Right:* The dotted rectangle shows the support of a $(\psi, a)$-maximal monomial that is bounded to the right by the scattered region. Here we use the intuitive convention that $(\psi, a)$-maximal monomials are represented as maximal rectangles (with horizontal and vertical sides) containing $a$ and included in a region illustrating the support of $\psi$.

for an example $e_j$, such that the formula $\varphi'$ consisting of these monomials misclassifies a small fraction of the fresh sample. This learner achieves the following properties.

**Theorem 3.5.** *For every constant $k$, distribution bound $q \geq 1$, error $\varepsilon > 0$, and confidence parameter $\delta > 0$,* `Learn-k-Term-DNF`$(d, q, \varepsilon, \delta)$ *learns $k$-term DNF formulas from positive examples strongly without false positives over $q$-bounded distributions. The sample size can be bounded by*

$$\begin{aligned}\sigma_k(d,q,\varepsilon,\delta) \;:=\; & \varepsilon^{-1}\, q^2\, k\, 2^{3k+1} \Big((2d+1)^k + \ln(4/\delta)\Big) \;+ 48\varepsilon^{-2}\,(4k^2+k)\ln(2) \\ & + 48\varepsilon^{-2}\,\Big(k \ln\Big(\varepsilon^{-1}\, q^2\, k\,\Big((2d+1)^k + \ln(4/\delta)\Big)\Big) + \ln(4/\delta)\Big)\;.\end{aligned}$$

*Its time complexity is polynomially bounded with respect to $(d, q, 1/\varepsilon, \log 1/\delta)$.*

We note that our learning algorithm can be made applicable even if the parameter $q$ is unknown: The method described by Flammini et al. [34] that makes their learning algorithm applicable for unknown $q$ can be used here as well. The learning algorithm starts with $q = 1$ and doubles $q$ until the sequential test finds a hypothesis that is satisfied by enough examples. To bound the error, $\delta$ is halved in each step. The average running time stays polynomial in $d$, $q$, $1/\varepsilon$, and $\log 1/\delta$.

Theorem 3.5 is proven in the following sections.

## 3.4 Generating Maximal Monomials for a CNF Formula

Let $\psi$ be a non-redundant $k$-CNF formula over variables $x_1, \ldots, x_d$ with a satisfying assignment $a$ and $c$ a natural number. For a set $\mathcal{M}$ of monomials, define $\mathtt{sat}(\mathcal{M}) := \bigcup_{M \in \mathcal{M}} \mathtt{sat}(M)$. For a set $I \subseteq [d]$, let $e_I = e_1 \ldots e_d \in \{0,1\}^d$ denote the string with $e_i = 1 \Leftrightarrow i \in I$. Now consider the following procedure for computing a $(\psi, a, c)$-set. The algorithm is based on a modified version of Algorithm I of [1] as follows. First, every literal from $\psi$ that is not satisfied by $a$ is removed, making the formula unate. Then the algorithm deals with such DNF formulas (instead of monotone ones). Every membership query in [1] is replaced by an evaluation of $\psi$.

**Input:** $k$-CNF formula $\psi$; satisfying assignment $a \in \mathtt{sat}(\psi)$;
requested number of monomials $c > 0$
**Output:** a $(\psi, a, c)$-set $\mathcal{M}$

1 let $\psi'$ denote the formula obtained by removing every literal from $\psi$ that is not satisfied by $a$;
2 $\mathcal{M} \leftarrow \emptyset$;
3 **while** $|\mathcal{M}| < c$ **do**
4 by brute-force search find a set $I \subseteq [d]$ of size at most $|\mathcal{M}|$ such that $a \oplus e_I \in \mathtt{sat}(\psi') \setminus \mathtt{sat}(\mathcal{M})$;
5 **if** *such a set does not exist* **then return** $\mathcal{M}$;
6 **while** *there exists some* $i \in [d] \setminus I$ *with* $a \oplus e_{I \cup \{i\}} \in \mathtt{sat}(\psi')$ **do** $I \leftarrow I \cup \{i\}$ ;
7 $M_I \leftarrow \bigwedge_{i \in [d] \setminus I} x_i^{a_i}$;
8 $\mathcal{M} \leftarrow \mathcal{M} \cup \{M_I\}$;
9 **return** $\mathcal{M}$;

**Algorithm 2:** `MaxMonomials`$(\psi, a, c)$

In the following analysis, $\psi, a, c$ always denote the inputs for `MaxMonomials` and $\psi'$ the formula generated in Line 1.

**Claim 3.6.** *The set $\mathcal{S}(\psi, a)$ of all $(\psi, a)$-maximal monomials is equal to the set $\mathcal{S}(\psi', a)$ of all $(\psi', a)$-maximal monomials.*

*Proof.* A monomial $M$ is a prime implicant of $\psi$ if and only if for every clause $K$ of $\psi$, at least one literal of $K$ appears in $M$ and for every literal $\ell$ in $M$, there is at least one clause $K$ in $\psi$ such that $\ell$ is the only literal of $K$ that occurs in $M$. For prime implicants $M$ that are $(\psi, a)$-maximal monomials, the condition $a \in \mathtt{sat}(M)$ implies that all literals in $M$ must be satisfied by $a$.

Thus, since $a$ is a satisfying assignment of both $\psi$ and $\psi'$, we can conclude that for all monomials $M$, with $a \in \mathtt{sat}(M)$, the following equivalence is true.

- For every clause $K'$ of $\psi'$ at least one literal of $K'$ appears in $M$ and
- for every literal $\ell$ in $M$, there is at least one clause $K'$ in $\psi'$ such that $\ell$ is the only literal of $K'$ that occurs in $M$

*if and only if*

- for every clause $K$ of $\psi$ at least one literal of $K$ appears in $M$ and
- for every literal $\ell$ in $M$, there is at least one clause $K$ in $\psi$ such that $\ell$ is the only literal of $K$ that occurs in $M$.

This implies $\mathcal{S}(\psi, a) = \mathcal{S}(\psi', a)$. □

**Claim 3.7.** *For every monomial $M$ that is contained in the final output $\mathcal{M}$ of* `MaxMonomials`*, the following holds:*

(*i*) $a \in \texttt{sat}(M)$,
(*ii*) $\texttt{sat}(M) \subseteq \texttt{sat}(\psi')$.

*Proof.* Property (*i*) is fulfilled because only literals that are satisfied by assignment $a$ are added to $M$ in Line 7 of `MaxMonomials`. For property (*ii*) it suffices to show that for every index set $I \subseteq [d]$ the condition $a \oplus e_I \in \texttt{sat}(\psi')$ implies $\texttt{sat}(M_I) \subseteq \texttt{sat}(\psi')$. Since $\psi'$ only contains literals that are satisfied by $a$, $a \oplus e_I \in \texttt{sat}(\psi')$ implies $\forall I' \subseteq I : a \oplus e_{I'} \in \texttt{sat}(\psi')$ and thus $\texttt{sat}(M_I) \subseteq \texttt{sat}(\psi')$. □

**Claim 3.8.** *Let $\mathcal{M}'$ be the value of the variable $\mathcal{M}$ in Line 8 of* `MaxMonomials` *at an arbitrary iteration of the outer while loop.*

(*iii*) *If $\texttt{sat}(\mathcal{M}') \subsetneq \texttt{sat}(\psi')$ and $a \in \texttt{sat}(M)$ for all $M \in \mathcal{M}'$ then there exists a set $I \subseteq [d]$ of size $|I| \leq |\mathcal{M}'|$ such that $a \oplus e_I \in \texttt{sat}(\psi') \setminus \texttt{sat}(\mathcal{M}')$.*
(*iv*) *If $\texttt{sat}(\mathcal{M}') \subsetneq \texttt{sat}(\psi')$, $a \in \texttt{sat}(M)$ for all $M \in \mathcal{M}'$, and in Line 4 a set $I \subseteq [d]$ satisfying $a \oplus e_I \in \texttt{sat}(\psi') \setminus \texttt{sat}(\mathcal{M}')$ is found then* `MaxMonomials` *finds a new $(\psi', a)$-maximal monomial $M_I$ in Line 7.*
(*v*) *If $\texttt{sat}(\mathcal{M}') = \texttt{sat}(\psi')$ then $\mathcal{M}'$ contains every $(\psi', a)$-maximal monomial.*

*Proof.* Every vector $b \in \{0,1\}^d$ can be written as $b = a \oplus e_{I'}$ for a suitable index set $I' \subseteq [d]$. Thus, take an arbitrary $b \in \texttt{sat}(\psi') \setminus \texttt{sat}(\mathcal{M}')$ and the corresponding $I'$. Since $\psi'$ can only be satisfied with the help of a variable $x_i$ by taking the value $a_i$ it holds that $a \oplus e_I \in \texttt{sat}(\psi')$ for every subset $I$ of $I'$ as well.

We exploit the following property of monomials. If $a \in \texttt{sat}(M)$ and $a \oplus e_{I'} \notin \texttt{sat}(M)$ then an index $i \in I'$ exists such that $a \oplus e_{\{i\}} \notin \texttt{sat}(M)$. Furthermore, $a \oplus e_I \notin \texttt{sat}(M)$ for every set $I$ with $\{i\} \subseteq I$. Now, we construct the set $I$ by selecting such an index $i$ for every monomial in $\mathcal{M}'$. Then, $|I| \leq |\mathcal{M}'|$ and $a \oplus e_I \notin \texttt{sat}(M)$ for every $M$ in $\mathcal{M}'$, which proves property (*iii*).

If the search in Line 4 is successful a monomial $M_I$ covering $a \oplus e_I$ is missing. The set $I$ contains all literals that definitely will not occur in $M_I$. In Line 6 further literals are selected for removal, but ensuring that one stays within $\texttt{sat}(\psi')$. When Line 6 is completed, no more literals can be removed without

violating this condition. Therefore, the monomial $M_I$ constructed in Line 7 is a new prime implicant of $\psi'$. Since $a \in \mathtt{sat}(M_I)$ (by $(i)$ in Claim 3.7) $M_I$ is a $(\psi', a)$-maximal monomial. This proves property $(iv)$.

A prime implicant $M$ is called *essential* if there is some assignment satisfying $M$, but no other prime implicant. It is well known that for unate formulas every prime implicant is essential. Equation $\mathtt{sat}(\mathcal{M}') = \mathtt{sat}(\psi')$ implies that every essential and hence every prime implicant of $\psi'$ must be in $\mathcal{M}'$. In particular, $\mathcal{M}'$ contains all $(\psi', a)$-maximal monomials. □

**Lemma 3.9.** `MaxMonomials` *correctly computes a* $(\psi, a, c)$*-set and needs time* $\mathcal{O}(d^c)$*.*

*Proof.* First we note that the algorithm computes a set $\mathcal{M}$ of $(\psi', a)$ maximal monomials. By Claim 3.6 we have that $\mathcal{S}(\psi, a) = \mathcal{S}(\psi', a)$.

Consider the case $c > |\mathcal{S}(\psi', a)|$. By $(iv)$ in Claim 3.8, every monomial $M_I$ generated is a $(\psi', a)$-maximal monomial. Property $(ii)$ in Claim 3.7 and $(iii)$ in Claim 3.8, imply that new $(\psi', a)$-maximal monomials are added to $\mathcal{M}$ until $\mathtt{sat}(\mathcal{M}) = \mathtt{sat}(\psi')$. Then, by property $(v)$, every $(\psi, a)$-maximal monomial has been found.

If $c \leq |\mathcal{S}(\psi, a)|$, by the same arguments as above, a new $(\psi, a)$-maximal monomial is constructed until $|\mathcal{M}| = c$ holds and then $\mathcal{M}$ is provided as output. Thus, $\mathcal{M}$ is a $(\psi, a, c)$-set.

The brute-force search in Line 4 needs time $\mathcal{O}(d^{|\mathcal{M}|})$. The other lines within the outer while-loop can be executed in time $\mathcal{O}(d)$. This results in a running time bounded by $\mathcal{O}(\sum_{i=1}^{c} d^i) = \mathcal{O}(d^c)$. □

## 3.5 Bounding the Subset Size for Maximal Monomials

In this section we analyze the $(\psi, e_j, 2^k - 1)$-sets $\mathcal{M}_j$ that are generated by `Learn-`$k$`-Term-DNF` in Line 8. For some examples $e_j$ there can be a huge number of $(\psi, e_j)$-maximal monomials. Since the algorithm has to run in polynomial time, it computes only $(\psi, e_j, 2^k - 1)$-sets for $\mathcal{M}_j$ instead of $\mathcal{S}(\psi, e_j)$. By this strategy, random monomials may be skipped and only monomials with small supports might be selected, i. e. those that cover only small sub-regions of $\psi$. As a consequence, $(\psi, e_j)$-maximal monomials with large supports that are crucial to construct a good output hypothesis might be skipped. Thus, the restriction to $(\psi, e_j, 2^k - 1)$-sets requires extra attention for the analysis of the algorithm.

Figure 3.2 illustrates this problem for the 3-term DNF target formula $\varphi$ and the 3-CNF hypothesis $\psi$ that have been presented in Fig. 3.1. For the example $e$ on the left that satisfies the monomial $M_1$ of $\varphi$ with large support, marked as a dashed-line rectangle, there are nine $(\psi, e)$-maximal monomials: eight monomials – two of them are drawn in grey – which cover only a small portion of satisfying assignments of $\psi$ (the grey region) and one large $(\psi, e)$-maximal

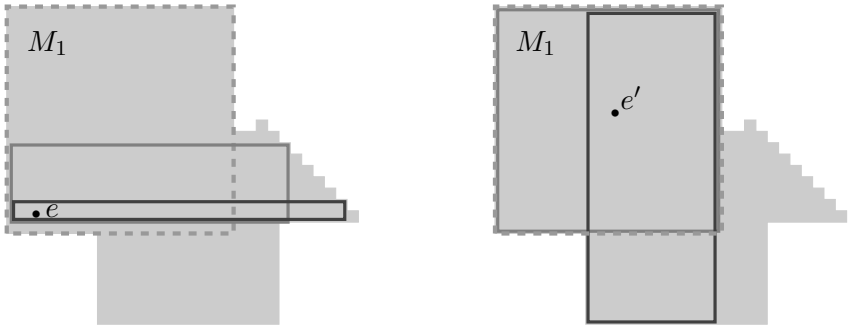


Figure 3.2: The grey region represents the support of the 3-CNF formula $\psi$ of Fig. 3.1 that approximates the target 3-term DNF $\varphi = M_1 \vee M_2 \vee M_3$. The dashed-line rectangle illustrates the support of $M_1$.
*Left:* The assignment $e$ shows a positive example and the two solid-line rectangles the supports of two possible $(\psi, e)$-maximal monomials of which there are many more.
*Right:* For the assignment $e'$, only two $(\psi, e')$-maximal monomials exist, one of them coinciding with $M_1$.

monomial which coincides with $M_1$. However, since $9 > 2^k - 1$ for $k = 3$, the large maximal monomial could be omitted. Thus, with Proposition 3.11 it will be proven that if the algorithm gets enough positive examples $E$, then for every monomial $M_i$ of $\varphi$ of large support there exists an example $e' \in E$ as illustrated in Fig. 3.2 on the right with high probability. The example $e'$ has at most $2^k - 1$ maximal monomials and one of them includes the support $\mathtt{sat}(M_i)$. For such an example it is guaranteed that `Learn-`$k$`-Term-DNF` computes a set $\mathcal{M}_j$ in Line 8 which contains a $(\psi, e')$-maximal monomial with support including $\mathtt{sat}(M_i)$. Since it will be shown that such an example exists for every relevant monomial, all monomials of $\varphi$ with support large enough will be added to $\mathcal{M}$ in Line 9 of `Learn-`$k$`-Term-DNF` and thus be considered by the brute-force search in the second phase of the algorithm. In Fig. 3.2, $e'$ has only two $(\psi, e')$-maximal monomials. One of them is identical to the dashed-line monomial $M_1$.

Let the unknown $k$-term DNF formula $\varphi$ to be learned consist of monomials $M_1 \ldots, M_k$ ordered by increasing length.

**Definition 3.10.** *For $\gamma$ with $0 \leq \gamma \leq 1$ we call $M_i$ $\gamma$-relevant if $|\,\mathtt{sat}(M_i)| \geq \gamma\,|\,\mathtt{sat}(\varphi)|$. Furthermore, let*

$$\hat{\gamma} \;:=\; \frac{\varepsilon}{2\,q\,k} \;.$$

In the proposition below we analyze $\hat{\gamma}$-relevant monomials of $\varphi$.

**Proposition 3.11.** *With probability at least $1 - \delta/2$, for every $\hat{\gamma}$-relevant monomial $M_i$ of $\varphi$ the following holds:* `Learn-`$k$`-Term-DNF` *draws at least one example $e \in \mathtt{sat}(M_i)$ in Line 3 such that*
*(a) the number of $(\psi, e)$-maximal monomials is at most $2^k - 1$, and*
*(b) there exists a $(\psi, e)$-maximal monomial $M_i'$ with $\mathtt{sat}(M_i) \subseteq \mathtt{sat}(M_i')$.*

*Proof.* As illustrated in Fig. 3.1, $\mathtt{sat}(\psi)$ may cover only a part of the satisfying region of a monomial in a scattered way and there could exist many $(\psi, e)$-maximal monomials.

**Lemma 3.12** ([18, 47])**.** *For every k-term DNF, the number of its prime implicants is bounded by* $2^k - 1$*.*

This implies that the number of different $(\varphi, a)$-maximal monomials over all $a \in \mathtt{sat}(\varphi)$ is bounded by $2^k - 1$ as well. Next we will derive a bound on the number of satisfying assignments for those maximal monomials that potentially intersect scattered regions of $\varphi$, outside of $\hat{\gamma}$-relevant monomials.

**Lemma 3.13.** *For* $u \in \mathbb{N}$ *let* $m(u)$ *be the number of monomials* $M_i$ *of* $\varphi$ *with* $|\,\mathtt{sat}(M_i)| \geq 2^u$*. Let*

$$\varphi_u = M_1 \vee \cdots \vee M_{m(u)}$$

*be the disjunction of these monomials (remember the monomials* $M_1, \ldots, M_k$ *of* $\varphi$ *are ordered increasingly by length) and*

$$\chi_u = M_{m(u)+1} \vee \cdots \vee M_k$$

*the disjunction of the remaining monomials. Define*

$$\mathit{Tail}_u \;:=\; \mathtt{sat}(\chi_u) \setminus \mathtt{sat}(\varphi_u) \quad \text{and} \quad \xi_u \;:=\; \bigvee_{a \in \mathit{Tail}_u} \bigvee_{M \in \mathcal{S}(\varphi, a)} M.$$

*Then* $|\,\mathtt{sat}(\xi_u)| \;\leq\; 2^{u+k-1}$*.*

*Proof.* If $\mathtt{sat}(\chi_u) \subseteq \mathtt{sat}(\varphi_u)$ then $|\,\mathtt{sat}(\xi_u)| = 0$. Otherwise, consider an assignment $a \in \mathit{Tail}_u$. Let $\varphi_u\langle a\rangle$ denote the formula derived from $\varphi_u$ by removing every literal that supports $a$, thus $\mathtt{sat}(\varphi_u\langle a\rangle) \supseteq \mathtt{sat}(\varphi_u)$, but still $a \notin \varphi_u\langle a\rangle$. Further, let $M^a$ denote the unique monomial with $\mathtt{sat}(M^a) = \{a\}$. To continue the analysis we make some claims – their proofs will be given at the end of this proof.

**Claim 3.14.** *For every* $z \in \mathtt{sat}(\xi_u)$*, there exists an assignment* $a \in \mathit{Tail}_u$*, such that every* $(\varphi_u\langle a\rangle \vee M^a, a)$*-maximal monomial is satisfied by* $z$*.*

Next, for every $a \in \mathit{Tail}_u$ we select a fixed, but arbitrary $(\varphi_u\langle a\rangle \vee M^a, a)$-maximal monomial $M_{u,a}$ and define $\xi'_u$ to be a disjunction of such monomials:

$$\xi'_u \;:=\; \bigvee_{a \in \mathit{Tail}_u} M_{u,a}.$$

Due to Claim 3.14 we know that $\mathtt{sat}(\xi_u) \subseteq \mathtt{sat}(\xi'_u)$.

**Claim 3.15.** *Every* $(\varphi_u\langle a\rangle \vee M^a, a)$*-maximal monomial* $M_{u,a}$ *has at most* $2^{m(u)}$ *satisfying assignments.*

Using these properties we can conclude the upper bound stated in Lemma 3.13 as follows:

$$|\,\mathtt{sat}(\xi_u)| \;\leq\; |\,\mathtt{sat}(\xi'_u)| \;\leq\; \sum_{a\in \mathit{Tail}_u} |\,\mathtt{sat}(M_{u,a})| \;\leq\; |\,\mathtt{sat}(\chi_u)|\; 2^{m(u)}$$
$$\leq\; 2^u\;(k-m(u))\;2^{m(u)} \;\leq\; 2^{u+k-1}.$$

The last inequality is true since $m(u) \leq k$.

It remains to prove Claim 3.14 and Claim 3.15.

*Proof of Claim 3.14.* Consider an arbitrary assignment $z \in \mathtt{sat}(\xi_u)$.

- **Case 1:** $z \notin \mathtt{sat}(\varphi_u)$.
  Since $z \in \mathtt{sat}(\varphi_u \vee \chi_u)$, we get $z \in \mathit{Tail}_u$. Obviously, for $a = z$, every $(\varphi_u\langle a\rangle \vee M^a, a)$-maximal monomial is satisfied by $z$.
- **Case 2:** $z \in \mathtt{sat}(\varphi_u)$.
  Note that in this case $z \notin \mathit{Tail}_u$. Let $M$ be a monomial of $\xi_u$, with $z \in \mathtt{sat}(M)$. Furthermore, there must be at least one assignment in $(\mathtt{sat}(M) \cap \mathtt{sat}(\chi_u)) \setminus \mathtt{sat}(\varphi_u)$ due to the construction of $\xi_u$. From this set we choose an assignment $a$ with minimal Hamming distance to $z$.

  Let $y$ be an arbitrary assignment obtained from $a$ by flipping one of the positions at which $a$ and $z$ differ. Note that $y \in \mathtt{sat}(M) \cap \mathtt{sat}(\varphi_u)$. Let $M_j$ be a monomial in $\varphi_u$ that is satisfied by $y$. During the construction of $\varphi_u\langle a\rangle$, all literals that are satisfied by $a$ are removed from $M_j$. Thus, the counterpart of $M_j$ in $\varphi_u\langle a\rangle$ – monomial $M_j\langle a\rangle$ – consists only of one literal $\ell_{a,y}$, i. e. $M_j\langle a\rangle = \ell_{a,y}$, with $a \notin \mathtt{sat}(\ell_{a,y})$ and $y \in \mathtt{sat}(\ell_{a,y})$. Hence, no $(\varphi_u\langle a\rangle \vee M^a, a)$-maximal monomial contains the literal $\neg\ell_{a,y}$.

  This argument can be repeated for every choice of $y$ and all $\neg\ell_{a,y}$ that satisfy $a$, but not $z$. Since every $(\varphi_u\langle a\rangle \vee M^a, a)$-maximal monomial is satisfied by $a$, it is also satisfied by $z$.

□

*Proof of Claim 3.15.* Let $M_{u,a}$ be a $(\varphi_u\langle a\rangle \vee M^a, a)$-maximal monomial. It holds $\{a\} = \mathtt{sat}(M^a) \subseteq \mathtt{sat}(M_{u,a})$. Thus, $M^a$ only differs from $M_{u,a}$ in some additional literals, denoted as $\ell_1, \ldots, \ell_r$, i. e. $M^a = M_{u,a} \wedge \ell_1 \wedge \ldots \wedge \ell_r$. This implies $|\,\mathtt{sat}(M_{u,a})| = 2^r\; |\,\mathtt{sat}(M^a)| = 2^r$ and below we argue that $r$ cannot exceed the value of $m(u)$.

Let $M^{a,\neg\ell}$ denote the monomial that is formed by replacing $\ell$ by $\neg\ell$ in $M^a$. We consider a literal $\ell_s$, with $s \in \{1, \ldots, r\}$. From $\mathtt{sat}(M^{a,\neg\ell_s}) \subseteq \mathtt{sat}(M_{u,a}) \subseteq \mathtt{sat}(\varphi_u\langle a\rangle \vee M^a)$, it follows that the only assignment satisfying $M^{a,\neg\ell_s}$ satisfies $\varphi_u\langle a\rangle$, too. Hence, there must be at least one monomial $M_t$ in $\varphi_u\langle a\rangle$, with $t \in \{1, \ldots, m(u)\}$, that is satisfied by the same assignment as $M^{a,\neg\ell_s}$. This means that $M_t$ cannot contain any negated literal from $M^{a,\neg\ell_s}$. By the construction of $\varphi_u\langle a\rangle$, the monomial $M_t$ cannot contain any literal from $M^a$. Therefore, the

only possibility remaining is that $M_t = \neg\ell_s$. If $r > m(u)$, then the number of monomials of $\varphi_u\langle a\rangle$ would be larger than $m(u)$. This is a contradiction. □

This completes the proof of Lemma 3.13. □

Next, for every monomial $M_i$ of $\varphi$ a lower bound on the number of assignments in $\mathtt{sat}(\varphi)$ that satisfy only $M_i$ will be shown. For $i \in [k]$ let formula $\varphi_{\text{-}i}$ be generated from $\varphi$ by removing $M_i$.

**Lemma 3.16.** *Choose a monomial $M_i$ of $\varphi$. Then $|\,\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_{\text{-}i})| \;\geq\; |\,\mathtt{sat}(M_i)| \cdot 2^{-k+1}$.*

*Proof.* Consider a monomial $M_j \neq M_i$ of $\varphi$. There must be at least one literal $\ell_j$ in $M_j$ that is not contained in $M_i$. So, $\mathtt{sat}(M_i \wedge \neg\ell_j) \cap \mathtt{sat}(M_j) = \emptyset$ and $|\,\mathtt{sat}(M_i) \setminus \mathtt{sat}(M_j)| \geq |\,\mathtt{sat}(M_i \wedge \neg\ell_j)| \geq |\,\mathtt{sat}(M_i)| \cdot 2^{-1}$. Successively, we obtain $|\,\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_{\text{-}i})| \geq |\,\mathtt{sat}(M_i)| \cdot 2^{-k+1}$. □

Now in order to bound the size of the scattered regions $\chi_u$, let us estimate how well a $k$-CNF formula $\psi$ can approximate the monomials $M_i$ of $\varphi$. Let $\varepsilon_1 = \varepsilon\; q^{-2}\, k^{-1}\; 2^{-(3k+1)}$ be the error parameter defined in `Learn-`$k$`-Term-DNF`.

**Lemma 3.17.** *Let $\psi$ be a $k$-CNF formula with $\mathtt{sat}(\psi) \subseteq \mathtt{sat}(\varphi)$.*
*If $\mathsf{error}_{\mathsf{rel}}(\psi) < \varepsilon_1$ then $\mathtt{sat}(M_i) \subseteq \mathtt{sat}(\psi)$ for every $(2^{-2k}\,\hat{\gamma})$-relevant monomial $M_i$ of $\varphi$.*

*Proof.* Assume $\mathsf{error}_{\mathsf{rel}}(\psi) < \varepsilon_1$, let $M_i$ be a $(2^{-2k}\,\hat{\gamma})$-relevant monomial of $\varphi$, and consider an arbitrary $a \in \mathtt{sat}(M_i) \setminus \mathtt{sat}(\psi)$. There must be a clause $K$ in $\psi$ with $a \notin \mathtt{sat}(K)$. This clause cannot contain any literal from $M_i$. So, $K$ is not satisfied by at least $2^{-k}|\,\mathtt{sat}(M_i)|$ assignments that satisfy $M_i$. Even in the case that for the unknown $q$-bounded distribution $D$ all these assignments are $q$-times less likely than any other assignment satisfying $\varphi$, one has

$$\begin{aligned}\mathsf{error}_{\mathsf{rel}}(\psi) \;&\geq D(\mathtt{sat}(M_i \wedge \neg\psi)) \,/\, D(\mathtt{sat}(\varphi)) \;\geq\; 2^{-k}\,|\,\mathtt{sat}(M_i)| \,/\, (q\,|\,\mathtt{sat}(\varphi)|)\\ &\geq 2^{-k}\; 2^{-2k}\; \varepsilon\; q^{-1}\; k^{-1}\; 2^{-1} \quad q \;=\; 2^{-(3k+1)}\; \varepsilon\; q^{-2}\; k^{-1} \;=\; \varepsilon_1 \;.\end{aligned}$$

This is a contradiction. Consequently, such an assignment $a$ does not exist and the lemma is proven. □

Thus, if a $k$-CNF formula $\psi$ approximates a $k$-term DNF formula $\varphi$ without false positives quite well, then every monomial of $\varphi$ with support not too small is completely covered by $\psi$. Only monomials with very small support may give rise to errors in the approximation.

Now the proof of Proposition 3.11 can be finished. Let $M_i$ be a $\hat{\gamma}$-relevant monomial of $\varphi$ and let $\varphi_{\text{-}i}$ be defined as above. Applying Lemma 3.13 with $u = \lceil \log(\hat{\gamma}\,|\,\mathtt{sat}(\varphi)|) - 2k \rceil$ one gets

$$|\,\mathtt{sat}(\xi_u)| \leq 2^{u+k-1} \;\leq\; 2^{-k}\,\hat{\gamma}\,|\,\mathtt{sat}(\varphi)| \leq 2^{-k}\,|\,\mathtt{sat}(M_i)| \;.$$

Define $\mathtt{usat}_d(M_i) := \mathtt{sat}(M_i) \setminus (\mathtt{sat}(\varphi_{\text{-}i}) \cup \mathtt{sat}(\xi_u))$ as the set of assignments uniquely satisfying $M_i$. Then, by Lemma 3.16 we obtain

$$\begin{aligned} |\,\mathtt{usat}_d(M_i)| &\geq 2^{-k} |\,\mathtt{sat}(M_i)| \\ &\geq 2^{-k}\,\hat{\gamma}\,|\,\mathtt{sat}(\varphi)| \;=\; \varepsilon\, q^{-1}\, k^{-1}\, 2^{-(k+1)}\,|\,\mathtt{sat}(\varphi)| \;. \end{aligned} \tag{3.1}$$

**Claim 3.18.** *With probability at least $1-\delta/4$ for every $\hat{\gamma}$-relevant monomial $M_i$ of $\varphi$, the sample $E$ given to* `Learn-`$k$`-Term-DNF` *in Line 3 contains at least one element $e$ such that $e \in \mathtt{usat}_d(M_i)$.*

Using this claim (to be shown below) and the previous Lemmata both properties $(a)$ and $(b)$ of Proposition 3.11 can be proven as follows. First let us argue that the number of $(\psi, e)$-maximal monomials is bounded by $2^k - 1$. Due to Claim 3.18 we know that with probability at least $1-\delta/4$ for every $M_i$, as assumed in the proposition, the sample $E$ contains an element $e \in \mathtt{usat}_d(M_i)$. Consider the monomials $\tilde{M}$ that are $(\varphi, e)$-maximal. Since $e \notin \mathtt{sat}(\xi_u)$, the construction of $\xi_u$ yields $\mathtt{sat}(\tilde{M}) \cap (\mathtt{sat}(\chi_u) \setminus \mathtt{sat}(\varphi_u)) = \emptyset$, because otherwise an assignment $a$ in $\mathtt{sat}(\tilde{M}) \cap (\mathtt{sat}(\chi_u) \setminus \mathtt{sat}(\varphi_u))$ would cause a $(\varphi, a)$-maximal monomial $M = \tilde{M}$ in $\xi_u$ and therefore $e \in \mathtt{sat}(\xi_u)$. Therefore, $\mathtt{sat}(\tilde{M}) \subseteq \mathtt{sat}(\varphi_u)$ and $\tilde{M}$ is $(\varphi_u, e)$-maximal. This implies $\mathcal{S}(\varphi, e) = \mathcal{S}(\varphi_u, e)$.

If $\mathsf{error}_{\mathsf{rel}}(\psi) < \varepsilon_1$ Lemma 3.17 yields $\mathtt{sat}(\varphi_u) \subseteq \mathtt{sat}(\psi) \subseteq \mathtt{sat}(\varphi)$ and hence $\mathcal{S}(\psi, e) = \mathcal{S}(\varphi_u, e)$. The number of $(\varphi_u, e)$-maximal monomials is bounded by $2^k - 1$ due to Lemma 3.12. Further we can conclude that at least one of the $(\psi, e)$-maximal monomials $M_i'$ covers $M_i$ completely.

By adding the probabilities of failure of the event in Claim 3.18 and $[\mathsf{error}_{\mathsf{rel}}(\psi) < \varepsilon_1]$ (Fact 3.4) together, with probability at least $1-\delta/2$ for every $\hat{\gamma}$-relevant monomial $M_i$ of $\varphi$ such an example $e$ exists in the sample $E$.

It remains to prove Claim 3.18.

*Proof of Claim 3.18.* For the number $n_1$ of examples in $E$ it holds

$$n_1 = \varepsilon^{-1}\, q^2\, k\, 2^{3k+1}\, ((2d+1)^k + \ln(4/\delta)) > \varepsilon^{-1}\, q^2\, k\, 2^{k+1}\, \ln(4k/\delta) \;.$$

Fix a $\hat{\gamma}$-relevant monomial $M_i$ of $\varphi$. The probability $p$ that the sample $E$ given to `Learn-`$k$`-Term-DNF` in Line 3 does not contain any element $e \in \mathtt{usat}_d(M_i)$ is bounded by

$$p \leq \left(1 - \frac{|\,\mathtt{usat}_d(M_i)|}{q\,|\,\mathtt{sat}(\varphi)|}\right)^{N_1} \;,$$

because the underlying distribution $D$ is $q$-bounded. Due to (3.1),

$$\begin{aligned} p &\leq (1 - \varepsilon\, q^{-1}\, k^{-1}\, 2^{-(k+1)} \,/\, q)^{N_1} \\ &< (1 - \varepsilon\, q^{-2}\, k^{-1}\, 2^{-(k+1)})^{\varepsilon^{-1}\, q^2\, k\, 2^{k+1}\, \ln(4k/\delta)} \\ &\leq (1/e)^{\ln(4k/\delta)} = \delta/(4k) \;, \end{aligned}$$

because for every $0 < \alpha < 1$ it holds that $(1-\alpha)^{1/\alpha} \leq 1/e$.

The probability that for at least one $\hat{\gamma}$-relevant monomial $M_i$ of $\varphi$ the sample $E$ given to `Learn-`$k$`-Term-DNF` in Line 3 does not contain any element $e \in \mathtt{usat}_d(M_i)$ is bounded by $k \cdot p \leq \delta/4$. □

This completes the proof of Proposition 3.11. □

## 3.6 The Correctness and Complexity of `Learn-`$k$`-Term-DNF`

In this section we conclude that `Learn-`$k$`-Term-DNF`$(d, q, \varepsilon, \delta)$ satisfies the properties stated in Theorem 3.5 for the target $k$-term DNF $\varphi$.

*Proof of Theorem 3.5.* The output hypothesis $\varphi'$ of the learner is a disjunction of maximal monomials. The support of every maximal monomial is a subset of the support of the $k$-CNF formula $\psi$. Since $\mathtt{sat}(\psi) \subseteq \mathtt{sat}(\varphi)$ it follows that the output hypothesis has no false positives.

Suppose that for every monomial $M_i$ of $\varphi$ with $|\,\mathtt{sat}(M_i)| \geq g$ the learner has added a monomial $M_i'$ with $\mathtt{sat}(M_i') \supseteq \mathtt{sat}(M_i)$ to $\mathcal{M}$. In the final stage as possible hypotheses the set of all up to $k$-fold disjunctions of monomials of $\mathcal{M}$ are used. Denote the set as $\mathcal{H}_\mathcal{M}$. Its size can be bounded by

$$|\mathcal{H}_\mathcal{M}| \;\leq\; \sum_{i=0}^{k} \binom{|\mathcal{M}|}{i} < \left(n_1 \cdot 2^k\right)^k \quad . \tag{3.2}$$

Let us call a hypothesis $h \in \mathcal{H}_\mathcal{M}$ *good* if for every monomial $M_i$ of $\varphi$ with $|\,\mathtt{sat}(M_i)| \geq g$, $h$ includes a maximal monomial $M_i'$ as above. For every good hypothesis, fewer than $k$ monomials from $\varphi$ with fewer than $g$ satisfying assignments each are not covered. Hence, the cardinality of the error region of $h$ is bounded as

$$|\,\mathtt{sat}(\varphi) \setminus \mathtt{sat}(h)| \;\leq\; k\,g \;=\; \frac{\varepsilon}{2q}\,|\,\mathtt{sat}(\varphi)|$$

and the error of $h$ can be estimated as follows:

$$\mathsf{error}_{\mathsf{rel}}(h) \;\leq\; q \cdot |\,\mathtt{sat}(\varphi) \setminus \mathtt{sat}(h)| \,/\, |\,\mathtt{sat}(\varphi)| \;\leq\; \varepsilon/2\;. \tag{3.3}$$

We call a hypothesis $h \in \mathcal{H}_\mathcal{M}$ *bad* if $\mathsf{error}_{\mathsf{rel}}(h) > \varepsilon$. Note that some hypotheses of $\mathcal{H}_\mathcal{M}$ may be neither good nor bad. Let us analyze the probability that `Learn-`$k$`-Term-DNF` returns a bad hypothesis (either in Line 14 or in Line 15) despite $\mathcal{H}_\mathcal{M}$ contains at least one good hypothesis.

**Case 1:** A good hypothesis $h$ is not satisfied by enough examples in $S$ to be selected as a hypothesis.
The probability $p_1$ that this happens can be bounded by Proposition 2.4 of

Angluin and Valiant [3] derived from Chernoff's bound. For $\beta = \varepsilon/(4-2\varepsilon)$, we can bound $p_1$ by the following inequality:

$$\begin{aligned} p_1 &\leq \exp\left(-\frac{\varepsilon^2 n_2\,(1-\frac{\varepsilon}{2})}{2(4-2\varepsilon)^2}\right) \\ &= \exp\left(-\frac{\varepsilon^2\frac{48}{\varepsilon^2}(k^2\ln(2)\ +k\ln(n_1)+\ln(\frac{4}{\delta}))\,(1-\frac{\varepsilon}{2})}{32\,(1-\frac{\varepsilon}{2})^2}\right) \\ &= \left(\frac{\delta}{4\cdot(n_1)^k\cdot 2^{k^2}}\right)^{3/(2-\varepsilon)} . \end{aligned}$$

Since $0<\varepsilon<1$ implies $3/(2-\varepsilon)>1$ and due to $0<\delta<1$ we get $p_1<\delta/4$.

**Case 2:** In Line 14, the learner chooses a bad hypothesis $h' \in \mathcal{H}_\mathcal{M}$ before reaching a good hypothesis $h$, it returns $h'$, and terminates.
The probability $p_2$ that $h'$ is satisfied by enough examples to be accepted as a resulting hypothesis can be bounded similarly to $p_1$. For $\beta = \varepsilon/(4-4\varepsilon)$, one can bound $p_2$ using the following inequality:

$$\begin{aligned} p_2 &\leq \exp\left(-\frac{\varepsilon^2 n_2(1-\varepsilon)}{3(4-4\varepsilon)^2}\right) \\ &= \exp\left(-\frac{\varepsilon^2\frac{48}{\varepsilon^2}(k^2\ln(2)\ +k\ln(n_1)+\ln(\frac{4}{\delta}))(1-\varepsilon)}{48(1-\varepsilon)^2}\right) \\ &= \left(\frac{\delta}{4\cdot(n_1)^k\cdot 2^{k^2}}\right)^{1/(1-\varepsilon)} . \end{aligned}$$

Since $1/(1-\varepsilon)>1$ and $0<\delta<1$, using the estimation (3.2), we get $p_2 < |\mathcal{H}_\mathcal{M}|^{-1}\,\delta/4$.

Thus, assuming a good hypothesis $h \in \mathcal{H}_\mathcal{M}$ exists, the probability that the learner neither rejects $h$ and finally outputs a (probably bad) default hypothesis in Line 15 nor outputs a bad hypothesis in Line 14 is bounded from below by

$$1-(p_1+|\mathcal{H}_\mathcal{M}|\,p_2)\ >\ 1-\delta/2\ .$$

In this case, the error of the hypothesis $\varphi'$ returned by the learner is bounded by $\mathsf{error}_{\mathsf{rel}}(\varphi') \leq \varepsilon$. According to Proposition 3.11, the probability that the learner computes a set $\mathcal{M}$ such that $\mathcal{H}_\mathcal{M}$ contains at least one good hypothesis is at least $(1-\delta/2)$. So, with probability at least $1-\delta$, the hypothesis generated by `Learn-`$k$`-Term-DNF` satisfies the error bound $\varepsilon$.

It is left to show that for constant $k$ the algorithm runs in time polynomial in $d$, $q$, $1/\varepsilon$, and $\log 1/\delta$. The number of examples $\sigma_k(d,q,\varepsilon,\delta)$ fulfills this bound. The $k$-CNF formula can be learned in polynomial time with respect to all parameters except $k$. The algorithm computes at most $2^k-1$ maximal monomials for each example. By Lemma 3.9 this can be done in polynomial time. The size of the

hypothesis space for the sequential test is $|\mathcal{H}_{\mathcal{M}}| \leq (n_1)^k \cdot 2^{k^2}$. The test itself can be performed in polynomial time. □

## 3.7 Infeasibility for Unrestricted DNF Formulas

Verbeurgt [64] has developed a method for learning poly($d$)-term DNF over the uniform distribution from a polynomial number of positive and negative examples with a quasi-polynomial running time. In contrast, excluding false positives it can be shown:

**Theorem 3.19.** *For every hypothesis space $\mathcal{H}$ and sufficiently small error parameters $\varepsilon, \delta$, learning $d$-term DNF formulas weakly without false positives requires an exponential number of positive examples. This even holds when fixing the family of possible distributions for every $d$ to a single arbitrary $q$-bounded distribution.*

*Proof.* Kearns et al. [39] have shown for the uniform distribution that the number of positive examples needed to learn (monotone) clauses weakly without false positives is $\Omega(2^{d/4})$ for sufficiently small error parameters $\varepsilon, \delta$, independent of the hypothesis space. The class $d$-term DNF in particular contains all nontrivial clauses, thus the number of examples needed to learn $d$-term DNF weakly without false positives cannot be smaller for the uniform distribution.

This lower bound immediately applies if one requires successful learning for every $q$-bounded distribution since the uniform distribution belongs to this class. But what happens if we consider only a single fixed distribution other than the uniform one? Being the *simplest* $q$-bounded distribution, one would expect that the lower bound easily translates to every other $q$-bounded distribution. However, the formal proof of [39] uses the symmetry of the uniform distribution explicitly. Thus, it is not obvious whether it could be extended. Furthermore, this lower bound cannot hold for every distribution, for example not for distributions with a support of polynomial size. In the following we generalize the proof technique to the case that the learner has to deal with a single $q$-bounded distribution only and this may even be known.

Let $q \geq 1$ and $D$ be an arbitrary $q$-bounded distribution over $\{0,1\}^d$, thus $q^{-1} \, |S| \, 2^{-d} \leq D(S) \leq q \, |S| \, 2^{-d}$ for every subset $S \subseteq \{0,1\}^d$. To keep the calculation simple we may assume that $d$ is even. Let

$$\mathcal{C}_d = \{\mathtt{sat}(\varphi) : \varphi \text{ is a monotone clause with at least } d/2 \text{ variables}\}$$

and $c \in \mathcal{C}_d$. Then $|\mathcal{C}_d| = 2^{d-1}$ and $D(\overline{c}) \leq q \cdot 2^{-d/2}$. Such a concept $c$ can uniquely be associated with an assignment $u_c \in \{0,1\}^d$ where the $i$-th bit of $u_c$ is 0 if and only if the clause representing $c$ contains the variable $x_i$. Obviously, $u_c \notin c$, thus $u_c$ cannot be included in any hypothesis for $c$ if false positives are not allowed.

It will be shown that the lower bound even holds when restricting the concept class to $\mathcal{C}_d$. For a contradiction suppose there exists an algorithm $\mathcal{A}$ which learns $\mathcal{C}_d$ from positive examples strongly without false positives with respect to $D$ such that the number of training examples used by $\mathcal{A}$ is bounded by $n = 2^{d/4}$. The case of learning weakly without false positives will be discussed at the end of this proof.

Let $\Psi$ be the set of all sample sequences $T$ over $\{0,1\}^d$ of length $n$. A randomly drawn $T \in \Psi$ may contain negative examples for a clause $c \in \mathcal{C}_d$, but this happens with probability at most $\pi(d,q,n) = n \cdot q \cdot 2^{-d/2}$. We denote this property by $T \not\subseteq c$ and its complement by $T \subseteq c$. Let $\Psi_c := \{T \in \Psi : T \subseteq c\}$, then $\Pr[\Psi_c] \geq 1 - \pi(d,q,n)$.

By $[u_c \in \mathcal{A}(T)]$ we denote the event that the hypothesis $\mathcal{A}(T)$ generated by $\mathcal{A}$ on sample sequence $T$ contains the negative example $u_c$. It is not determined how $\mathcal{A}$ behaves on *illegal* sequences $T \notin \Psi_c$, but if false positives are strongly not allowed for correct sequences, we can estimate

$$\begin{aligned} \Pr_{T\in\Psi}[u_c \in \mathcal{A}(T)] &= \Pr_{T\in\Psi}[u_c \in \mathcal{A}(T) \wedge T \in \Psi_c] \\ &\quad + \Pr_{T\in\Psi}[u_c \in \mathcal{A}(T) \wedge T \notin \Psi_c] \\ &\leq 0 + \Pr[T \notin \Psi_c] \leq \pi(d,q,n)\ . \end{aligned} \tag{3.4}$$

Since this holds for every $c \in \mathcal{C}_d$ summing up gives

$$\sum_{c\in\mathcal{C}_d} \sum_{T\in\Psi} \Pr[T] \cdot \Pr[u_c \in \mathcal{A}(T)] \leq 2^{d-1}\, \pi(d,q,n)$$

or exchanging the summation

$$\sum_{T\in\Psi} \Pr[T] \sum_{c\in\mathcal{C}_d} \Pr[u_c \in \mathcal{A}(T)] \leq 2^{d-1}\, \pi(d,q,n)\ .$$

Define

$$\Psi^+ := \{T \in \Psi : \sum_{c\in\mathcal{C}_d} \Pr[u_c \in \mathcal{A}(T)] \geq |C_d|/2\}$$

and $\Psi^- := \Psi \setminus \Psi^+$. Then

$$\sum_{T\in\Psi^+} \Pr[T] \leq 2\, \pi(d,q,n)\ ,$$

since otherwise

$$\sum_{T\in\Psi} \Pr[T] \sum_{c\in\mathcal{C}_d} \Pr[u_c \in \mathcal{A}(T)] > 2\, \pi(d,q,n) \cdot |C_d|/2 = 2^{d-1}\, \pi(d,q,n)\ .$$

For $T \in \Psi^-$, the following holds:

$$\begin{aligned} \sum_{c\in\mathcal{C}_d} \Pr[u_c \notin \mathcal{A}(T)] &= \sum_{c\in\mathcal{C}_d} (1 - \Pr[u_c \in \mathcal{A}(T)]) = |C_d| - \sum_{c\in\mathcal{C}_d} \Pr[u_c \in \mathcal{A}(T)] \\ &> |C_d|/2\ . \end{aligned}$$

Since the $u_c$ are all different elements of $\{0,1\}^d$ this implies that the expected size of the complement of $\mathcal{A}(T)$ is at least $|C_d|/2$ for every $T \in \Psi^-$. Then also the probability that $|\overline{\mathcal{A}(T)}|$ is not too small has to achieve a certain minimal level, more precisely

$$\Pr[\ |\overline{\mathcal{A}(T)}| \ \leq \ 2^{d-3}] \ < \ 7/8$$

since otherwise

$$\mathrm{E}[\ |\overline{\mathcal{A}(T)}|\ ] \leq \frac{7}{8}\ 2^{d-3} + \frac{1}{8}\ 2^d \ < \ |C_d|/2\ .$$

Thus, the probability that a sample sequence belongs to $\Psi^-$ is at least $1 - 2\ \pi(d,q,n)$ and for $T \in \Psi^-$ with probability at least $1/8$ the hypothesis $\mathcal{A}(T)$ excludes at least $2^{d-3}$ assignments $u_c$ with $c \in \mathcal{C}_d$. For every $c$ holds

$$\Pr[\Psi_c \cap \Psi^-] \ \geq \ \Pr[\Psi_c] - \Pr[\Psi^+] \ \geq \ 1 - 3\ \pi(d,q,n)\ .$$

Since $c$ has at most $2^{d/2}$ negative assignments, on examples in $T \in \Psi_c \cap \Psi^-$ with probability at least $1/8$ at least $2^{d-3} - 2^{d/2}$ elements in $\{0,1\}^d$ are misclassified by $\mathcal{A}(T)$. Thus, for $d \geq 8$ with probability at least $1/8 - 3\ \pi(d,q,n) = 1/8 - 3\ q\ 2^{-d/4}$ the hypothesis $\mathcal{A}(T)$ yields an error

$$\mathsf{error}_{\mathsf{rel}}(\mathcal{A}(T)) \ = \ \frac{D(c \setminus \mathcal{A}(T))}{D(c)} \ \geq \ D(c \setminus \mathcal{A}(T)) \ \geq \ q^{-1}\ \frac{2^{d-3} - 2^{d/2}}{2^d} \ \geq \ (16q)^{-1}\ .$$

The value $1/8 - 3\ q\ 2^{-d/4}$ can be made larger than any error $\delta < 1/8$ by choosing $d$ large enough.

Looking at the estimation (3.4) one can notice that the condition *strongly without false positives* can be relaxed by requiring $\Pr_{T \in \Psi}[u_c \in \mathcal{A}(T) \wedge T \in \Psi_c] \ \leq \ o(1)$ for every $c$, which allows to prove the statement for the weak case as well. □

Summarizing, a polynomial-time algorithm for learning $k$-term DNF formulas properly from positive samples alone for every fixed $k$ has been presented in this chapter. On the other hand, we have seen that due to information theoretical reasons $k$-term DNF formulas cannot be learned from positive samples weakly without false positives if $k$ grows linearly with the number of variables.

# 4 Learning Residual Alternating Automata

Regular languages can be described in different ways, commonly either by regular grammars or by a finite automata. There are different types of finite automata (FA) to describe regular languages. They all have the same expressive power, but differ in succinctness and efficiency when solving automata-related problems: starting with deterministic finite automata (DFA), over their generalization to nondeterministic ones (NFA) and the dual of NFAs, universal finite automata (UFA), up to alternating FAs (AFA), which are the common generalization of NFAs and UFAs. A *minimal* (measured by the number of states) DFA might be exponentially larger than an NFA and double-exponentially larger than an AFA. Since the membership problem for AFAs remains solvable efficiently [17], one may want to work with these more succinct automata.

Learning of regular languages has great relevance for practice applications. To name a few: model checking, pattern recognition, robot navigation, automated verification, and many others [22]. However, regular languages are hard to learn. Independent of their representation, deterministic finite automata (DFAs) cannot be PAC learned efficiently unless the RSA cryptosystem is not secure [41]. Positive results for the PAC setting can be obtained if the concept class is restricted to subclasses like e. g. Terminal Distinguishable Regular Languages (TDRL). A TDRL is produced by a regular grammar that is backward deterministic and fulfills particular constraints regarding discriminability of production rules by disjoint sets of terminal symbols. The class of TDRL can be PAC learned in the distribution-free setting [13].

In contrast, when learning from membership queries and counterexamples, general DFAs can be learned efficiently and without restrictions. The learning algorithm for DFAs, called $\mathtt{L}^\star$, has been provided by Angluin in [5]. It learns the *minimal* DFA, which is unique. The time and query complexity is bounded by a polynomial with respect to the size of the minimal equivalent DFA.

Yokomori [66] has constructed a learning algorithm for NFAs. Unfortunately, in some cases the NFAs produced by his learning algorithm may even be larger than the minimal DFA, i. e. than the output of $\mathtt{L}^\star$ [42]. A generalization of $\mathtt{L}^\star$ to NFAs that produces more compact automata has been proposed by Bollig et al. [15]. Their $\mathtt{NL}^\star$ algorithm builds upon the following observation. Every DFA has a useful property called *residuality*. An automaton $\mathfrak{A}$ accepting a language $L$ is residual if every state $q$ of $\mathfrak{A}$ corresponds to a residual language $w_q^{-1}L := \{v \in \Sigma^* : w_q v \in L\}$ of $L$. Denis, Lemay, and Terlutte [26] introduced

the class of residual NFA (RNFA). In a later work [27], they have presented a learning algorithm for RNFAs in the limit from an informant, a famous learning model introduced by Gold [35]. While the minimal NFA (or UFA, resp.) is not unique, the minimal RNFA (or RUFA) with a maximum number of transitions is unique. It is called *canonical*. Using the residuality property, Bollig et al. have created `NL*` that learns the canonical RNFA with a polynomial number of membership and equivalence queries with respect to the size of the minimal equivalent DFA. The algorithm can be adapted to learn the dual, residual universal FAs (RUFA). This variation, called `UL*`, is described by Kern [42].

Both algorithms, `NL*` and `UL*`, are implemented in the `libalf` learning library [16]. In the area of formal verification, among others learning algorithms for automata from `libalf` have been used for the compositional verification of probabilistic systems [31, 32] and verification and model synthesis of sequential programs [20].

Angluin, Eisenstat, and Fisman [7] extended the definition of residual automata to residual alternating finite automata (RAFA) and provided `AL*`, a learning algorithm for AFAs. Based on experiments with randomly generated languages, the authors conjectured that `AL*` always outputs residual AFAs, but they were not able to give a proof. Although no natural definition of a minimal unique RAFA (i. e. a canonical RAFA) is known yet, residuality is still desirable when learning AFAs, because residual FAs contain a certain structure that may be helpful for later use in practical applications, especially for involved automata. The states of a residual automaton represent specific semantics of the language, which may simplify the analysis of the automaton.

In this chapter we disprove the conjecture of Angluin et al. by providing a carefully designed counterexample. Afterwards, we construct an efficient learning algorithm, named `AL**`, and give a proof that `AL**` outputs residual AFAs only. Finally, we investigate the succinctness of different FA types.

## 4.1 On Automata and Regular Languages

The model of alternating finite automata has been introduced by Chandra, Kozen, and Stockmeyer [17]. Let the symmetric difference of sets be denoted by $\triangle$, the set of all suffixes of a string $w$ denoted by $\mathrm{Suffs}(w)$, the empty string as $\lambda$, and the Boolean values "true" as $\top$ and "false" as $\bot$. For a set $S$ let $\mathcal{F}(S)$ be the set of all formulas over $S$ using the binary operators $\wedge$ and $\vee$ plus the trivial formulas $\top$ and $\bot$ that are always, respectively never satisfied. The restriction $\mathcal{F}_\vee(S)$, respectively $\mathcal{F}_\wedge(S)$ denotes the subset of formulas containing only the $\vee$ operator plus the formula $\bot$ (respectively only $\wedge$ and $\top$).

**Definition 4.1.** *Given a finite alphabet $\Sigma$, an* alternating finite automaton (AFA) *is a four-tuple $(Q, Q_0, F, \tau)$, where $Q$ is the* set of states*, $Q_0 \in \mathcal{F}(Q)$ the* initial configuration*, $F \subseteq Q$ the subset of* accepting states*, and $\tau\colon Q \times \Sigma \to \mathcal{F}(Q)$ the* transition function*.*

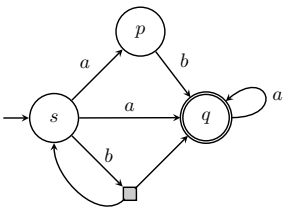


Figure 4.1: An AFA for the language $L_1 = a^+ \cup ba^+ \cup aba^*$.
The initial configuration is $Q_0 = s$ and the set of accepting states is $F = \{q\}$.
From state $s$ the automaton has the transitions $\tau(s, a) = p \vee q$ and $\tau(s, b) = s \wedge q$.

*If $Q_0$ and, for all $q \in Q$ and all $a \in \Sigma$, the transition $\tau(q, a)$ consist of a single state then the automaton is called* deterministic (DFA). *If $Q_0 \in \mathcal{F}_\vee(Q)$ and $\tau(q, a) \in \mathcal{F}_\vee(Q)$ for all $q \in Q$ and all $a \in \Sigma$, it models a* nondeterministic automaton (NFA). *If $Q_0 \in \mathcal{F}_\wedge(Q)$ and $\tau(q, a) \in \mathcal{F}_\wedge(Q)$ for all $q \in Q$ and all $a \in \Sigma$, the automaton is called* universal (UFA).

A transition $\tau(q, a)$ of an AFA can be a nested formula of $\vee$ and $\wedge$ operators. Such a formula may be difficult to illustrate graphically. However, it can equivalently be represented by its monotone disjunctive normal form (MDNF). Each monomial in such an MDNF is represented by an edge from $q$ marked with the symbol $a$ leading to a little square. From this square we draw edges to all states that are contained in this monomial. If the monomial consists of a single state only the square can be omitted. For an example, see the AFA in Fig. 4.1.

The function $\tau$ is extended to arbitrary formulas $\varphi \in \mathcal{F}(Q)$ and strings $w \in \Sigma^*$ as follows. Let $\varphi_{\text{MDNF}} = \bigvee_i M_i$ with $M_i = \bigwedge_j q_{i,j}$ be an MDNF formula equivalent to $\varphi$. Then $\tau(\varphi, a) := \bigvee_i \bigwedge_j \tau(q_{i,j}, a)$ for a single symbol $a \in \Sigma$ and $\tau(\varphi, \lambda) := \varphi$ for the empty string $\lambda$. For $w \in \Sigma^+$, we define $\tau(\varphi, wa) := \tau(\tau(\varphi, w), a)$. For an NFA, this simply reduces to $\tau(q \vee p, a) = \tau(q, a) \vee \tau(q, b)$.

**Definition 4.2.** *For an AFA $\mathfrak{A} = (Q, Q_0, F, \tau)$ and a formula $\varphi \in \mathcal{F}(Q)$, we define the* evaluation *of $\varphi$, denoted as $(\!|\varphi|\!)$, recursively as follows: $(\!|\top|\!) := \top$ and $(\!|\bot|\!) := \bot$. For singletons let $(\!|q|\!) := \top$ if $q \in F$ and equal $\bot$ otherwise. Finally, $(\!|\varphi R \psi|\!) := (\!|\varphi|\!)\ R\ (\!|\psi|\!)$ for $R \in \{\wedge, \vee\}$.*
*The automaton $\mathfrak{A}$* accepts *a word $w$ if $(\!|\tau(Q_0, w)|\!) = \top$. The* language *$L(\mathfrak{A})$ is the set of all accepted strings. For a state $q \in Q$, we write $\mathfrak{A}_q$ for the automaton $(Q, q, F, \tau)$ that starts in configuration $q$ instead of $Q_0$.*

For an NFA with $\tau(Q_0, w) = q_1 \vee \ldots \vee q_k$ the evaluation $(\!|\tau(Q_0, w)|\!) = \top$ corresponds to the usual condition $\{q_1, \ldots, q_k\} \cap F \neq \emptyset$, i. e. when starting with initial configuration $Q_0$ and reading the word $w$ some accepting state is reached. For a UFA with $\tau(Q_0, w) = q_1 \wedge \ldots \wedge q_k$ it requires $\{q_1, \ldots, q_k\} \subseteq F$, i. e. all states reached are accepting.

**Definition 4.3.** *Let $L \subseteq \Sigma^*$ be a regular language.*

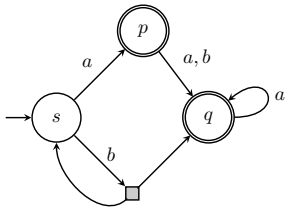


Figure 4.2: A residual AFA (RAFA) for the language $L_1 = a^+ \cup ba^+ \cup aba^*$. State $s$ corresponds to $\lambda^{-1}L = a^+ \cup ba^+ \cup aba^*$, state $p$ to $a^{-1}L = a^* \cup ba^*$, and state $q$ to $(ab)^{-1}L = a^*$.
Note that these residual languages $\lambda^{-1}L$, $a^{-1}L$ and $(ab)^{-1}L$ are both $\cup$-prime and $\cap$-prime.

- *For a word $u \in \Sigma^*$, we define the* residual language $u^{-1}L$ *as* $\{v \in \Sigma^* : uv \in L\}$.
- *The set of all residual languages of language $L$ is denoted by* $\mathrm{RES}(L)$.
- *A residual language $u^{-1}L$ is called* $\cup$-prime, *respectively* $\cap$-prime *if $u^{-1}L$ cannot be defined as the union, respectively intersection of other residual languages. We denote the set of all $\cup$-prime, respectively $\cap$-prime residuals of $L$ by $\cup$-Primes$(L)$, respectively $\cap$-Primes$(L)$.*
- *An automaton $\mathfrak{A}$ with states $Q$ is* residual, *if $L(\mathfrak{A}_q) \in \mathrm{RES}(L(\mathfrak{A}))$ for all $q \in Q$, i. e. if every state corresponds to a prefix $u$ and its residual language $u^{-1}L(\mathfrak{A})$.*
- *Let RNFA, RUFA and RAFA denote the appropriate residual restrictions.*

For an example, see the residual AFA in Fig. 4.2 that accepts the same language $L_1 = a^+ \cup ba^+ \cup aba^*$ as the nonresidual AFA illustrated in Fig. 4.1

## 4.2 Learning Automata

Given an alphabet $\Sigma$, let the learning domain be $\mathcal{X}_d = \Sigma^*$ for every dimension $d \in \mathbb{N}$. We define the concept class of regular languages dependent on $d$ as

$$\begin{aligned}\mathcal{C}_d := \{L \subseteq \Sigma^* : {} & L \text{ is a regular language,} \\ & \text{the minimal DFA } \mathfrak{A} \text{ with } L(\mathfrak{A}) = L \text{ has at most } d \text{ states, and} \\ & \text{the minimal DFA } \hat{\mathfrak{A}} \text{ with } L(\hat{\mathfrak{A}}) = L^{\mathrm{rev}} \text{ has at most } d \text{ states}\} \ ,\end{aligned}$$

where $L^{\mathrm{rev}}$ denotes the reverse language

$$L^{\mathrm{rev}} := \{w \in \Sigma^* : \text{the reverse string of } w \text{ is in } L\} \ .$$

We study perfect learning from membership and equivalence queries of $\mathcal{C} = \{\mathcal{C}_d : d \in N\}$. The distribution $D$ over $\mathcal{X}_d$ is not used by the oracles in this setting. However, in order to provide efficient learning algorithms we assume that the maximal length of a counterexample presented by the $EQUIV$oracle is bounded

by a polynomial in the dimension $d$[1]. Since the distribution $D$ affects the error measure, we consider the distribution-free setting to ensure that perfect learning implies the equivalence of the target language $L$ and the language $\mathfrak{A}(L)$ accepted by the automaton $\mathfrak{A}$ output by the learner.

We require the hypothesis to be a specific type of automaton, e. g. minimal DFA or canonical RNFA, and so on. This is an even stronger requirement than proper learning, because for proper learning it is sufficient that $\mathcal{H}_d = \mathcal{C}_d$ for every $d \in \mathbb{N}$, which is already guaranteed by perfect learning.

Now that we have defined the setting, we present the known learning algorithms $\texttt{XL}^\star$ for automata (i. e. $\texttt{L}^\star$, $\texttt{NL}^\star$, $\texttt{UL}^\star$, and $\texttt{AL}^\star$). All these algorithms follow a similar pattern, and the new $\texttt{AL}^{\star\star}$ also does. Two sets $U, V \subseteq \Sigma^*$ are constructed, where $U$ is prefix-closed and $V$ is suffix-closed. For all strings $uv \in UV$ or $uav \in U\Sigma V$ a membership query is performed. The resulting matrix, indexed by $U \cup U\Sigma$ and $V$ is called a *table*. The rows indexed by $U$ correspond to possible states. To minimize the number of states, a subset $P$ of rows (a *basis*) is constructed such that all rows can be built based on the elements of $P$. The specific way to build a row depends on the type of automaton. A hypothesized automaton is constructed from this subset $P$. For a row $r_u$ indexed by $u \in U$ and a symbol $a \in \Sigma$, the transition $\tau(r_u, a)$ equals the formula that builds the row indexed by $ua$. For this purpose, similar to Bollig et al. [15] we introduce the following notion.

**Definition 4.4.** *Let $L$ be a regular language. For a prefix-closed set $U$ and a suffix-closed set $V$, a $|U \cup U\Sigma| \times |V|$* table *$\mathcal{T} = (T, U, V)$ for $L$ with entries in $\{+, -\}$ is determined by a function $T\colon \Sigma^* \to \{+, -, \bot\}$ specified as follows. Let $W(\mathcal{T})$ denote the set $(U \cup U\Sigma)\,V$ described by $\mathcal{T}$. Then for $w \in \Sigma^*$*

$$T(w) = \begin{cases} \bot & \text{if } w \notin W(\mathcal{T}), \\ + & \text{if } w \in W(\mathcal{T}) \cap L, \\ - & \text{if } w \in W(\mathcal{T}) \setminus L. \end{cases}$$

*The entry of $\mathcal{T}$ in row $x$ and column $y$ is equal to $T(xy)$.*

Note that to define $\mathcal{T}$ we only need values $T$ on $W(\mathcal{T})$. We extend the domain of $T$ to all words over $\Sigma$ for the sake of completeness. An example of a table is given in Fig. 4.3.

**Definition 4.5.** *Let $L$ be a regular language and $\mathcal{T} = (T, U, V)$ be a table for $L$.*

- *An automaton $\mathfrak{A}$ and a table $\mathcal{T}$ are called* compatible *if for every $w \in W(\mathcal{T})$ holds: $\mathfrak{A}$ accepts $w$ if and only if $T(w) = +$.*

[1] If the automaton presented to the *EQUIV*oracle is huge, the shortest counterexample may become very long, too. Since all learning algorithms try to generate automata that are as small as possible, this case will not occur as we will see later. Thus we ignore this issue here.

| | | $V$: $\lambda$ | $ab$ | $b$ |
|---|---|---|---|---|
| $U$ | $\lambda$ | $-$ | $+$ | $-$ |
| | $a$ | $-$ | $-$ | $+$ |
| $R$ | $b$ | $-$ | $-$ | $-$ |
| | $aa$ | $-$ | $-$ | $-$ |
| | $ab$ | $+$ | $-$ | $+$ |

Figure 4.3: Table $\mathcal{T} = (T, U, V)$ for the language $L = ab^+$, with $U = \{\lambda, a\}$, $V = \{\lambda, ab, b\}$, and $R = U\Sigma \setminus U = \{b, aa, ab\}$. The entries of the table are determined by $T$: the value in row $x$ and column $y$ equals $T(xy)$. For example, the value in row $ab$ and column $b$ is $+$ since $T(abb) = +$ ($abb \in L$) and $abb \in W(\mathcal{T})$. An example for a row is $r_\lambda = (- + -)$. Furthermore, $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T}) = \{r_\lambda, r_a\}$.

- *For every $u \in U \cup U\Sigma$ we associate a vector $r_u$ of length $|V|$ over $\{+, -\}$ with $r_u[v] = T(uv)$ for $v \in V$, called the* row *of $u$. The set of all rows is denoted by* $\mathrm{Rows}(\mathcal{T})$ *and the subset of those $r_u$ with $u \in U$ by* $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$.
- *A table $\mathcal{T}$ is* consistent *if for every $u, u' \in U$ with $r_u = r_{u'}$ the condition $r_{ua} = r_{u'a}$ is fulfilled for every $a \in \Sigma$.*[2]
- *To simplify the notation, for a consistent table $\mathcal{T}$, row $r \in \mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$, and symbol $a \in \Sigma$, let $ra$ denote the vector $r_{ua}$ where $u \in U$ is an arbitrary string with $r_u = r$.*

The order $- \leq +$ on the set $\{+, -\}$ is extended to a partial order on vectors by requiring $\leq$ to hold for each component. The binary operators $\sqcap, \sqcup$ on the set $\{+, -\}$ are defined by $a \sqcap b = \min\{a, b\}$ and $a \sqcup b = \max\{a, b\}$. For vectors, these operators are extended by performing the operation componentwise.

For a formula $\varphi \in \mathcal{F}(\mathrm{Rows}(\mathcal{T}))$ on the rows of a table, we define the evaluation $[\![\varphi]\!]$ by $[\![\top]\!] = +^{|V|} = + \cdots +$, $[\![\bot]\!] = -^{|V|} = - \cdots -$, $[\![r_u]\!] = r_u$, $[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \sqcap [\![\psi]\!]$ and $[\![\varphi \vee \psi]\!] = [\![\varphi]\!] \sqcup [\![\psi]\!]$ and extend this to a set $P$ of formulas by $[\![P]\!] = \{[\![\varphi]\!] : \varphi \in P\}$. For example,

$$[\![(+ - + \wedge - - +) \vee - + -]\!] = - + +.$$

**Definition 4.6.** *A table $\mathcal{T}$ is called*

- D-closed *if* $\mathrm{Rows}(\mathcal{T}) \subseteq \mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$,
- N-closed *if* $\mathrm{Rows}(\mathcal{T}) \subseteq [\![\mathcal{F}_\vee(\mathrm{Rows}_{\mathrm{high}}(\mathcal{T}))]\!]$,
- U-closed *if* $\mathrm{Rows}(\mathcal{T}) \subseteq [\![\mathcal{F}_\wedge(\mathrm{Rows}_{\mathrm{high}}(\mathcal{T}))]\!]$, *and*
- A-closed *if* $\mathrm{Rows}(\mathcal{T}) \subseteq [\![\mathcal{F}(\mathrm{Rows}_{\mathrm{high}}(\mathcal{T}))]\!]$.

[2] This is a weaker requirement than the *RFSA-consistency* of [15], which requires that $r_u \leq r_{u'}$ implies $r_{ua} \leq r_{u'}a$.

Algorithm 3 shows the general form of an $\texttt{XL}^\star$ algorithm, i. e. $\texttt{L}^\star$, $\texttt{NL}^\star$, $\texttt{UL}^\star$, and $\texttt{AL}^\star$. The construction of the XFA $\mathfrak{A}(\mathcal{T})$ from a table $\mathcal{T}$ will be presented in the following subsections, as well as a discussion of the individual properties of all the $\texttt{XL}^\star$ algorithms each.

**Input:** type of automaton X, membership oracle $MEMBER(L)$ and equivalence oracle $EQUIV(L)$ respecting the unknown target language $L \subseteq \Sigma^*$
**Output:** an XFA $\mathfrak{A}^{\mathrm{X}}$

```
1  U ← {λ}; V ← {λ};
2  initialize T = (T, U, V) with |Σ| + 1 membership queries;
3  while true do
4      while T is not X-closed do
5          find a row r_ua ∈ Rows(T) such that r_ua violates the inclusion for
             X-closedness in Definition 4.6;
6          add ua to U;
7          complete T via membership queries;
8      construct the XFA A^X(T);
9      if L(A^X(T)) = L then
10         return A^X(T);
11     else
12         get a counterexample w ∈ L △ L(A^X(T));
13         set V ← V ∪ Suffs(w);
14         complete T via membership queries;
```

**Algorithm 3:** $\texttt{XL}^\star$. For $\texttt{L}^\star$ we set $\mathrm{X} = \mathrm{D}$.

We start the analysis of $\texttt{XL}^\star$ with the following observation, which allows to assume consistency of $\mathcal{T}$ within the constructions of the particular XFAs.

**Lemma 4.7.** *Every table $\mathcal{T}$ constructed by $\texttt{XL}^\star$ is consistent.*

*Proof.* Let $\mathcal{T} = (T, U, V)$ be any table constructed by $\texttt{XL}^\star$. It suffices to show that for different $u, u' \in U$ the rows $r_u \neq r_{u'}$ are different, too. Then the precondition for the consistency requirement, namely equal rows, is never fulfilled, and consistency holds trivially. Assume that $u'$ has been added to $U$ after $u$. This can only happen if the X-closedness condition is violated in Line 4. This, however, contradicts $r_{u'} = r_u \in \mathrm{Rows}_{\mathrm{high}}\,\mathcal{T}$. □

### 4.2.1 Learning Deterministic Automata

The construction of the DFA by $\texttt{L}^\star$ is as follows.

**Definition 4.8.** *Let $\mathcal{T}$ be a consistent and D-closed table. The DFA $\mathfrak{A}^{\mathrm{D}}(\mathcal{T}) = (Q, Q_0, F, \tau)$ consists of the following components: $Q = \mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$, $Q_0 = r_\lambda$ and $F = \{r \in Q : r[\lambda] = +\}$. For $r \in Q$ and $a \in \Sigma$ let $\tau(r, a) = ra$.*

| | $\lambda$ | $a$ | $ba$ |
|---|---|---|---|
| $\lambda$ | $-$ | $+$ | $-$ |
| $a$ | $+$ | $-$ | $-$ |
| $b$ | $-$ | $-$ | $-$ |
| $aa$ | $-$ | $+$ | $-$ |
| $ab$ | $-$ | $-$ | $-$ |
| $ba$ | $-$ | $-$ | $-$ |
| $bb$ | $-$ | $-$ | $-$ |

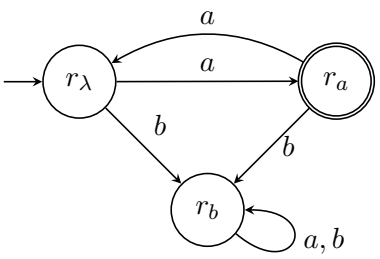


Figure 4.4: A consistent and D-closed table $\mathcal{T}$ and the corresponding DFA $\mathfrak{A}^{\mathrm{D}}(\mathcal{T})$.

For an example of this construction see Fig. 4.4.

In the original $\mathtt{L}^\star$ algorithm proposed by Angluin [5], a counterexample $w$ and its prefixes obtained from the equivalence oracle are added to $U$. For this original version of the algorithm, Lemma 4.7 does not hold. The set $V$ is then extended whenever the consistency criterion is violated. The version of $\mathtt{L}^\star$ presented in Algorithm 3 adds a counterexample $w$ and its suffixes in Line 13 to $V$ instead. This modification has been suggested by Maler and Pnueli originally [46].

**Theorem 4.9** (Angluin [5])**.** *For every regular language $L \in \mathcal{C}_d$, the algorithm $\mathtt{L}^\star$ always generates the minimal DFA $\mathfrak{A}^{\mathrm{D}}$ such that $L(\mathfrak{A}^{\mathrm{D}}) = L$. The algorithm runs in time bounded by a polynomial in $d$, $|\Sigma|$, and $\ell$, where $\ell$ is the size of the longest counterexample obtained from the equivalence oracle.*

### 4.2.2 Learning Residual Nondeterministic and Universal Automata

The following definition helps to conclude that $\mathtt{NL}^\star$ and $\mathtt{UL}^\star$ always learn a unique minimal RNFA or RUFA, respectively. Canonical RNFAs have been introduced by [26]. The definition has been adapted to canonical RUFAs by [42].

**Definition 4.10** (Canonical RNFA and canonical RUFA)**.** *Let $L$ be a regular language.*

- *The canonical RNFA for $L$ is the tuple $(Q, Q_0, F, \tau)$ with $Q = \cup$-Primes$(L)$, $Q_0 = \{L' \in Q : L' \subseteq L\}$, $F = \{L' \in Q : \lambda \in L'\}$, and $\tau(L_1, a) = \{L_2 \in Q : L_2 \subseteq a^{-1}L_1\}$.*
- *The canonical RUFA for $L$ is the tuple $(Q, Q_0, F, \tau)$ with $Q = \cap$-Primes$(L)$, $Q_0 = \{L' \in Q : L \subseteq L'\}$, $F = \{L' \in Q : \lambda \in L'\}$, and $\tau(L_1, a) = \{L_2 \in Q : a^{-1}L_1 \subseteq L_2\}$.*

The canonical RNFA and RUFA have the minimal number of states and the maximal number of transitions between these states, which makes them unique. Figure 4.5 shows the canonical RUFA for the same language $L_1$ as used in Fig. 4.1 and 4.2.

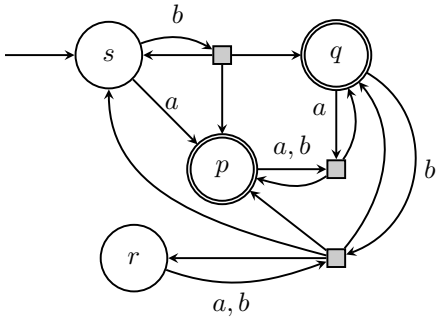


Figure 4.5: The canonical RUFA for the language $L_1 = a^+ \cup ba^+ \cup aba^*$.

**Definition 4.11.** *Let $L$ be a regular language and $\mathcal{T} = (T, U, V)$ be a table for $L$.*

- *A row $r_u$ of a table $\mathcal{T}$ is* $\sqcup$-composite *if there are rows $r_{u_1}, \ldots, r_{u_k} \in \mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$, with $r_{u_i} \neq r_u$, such that $r_u = \bigsqcup_{i=1}^{k} r_{u_i}$. Otherwise, $r_u$ is called* $\sqcup$-prime*. Let $\mathrm{Primes}_{\sqcup}(\mathcal{T})$ be the set of $\sqcup$-prime rows in $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$. For the $\sqcap$ operator,* $\sqcap$-composite*,* $\sqcap$-prime*, and $\mathrm{Primes}_{\sqcap}(\mathcal{T})$ are defined analogously.*
- *To simplify the notation, for every $r_u \in \mathrm{Rows}(\mathcal{T})$, let $\mathbb{B}_{\sqcup}(r_u) := \{r_{u'} \in \mathrm{Rows}_{\mathrm{high}}(\mathcal{T}) : r_{u'} \leq r_u\}$ and $\mathbb{B}_{\sqcap}(r_u) := \{r_{u'} \in \mathrm{Rows}_{\mathrm{high}}(\mathcal{T}) : r_u \leq r_{u'}\}$.*
- *A subset of rows that can generate all rows of $\mathcal{T}$ using $\sqcup$ (or $\sqcap$) is called a* $\sqcup$-basis *(or* $\sqcap$-basis*, resp.) for $\mathcal{T}$.*

For example, in Fig. 4.3, the row $r_b$ of $\mathcal{T}$ is $\sqcap$-composite as $r_b = r_\lambda \sqcap r_a$, whereas $r_\lambda, r_a, r_{ab}$ are $\sqcap$-prime and $\mathrm{Primes}_{\sqcap}(\mathcal{T}) = \{r_\lambda, r_a\}$.

Thus, $\mathcal{T}$ is $\sqcup$-closed if $\mathrm{Primes}_{\sqcup}(\mathcal{T})$ is a $\sqcup$-basis for $\mathcal{T}$. Analogously, $\mathcal{T}$ is $\sqcap$-closed if $\mathrm{Primes}_{\sqcap}(\mathcal{T})$ is a $\sqcap$-basis for $\mathcal{T}$ The table in Fig. 4.3 is not $\sqcap$-closed as the row $r_{ab} \in \mathrm{Rows}(\mathcal{T})$ is not composable by rows of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$.

In this thesis, a weaker form of *consistency* is used for $\mathtt{NL}^\star$ compared with Bollig et al. [15] (or Kern [42] for $\mathtt{UL}^\star$, respectively). Namely, the initial consistency notion as already introduced by Angluin [5] for $\mathtt{L}^\star$ is used. In [7], Angluin et al. have mentioned, but not proven, that this weaker form is sufficient for $\mathtt{NL}^\star$, $\mathtt{UL}^\star$, and $\mathtt{AL}^\star$. A formal justification has been given by Berndt et al. in [12]. The versions of $\mathtt{NL}^\star$ and $\mathtt{UL}^\star$ presented here use the weaker consistency notion. The original algorithms, which use the stronger consistency notion, have been presented in [15, 42]. With this weaker consistency notion, Lemma 4.7 can be applied for $\mathtt{NL}^\star$ and $\mathtt{UL}^\star$.

The construction of the NFA or UFA is as follows.

**Definition 4.12.** *Let $\mathcal{T}$ be a consistent and X-closed table. For $X = N$ or $X = U$, the XFA $\mathfrak{A}^{\mathrm{X}}(\mathcal{T}) = (Q, Q_0, F, \tau)$ consists of the following components:*

- *If $X = N$, $Q = \mathrm{Primes}_{\sqcup}(\mathcal{T})$, $Q_0 = \mathbb{B}_{\sqcup}(r_\lambda) \cap Q$ and $F = \{r \in Q : r[\lambda] = +\}$. For $r \in Q$ and $a \in \Sigma$ let $\tau(r, a) = \mathbb{B}_{\sqcup}(ra) \cap Q$.*

| | $\lambda$ | $a$ | $ba$ |
|---|---|---|---|
| $\lambda$ | $-$ | $+$ | $-$ |
| $a$ | $+$ | $-$ | $-$ |
| $b$ | $-$ | $-$ | $-$ |
| $aa$ | $-$ | $+$ | $-$ |
| $ab$ | $-$ | $-$ | $-$ |

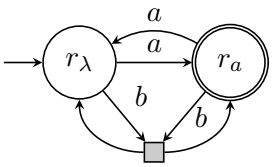


Figure 4.6: A consistent and U-closed table $\mathcal{T}$ and the corresponding UFA $\mathfrak{U}^{\mathrm{D}}(\mathcal{T})$.

- *If* $X = U$, $Q = \mathrm{Primes}_{\sqcap}(\mathcal{T})$, $Q_0 = \mathbb{B}_{\sqcap}(r_\lambda) \cap Q$ *and* $F = \{r \in Q : r[\lambda] = +\}$. *For* $r \in Q$ *and* $a \in \Sigma$ *let* $\tau(r, a) = \mathbb{B}_{\sqcap}(ra) \cap Q$.

These constructions complete the presentation of $\mathtt{NL}^\star$ and $\mathtt{UL}^\star$. For an example of building a UFA from a table, see Fig. 4.6. Both algorithms, $\mathtt{NL}^\star$ and $\mathtt{UL}^\star$, learn the corresponding canonical automata:

**Theorem 4.13** (Bollig et al. [15])**.** *For every regular language* $L \in \mathcal{C}_d$*, the algorithm* $\mathtt{NL}^\star$ *always generates the canonical RNFA* $\mathfrak{A}^{\mathrm{N}}$ *such that* $L(\mathfrak{A}^{\mathrm{N}}) = L$*. The algorithm runs in time bounded by a polynomial in* $d$*,* $|\Sigma|$*, and* $\ell$*, where* $\ell$ *is the size of the longest counterexample obtained from the equivalence oracle.*

**Theorem 4.14** (Kern [42])**.** *For every regular language* $L \in \mathcal{C}_d$*, the algorithm* $\mathtt{UL}^\star$ *always generates the canonical RUFA* $\mathfrak{A}^{\mathrm{U}}$ *such that* $L(\mathfrak{A}^{\mathrm{U}}) = L$*. The algorithm runs in time bounded by a polynomial in* $d$*,* $|\Sigma|$*, and* $\ell$*, where* $\ell$ *is the size of the longest counterexample obtained from the equivalence oracle.*

### 4.2.3 Learning Alternating Automata

In the following we describe the $\mathtt{AL}^\star$ algorithm of Angluin et al. [7]. The authors have presented a basic version (see Algorithm 1 in their paper) and suggested further optimizations afterwards. In our experiments (see Section 4.4.2), these optimizations are reducing the number of membership queries, but they are increasing the number of expensive equivalence queries. Our new algorithm $\mathtt{AL}^{\star\star}$ does not use these optimizations, nor do $\mathtt{L}^\star$, $\mathtt{NL}^\star$, or $\mathtt{UL}^\star$. Thus we present and analyze the basic version of $\mathtt{AL}^\star$ here. Note that our counterexample, which we will present in Section 4.3, works with both versions of $\mathtt{AL}^\star$.

**Definition 4.15.** *In the following* $P$ *will always denote a subset of* $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$.

- *The set* $P$ *is a* $(\sqcup, \sqcap)$-basis *for* $\mathcal{T}$ *(in the following simply called a basis) if* $\mathrm{Rows}(\mathcal{T}) \subseteq [\![\mathcal{F}(P)]\!]$ *and table* $\mathcal{T}$ *is then called* $P$-closed.
- *The table* $\mathcal{T}$ *is called* $P$-minimal *if* $P$ *is a minimal basis for* $\mathcal{T}$*, i. e. for all* $p \in P$*, the set* $P \setminus \{p\}$ *is not a basis.*
- *For a* $P$*-closed table* $\mathcal{T}$ *and* $v \in V$*, let* $M^P(v)$ *be the monomial defined by*

$$M^P(v) := \bigwedge_{p \in P, p[v] = +} p \ ,$$

*which is a maximal one over all monomials in* $\mathcal{F}_\wedge(P)$ *such that* $\left[\!\left[M^P(v)\right]\!\right][v] = +$. *If for all* $p \in P$ *we have* $p[v] = -$ *then* $M^P(v) := \top$.
- *For* $r \in \mathrm{Rows}(\mathcal{T})$ *of a* $P$*-closed table* $\mathcal{T}$ *let* $b^P(r) \in \mathcal{F}(P)$ *be the expression*

$$b^P(r) = \bigvee_{v \in V, r[v]=+} M^P(v)$$

*representing* $r$. *If for all* $v \in V$ *we have* $r[v] = -$ *then* $b^P(r) := \bot$.
- *For a monomial* $M$ *and* $a \in \Sigma$ *we define* $Ma$ *as the monomial derived from* $M$ *by replacing every row* $r \in P$ *of* $M$ *by* $ra$.

Note that $\left[\!\left[b^P(r)\right]\!\right] = r$.

**Definition 4.16.** *Let* $\varphi$ *be an MDNF formula consisting of monomials* $M_i$. *We use the notation* $M_i \sqsubset \varphi$ *and for a monomial* $M_i = \bigwedge_j x_j$ *the notation* $x_j \sqsubset M_i$ *for its literals* $x_j$.
*For formulas* $\varphi(x_1, \ldots, x_k)$ *and* $\psi(x_1, \ldots, x_k)$ *with literals* $x_1, \ldots, x_k$ *that represent vectors* $r$ *over* $\{+, -\}$, *we say that* $\varphi$ *and* $\psi$ *are* equivalent *(in symbols* $\varphi \equiv \psi$*) if* $\left[\!\left[\varphi(r_1, r_2, \ldots, r_k)\right]\!\right] = \left[\!\left[\psi(r_1, \ldots, r_k)\right]\!\right]$ *for all vectors* $r_1, \ldots, r_k$ *of identical length.*

Now all necessary tools have been defined to construct an AFA $\mathfrak{A}^P(\mathcal{T})$ from a table a $\mathcal{T}$.

**Definition 4.17.** *Let* $\mathcal{T}$ *be a consistent and* $P$*-closed table. The AFA* $\mathfrak{A}^P(\mathcal{T}) = (Q, Q_0, F, \tau)$ *consists of the following components:* $Q = P$, $Q_0 = b^P(r_\lambda)$ *and* $F = \{r \in P : r[\lambda] = +\}$. *For* $r \in Q$ *and* $a \in \Sigma$ *let* $\tau(r, a) = b^P(ra)$.

Recall that, according to our convention, the term $ra$ in the last expression denotes the vector $r_{ua}$ such that $u \in U$ is any string with $r_u = r$. Note, moreover, that $\tau(r, a) = b^P(ra)$ is always an MDNF formula. An example of constructing an AFA from a table is illustrated in Fig. 4.7.

The automaton $\mathfrak{A}^{\mathrm{A}}(\mathcal{T})$ computed by $\mathtt{AL}^\star$ (Line 8, Algorithm 3) has the form $\mathfrak{A}^{\mathrm{A}}(\mathcal{T}) = \mathfrak{A}^P(\mathcal{T})$, where $P$ is an arbitrary minimal basis for $\mathcal{T}$. $\mathtt{AL}^\star$ may adapt the old minimal basis whenever it computes a new automaton. The size of the basis equals the number of states of the automaton $\mathfrak{A}^P(\mathcal{T})$. Thus the basis should be as small as possible. However, $\mathtt{AL}^\star$ only constructs a minimal basis, because computing a minimum basis (i. e. of minimal cardinality) is $\mathcal{NP}$-hard, as shown by Angluin et al. in [7]. The approximation of the minimum basis is discussed in Section 4.4.1.

**Theorem 4.18** (Angluin et al. [7])**.** *For every regular language* $L \in \mathcal{C}_d$, *the algorithm* $\mathtt{AL}^\star$ *always generates an AFA* $\mathfrak{A}^P$ *such that* $L(\mathfrak{A}^P) = L$. *The algorithm runs in time bounded by a polynomial in* $d$, $|\Sigma|$, *and* $\ell$, *where* $\ell$ *is the size of the longest counterexample obtained from the equivalence oracle.*

| | $\lambda$ | $a$ | $ba$ |
|---|---|---|---|
| $\lambda$ | $-$ | $+$ | $+$ |
| $a$ | $+$ | $+$ | $-$ |
| $b$ | $-$ | $+$ | $-$ |
| $aa$ | $+$ | $+$ | $+$ |
| $ab$ | $-$ | $-$ | $-$ |

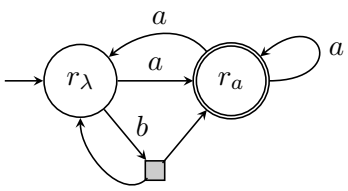


Figure 4.7: A consistent and $P$-closed table $\mathcal{T}$ with $P = \{r_\lambda, r_a\}$ and the corresponding AFA $\mathfrak{A}^P(\mathcal{T})$. For reasons of clarity, here and in the following figures, unnecessary monomials have been removed from the MDNF formulas: the formula $r_a \vee (r_\lambda \wedge r_a)$ has been replaced by the equivalent formula $r_a$, as well as $r_a \vee (r_\lambda \wedge r_a) \vee r_\lambda$ by $r_a \vee r_\lambda$.

## 4.3 Analysis of the AL$^\star$ Algorithm

The AL$^\star$ algorithm to learn alternating automata has been presented in [7] and its running time has been analyzed. However, properties of the automata produced have remained unclear. We close this gap by establishing several properties of AL$^\star$. These properties form a part of the proof that the new AL$^{\star\star}$ always generates residual automata. Thus, the counterexample to disprove the conjecture about residuality needs to be constructed around the properties of AL$^\star$ we prove in this section. We finish our analysis of AL$^\star$ with the presentation of the counterexample.

Let us analyze the properties of the automata generated by AL$^\star$ in detail. From the construction of the AFA one can easily derive the following lemma.

**Lemma 4.19.** *For every $\varphi \in \mathcal{F}(Q)$ and every automaton $\mathfrak{A}^P(\mathcal{T})$ it holds: $(\!|\varphi|\!) = \top$ if and only if $[\![\varphi]\!]\,[\lambda] = +$.*

*Proof.* We use induction upon the nesting of $\varphi$. For $r \in Q$ it holds $(\!|r|\!) = \top \Leftrightarrow r \in F \Leftrightarrow r[\lambda] = [\![r]\!]\,[\lambda] = +$. In the inductive step one can conclude

$$\begin{aligned}
(\!|\psi \wedge \xi|\!) = \top &\iff (\!|\psi|\!) = \top \wedge (\!|\xi|\!) = \top \\
&\iff [\![\psi]\!]\,[\lambda] = + \wedge [\![\xi]\!]\,[\lambda] = + \iff [\![\psi \wedge \xi]\!]\,[\lambda] = + \,, \\
(\!|\psi \vee \xi|\!) = \top &\iff (\!|\psi|\!) = \top \vee (\!|\xi|\!) = \top \\
&\iff [\![\psi]\!]\,[\lambda] = + \vee [\![\xi]\!]\,[\lambda] = + \iff [\![\psi \vee \xi]\!]\,[\lambda] = + \,.
\end{aligned}$$

□

In the following, fix a regular language $L$, a prefix-closed set $U$, a suffix-closed set $V$, the corresponding table $\mathcal{T}$ and a minimal basis $P$ of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$.

**Lemma 4.20.** *For all $r \in P$ and $v \in V$ holds $r[v] = [\![\tau(r, v)]\!]\,[\lambda]$.*

*Proof.* We prove the lemma by induction upon the length of $v$. For $v = \lambda$ one gets $r[\lambda] = [\![r]\!]\,[\lambda] = [\![\tau(r,\lambda)]\!]\,[\lambda]$.

For $v = av'$ we have $r[v] = r[av'] = ra[v']$, as $r \in P$ and $v' \in V$ due to its suffix-closedness.

- case $ra[v'] = +$:
  It holds $\tau(r,a) = b^P(ra) = \bigvee_{\substack{v \in V \\ ra[v]=+}} M^P(v)$. As $v' \in V$ and $ra[v'] = +$ one can conclude $M^P(v') \sqsubset \tau(r,a)$. For every $r' \sqsubset M^P(v')$ the induction hypothesis implies $[\![\tau(r',v')]\!]\,[\lambda] = r'[v'] = +$ by the definition of $M^P(v')$. Hence

  $$[\![\tau(M^P(v'),v')]\!]\,[\lambda] = \left[\!\!\left[\tau\Big(\bigwedge_{r' \in P,\, r'[v']=+} r', v'\Big)\right]\!\!\right][\lambda] = + \,.$$

  Finally, as $\tau(r,a)$ contains the monomial $M^P(v')$ one can conclude

  $$\begin{aligned} [\![\tau(r,v)]\!]\,[\lambda] \;&=\; [\![\tau(r,av')]\!]\,[\lambda] \;=\; [\![\tau(\tau(r,a),v')]\!]\,[\lambda] \\ &\geq\; [\![\tau(M^P(v'),v')]\!]\,[\lambda] = + \,. \end{aligned}$$

- case $ra[v'] = -$:
  For every monomial $M \sqsubset \tau(r,a)$ it must hold $[\![M]\!]\,[v'] = -$. Thus, there is a row $r_M \in P$ with $r_M[v'] = -$. The induction hypothesis then implies $[\![\tau(r_M,v')]\!]\,[\lambda] = -$. So, for every $M \sqsubset \tau(r,a)$ we get $[\![\tau(M,v')]\!]\,[\lambda] = -$, and finally

  $$[\![\tau(r,v)]\!]\,[\lambda] \;=\; [\![\tau(\tau(r,a),v')]\!]\,[\lambda] \;=\; [\![\tau(b^P(ra),v')]\!]\,[\lambda] = - \,.$$

Hence, $ra[v'] = +$ if and only if $[\![\tau(r,v)]\!]\,[\lambda] = +$ which implies $r[v] = +$ if and only if $[\![\tau(r,v)]\!]\,[\lambda] = +$. □

**Lemma 4.21.** *For all $\varphi \in \mathcal{F}(P)$ and $v \in V$ we have $[\![\varphi]\!]\,[v] = [\![\tau(\varphi,v)]\!]\,[\lambda]$.*

*Proof.* We may assume that $\varphi$ is in MDNF. If $[\![\varphi]\!]\,[v] = -$ then for every monomial $M \sqsubset \varphi$ it must hold $[\![M]\!]\,[v] = -$. Therefore, there exists some $r \sqsubset M$, such that $r[v] = -$. By Lemma 4.20, $[\![\tau(r,v)]\!]\,[\lambda] = -$ and hence $[\![\tau(\varphi,v)]\!]\,[\lambda] = -$.

Otherwise, if $[\![\varphi]\!]\,[v] = +$ there exists a monomial $M \sqsubset \varphi$ with $[\![M]\!]\,[v] = +$. Hence, for all $r \sqsubset M$ it must hold $r[v] = +$. Lemma 4.20 implies $[\![\tau(r,v)]\!]\,[\lambda] = +$ and thus $[\![\tau(\varphi,v)]\!]\,[\lambda] = +$. □

Using these properties we continue the analysis as follows.

**Lemma 4.22.** *If $\mathcal{T}$ and $\mathfrak{A}^P(\mathcal{T})$ are compatible then for every $u \in U$ with $r_u \in P$ it holds $L(\mathfrak{A}^P_{r_u}(\mathcal{T})) \subseteq u^{-1}L(\mathfrak{A}^P(\mathcal{T}))$.*

*Proof.* Assume $L(\mathfrak{A}^P_{r_u}(\mathcal{T})) \not\subseteq u^{-1}L(\mathfrak{A}^P(\mathcal{T}))$, i. e. there exists a string $\omega$ such that $\omega \in L(\mathfrak{A}^P_{r_u}(\mathcal{T}))$ and $\omega \notin u^{-1}L(\mathfrak{A}^P(\mathcal{T}))$. Since $\omega \in L(\mathfrak{A}^P_{r_u}(\mathcal{T}))$, we have $[\![\tau(r_u, \omega)]\!][\lambda] = +$ by definition. Moreover, $\omega \notin u^{-1}L(\mathfrak{A}^P(\mathcal{T}))$ implies $u\omega \notin L(\mathfrak{A}^P(\mathcal{T}))$ and thus $[\![\tau(\tau(Q_0, u), \omega)]\!][\lambda] = -$.

We will now prove that such an $\omega$ cannot come from $V$ or $\Sigma V$ by showing that $\omega \notin (\Sigma \cup \{\lambda\})V$. Assume that $\omega = av$ with $a \in \Sigma \cup \{\lambda\}, v \in V$. By Lemma 4.21, $[\![\tau(r_u, a)]\!][v] = [\![\tau(r_u, \omega)]\!][\lambda]$. Further, $[\![\tau(r_u, a)]\!] = r_{ua}$ by definition. Thus

$$r_{ua}[v] = [\![\tau(r_u, a)]\!][v] = [\![\tau(r_u, \omega)]\!][\lambda] = +,$$

but this contradicts compatibility, as $r_{ua}[v] = +$ implies that $uav = u\omega \in L(\mathfrak{A}^P(\mathcal{T}))$.

Now let $\omega = a\tilde{\omega}$. From the construction of $\tau$, we know that the row $r_{ua}$ is not completely filled with $-$, since

$$\begin{aligned}[\![\tau(b^P(r_{ua}), \tilde{\omega})]\!][\lambda] &= [\![\tau(b^P(-\cdots-), \tilde{\omega})]\!][\lambda] = [\![\tau(\bot, \tilde{\omega})]\!][\lambda] \\ &= [\![\bot]\!][\lambda] = -\end{aligned}$$

would contradict

$$\begin{aligned}+ &= [\![\tau(r_u, \omega)]\!][\lambda] = [\![\tau(r_u, a\tilde{\omega})]\!][\lambda] = [\![\tau(\tau(r_u, a), \tilde{\omega})]\!][\lambda] \\ &= [\![\tau(b^P(r_{ua}), \tilde{\omega})]\!][\lambda] \ .\end{aligned}$$

Let $\tau(Q_0, u)_{\text{MDNF}} = M_1 \vee M_2 \vee \cdots \vee M_k$ be the formula that is reached in the automaton after reading $u$. For every column $v \in V$ with $r_{ua}[v] = +$, consider all monomials $M_i$ with $[\![M_i a]\!][v] = +$. There must be at least one monomial, because otherwise $uav \notin L(\mathfrak{A}^P(\mathcal{T}))$, which would contradict the compatibility of $\mathcal{T}$ and $\mathfrak{A}^P(\mathcal{T})$. It holds $M^P(v) \sqsubset \tau(r_u, a)$ by the construction of $\tau(r_u, a) = b^P(r_{ua})$. For every row $r_{\tilde{u}} \sqsubset M_i$, we have $\tau(r_{\tilde{u}}, a) = b^P(r_{\tilde{u}a}) = \bigvee_{\tilde{v} \in V, r_{\tilde{u}a}[\tilde{v}]=+} M^P(\tilde{v})$. Hence, $M^P(v) \sqsubset \tau(r_{\tilde{u}a})$. Thus, $M^P(v) \sqsubset \tau(M_i, a)_{\text{MDNF}}$ and $M^P(v) \sqsubset \tau(M_1 \vee \cdots \vee M_k, a)_{\text{MDNF}}$.

So, for every monomial $M^P(v) \sqsubset \tau(r_u, a)$, we have $M^P(v) \sqsubset \tau(M_1 \vee \cdots \vee M_k, a)_{\text{MDNF}}$ and thus $M^P(v) \sqsubset \tau(Q_0, u)_{\text{MDNF}}$. Hence, $[\![\tau(r_u, a\tilde{\omega})]\!][\lambda] = +$ directly implies

$$[\![\tau(M_1 \vee \cdots \vee M_k, a\tilde{\omega})]\!][\lambda] = + \ .$$

But $[\![\tau(M_1 \vee \cdots \vee M_k, a\tilde{\omega})]\!][\lambda] = [\![\tau(\tau(Q_0, u), \omega)]\!][\lambda] = -$. Hence, this is a contradiction and no such $\omega$ exists. □

For NFAs and UFAs, the reverse inclusion between the two languages in the statement of Lemma 4.22 holds in the case of compatibility, too. Angluin et al. [7] have conjectured that this is also the case for AFAs since extensive tests of the algorithm $\texttt{AL}^\star$ never produced a non-residual AFA. With the help of specially developed software that simulates and visualizes the run of $\texttt{AL}^\star$ interactively, we have been able to construct a counterexample. With our software, the user can define the alphabet $\Sigma$, a target regular language $L$, the rows $U$ and columns $V$ of

the table $\mathcal{T}$, and a specific minimal base if desired. The software fills the entire table $\mathcal{T}$ including the rows $R = U\Sigma \setminus U$ by membership tests for $L$, computes a minimal basis $P$ if not fixed by the user, and generates a visualization of $\mathfrak{A}^P(\mathcal{T})$. Based on a failed attempt to prove the reverse inclusion of Lemma 4.22, the following counterexample has been constructed with the help of the software eventually.

**Lemma 4.23.** *There exists a regular language $L$ for which the algorithm* AL$^\star$ *constructs a table $\mathcal{T}$ defining a compatible AFA $\mathfrak{A}^P(\mathcal{T})$ with $L(\mathfrak{A}^P(\mathcal{T})) = L$, such that for some $r \in P$ and all $\omega \in \Sigma^*$ the residual language $\omega^{-1}L$ is not contained in $L(\mathfrak{A}_r^P(\mathcal{T}))$.*

*Proof.* It can be shown that the AFA in Fig. 4.8 is compatible to a table $\mathcal{T}$ that can be constructed by AL$^\star$ on a carefully designed language $L$. The state labeled $nr$ is not residual.

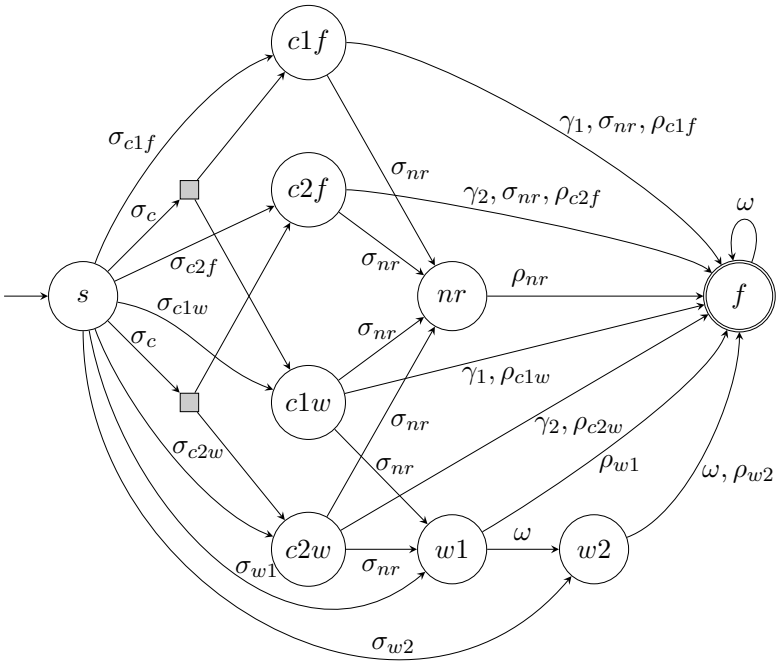


Figure 4.8: A non-residual AFA constructed by AL$^\star$ with initial configuration $Q_0 = s$ and accepting states $F = \{f\}$.

We give the construction of the language $L$ and the run of AL$^\star$ that results in the AFA mentioned above. Let

$$\begin{aligned}\Sigma = \{&\sigma_c, \sigma_{nr}, \sigma_{c1f}, \sigma_{c1w}, \sigma_{c2f}, \sigma_{c2w}, \sigma_{w1}, \sigma_{w2},\\ &\rho_{nr}, \rho_{c1f}, \rho_{c1w}, \rho_{c2f}, \rho_{c2w}, \rho_{w1}, \rho_{w2}, \gamma_1, \gamma_2, \omega\}\ ,\\ L_x = \{&\sigma_c\gamma_1,\ \sigma_c\gamma_2,\ \sigma_c\sigma_{nr}\rho_{nr},\\ &\sigma_{c1f}\rho_{c1f}, \sigma_{c1w}\rho_{c1w},\ \sigma_{c2f}\rho_{c2f},\ \sigma_{c2w}\rho_{c2w},\\ &\sigma_{w1}\rho_{w1},\ \sigma_{w2}\rho_{w2}\}\ ,\end{aligned}$$

$$
\begin{aligned}
L_y &= \{\sigma_{c1f}\gamma_1,\ \sigma_{c1f}\sigma_{nr}\rho_{nr},\ \sigma_{c1w}\gamma_1,\ \sigma_{c1w}\sigma_{nr}\rho_{nr},\\
&\qquad \sigma_{c2f}\gamma_2,\ \sigma_{c2f}\sigma_{nr}\rho_{nr},\ \sigma_{c2w}\gamma_2,\ \sigma_{c2w}\sigma_{nr}\rho_{nr}\}\ ,\\
L_z &= \{\sigma_{w2}\omega,\ \sigma_{w1}\omega\omega,\ \sigma_{w1}\omega\rho_{w2},\ \sigma_{c1f}\sigma_{nr},\ \sigma_{c2f}\sigma_{nr},\\
&\qquad \sigma_{c1w}\sigma_{nr}\rho_{w1},\ \sigma_{c1w}\sigma_{nr}\omega\rho_{w2},\ \sigma_{c1w}\sigma_{nr}\omega\omega,\\
&\qquad \sigma_{c2w}\sigma_{nr}\rho_{w1},\ \sigma_{c2w}\sigma_{nr}\omega\rho_{w2}\sigma_{c2w}\sigma_{nr}\omega\omega\}\ ,\\
L_2 &= (L_x \cup L_y \cup L_z \cup \{\sigma_c\sigma_{nr}\omega\omega\})\{\omega\}^*\ ,
\end{aligned}
$$

and $\mathfrak{A}$ be the AFA illustrated in Fig. 4.8. A detailed case analysis shows $L(\mathfrak{A}) = L_2$.

For learning $L_2$, the following implementation of an *EQUIV* oracle is used based on a total ordering $\prec$ over $L_x$ given by

$$
\begin{aligned}
&\sigma_c\sigma_{nr}\rho_{nr} \prec \sigma_{c1f}\rho_{c1f} \prec \sigma_{c2f}\rho_{c2f} \prec \sigma_c\gamma_2 \prec \sigma_c\gamma_1 \prec \sigma_{c2w}\rho_{c2w} \prec \sigma_{c1w}\rho_{c1w}\\
&\qquad \prec \sigma_{w1}\rho_{w1} \prec \sigma_{w2}\rho_{w2}\ .
\end{aligned}
$$

For a hypothesized AFA $\mathfrak{A}'$ that does not accept $L_2$, *EQUIV* according to $\prec$ searches the smallest element $\xi \in L_x \setminus L(\mathfrak{A}')$. If such a $\xi$ exists, *EQUIV* returns it as counterexample, otherwise an arbitrary counterexample is chosen.[3]

We have implemented $\mathtt{AL}^\star$ with access to this equivalence oracle. For the language $L_2$ the non-residual automaton $\mathfrak{A}$ in Fig. 4.8 has been obtained as final result. This holds for both cases, $\mathtt{AL}^\star$ with and without the optimizations suggested by Angluin et al. The complete table $\mathcal{T}$ of the corresponding run is not presented here since it has hundreds of rows. □

We now give some intuition concerning the construction of the non-residual automaton $\mathfrak{A}$ or rather the language $L_2$. Let $Q' = \{c1f, c1w, c2f, c2w, w1, w2\}$ be a subset of the states of $\mathfrak{A}$. For $q \in Q'$ the symbol $\sigma_q$ is used to generate a row of $\mathcal{T}$ of a specific form that will serve as a state of $\mathfrak{A}$. For $q$ to ensure that the corresponding row is prime the symbol $\rho_q$ is used which generates a unique $+$ in this row. Thus, we add $\sigma_q\rho_q$ to $L_x$ for every $q \in Q'$. In the subtable in Fig. 4.9 see for example the second row labelled $\sigma_{c1f}$ and column labelled $\rho_{c1f}$. This column has a single $+$ at this row that makes this row prime.

To get a non-residual AFA, Lemma 4.24 implies that we have to construct a prime row $r_u$ where $u = \sigma_c\sigma_{nr}$ such that $r_{\sigma_c}$ is not prime. This is achieved by symbols $\gamma_1, \gamma_2$. For the subtable in Fig. 4.9 one notices that

$$
r_{\sigma_c} \ = \ (r_{\sigma_{c1f}} \sqcap r_{\sigma_{c1w}}) \ \sqcup \ (r_{\sigma_{c2f}} \sqcap r_{\sigma_{c2w}})\ .
$$

[3] We cannot provide the sequence of counterexamples exactly because it depends on details of the implementation of $\mathtt{AL}^\star$. In [7] the authors have suggested some optimizations in order to save membership queries. However, these optimizations may increase the number of (expensive) equivalence queries, because now $\mathtt{AL}^\star$ may produce automata that do not classify already seen counterexamples correctly. In this case, the equivalence oracle defined above would simply provide a previous counterexample again.

The row with label $\sigma_c\sigma_{nr}$ plays a special role representing the non-residual state $nr$. This completes the construction of $L_x \subseteq L_2$.

| | $\lambda$ | $\gamma_1$ | $\gamma_2$ | $\rho_{c1f}$ | $\rho_{c1w}$ | $\rho_{c2f}$ | $\rho_{c2w}$ | $\rho_{nr}$ |
|---|---|---|---|---|---|---|---|---|
| $\sigma_c$ | − | + | + | − | − | − | − | − |
| $\sigma_{c1f}$ | − | + | − | + | − | − | − | − |
| $\sigma_{c1w}$ | − | + | − | − | + | − | − | − |
| $\sigma_{c2f}$ | − | − | + | − | − | + | − | − |
| $\sigma_{c2w}$ | − | − | + | − | − | − | + | − |
| $\sigma_c\sigma_{nr}$ | − | − | − | − | − | − | − | + |

Figure 4.9: A subtable of $\mathcal{T}$ used to construct non-residual AFA $\mathfrak{A} = \mathfrak{A}(\mathcal{T})$.

We still have to make sure that the state $nr$ corresponding to $r_u$ is non-residual. For this purpose, we add the string $\sigma_c\sigma_{nr}\omega\omega$ to the language $L_2$ and make sure that the string $\omega\omega$ is not accepted while the automaton is in state $nr$, i. e. $\omega\omega \notin L(\mathfrak{A}_{r_u})$. For the automaton to accept $\sigma_c\sigma_{nr}\omega\omega$ the suffix $\sigma_{nr}\omega\omega$ must be accepted from the configuration $(c1f \wedge c1w) \vee (c2f \wedge c2w)$. The non-residuality is achieved by the table not containing information on $\sigma_c\sigma_{nr}\omega\omega$ and $\omega\omega$. To hide this information, we have to make sure that $\omega$ is never added to $V$. This is done by two "waiting" states $w1$ and $w2$. As a path from the states $c1w$ and $c2w$ to the accepting state $f$ visits the states $w1$ and $w2$, the automaton has to "wait" for the string $\omega\omega$ to reach $f$. By construction of $r_{\sigma_c}$ either $c1w$ or $c2w$ have to be visited in order to accept a word. But, as $\omega \notin V$, this waiting behavior cannot be observed by $\texttt{AL}^\star$ and hence $\omega\omega \notin L(\mathfrak{A}_{r_u})$.

For technical reasons one has to add some words like $\sigma_{w1}\omega\omega$ to the language to get the final version of $L_2$. Based on these properties, the segmentation of $L_2$ is as follows. The words in $L_x$ are the counterexamples that let $\texttt{AL}^\star$ add necessary columns to $V$ and finally necessary rows to $U$. The words in $L_y$ ensure that row $r_{\sigma_c}$ gets suitably extended. Finally, the words in $L_z$ correspond to the waiting process before merging the different $\wedge$-branches in the accepting state $f$.

## 4.4 Learning Residual Alternating Automata

Let $L$ be a given regular language. In order to construct only residual AFAs for $L$ we build on $\texttt{AL}^\star$ and design a new algorithm $\texttt{AL}^{\star\star}$ presented as Algorithm 4 that solves this problem. The main obstacle that one encounters is the test of residuality of the constructed automaton. We use the power of the equivalence-oracle to incorporate this task into $\texttt{AL}^\star$ by reducing it to a single equivalence query of a larger automaton.

The main difference between $\texttt{AL}^\star$ and $\texttt{AL}^{\star\star}$ lies in the construction of the automaton $\mathfrak{A}^{P'}(\mathcal{T})$ in Line 10 of $\texttt{AL}^{\star\star}$. This modification of $\texttt{AL}^\star$ allows us to guarantee the residuality of the generated automaton. The lemmata shown for $\texttt{AL}^\star$ hold for $\texttt{AL}^{\star\star}$ as well, especially Lemma 4.7 and 4.22. As shown in the previous section,

**Input:** membership oracle $MEMBER(L)$ and equivalence oracle $EQUIV(L)$ respecting the unknown target language $L \subseteq \Sigma^*$
**Output:** an AFA $\mathfrak{A}$

```
1  U ← {λ}; V ← {λ};
2  initialize 𝒯 = (T, U, V) with |Σ| + 1 membership queries;
3  while true do
4      while 𝒯 is not A-closed do
5          find a row r_ua ∈ Rows(𝒯) with r_ua ∉ [[ℱ(Rows_high(𝒯))]];
6          add ua to U;
7          complete 𝒯 via membership queries;
8      construct a minimal basis P and the AFA 𝔄^P(𝒯) for P;
9      if L(𝔄^P(𝒯)) = L then
10         construct 𝔄^P'(𝒯) with P' = Rows_high(𝒯);
11         if L(𝔄^P'(𝒯)) = L then
12             return 𝔄^P(𝒯);
13         else
14             get a counterexample w ∈ L △ L(𝔄^P'(𝒯));
15             set V ← V ∪ Suffs(w);
16             complete 𝒯 via membership queries;
17     else
18         get a counterexample w ∈ L △ L(𝔄^P(𝒯));
19         set V ← V ∪ Suffs(w);
20         complete 𝒯 via membership queries;
```

**Algorithm 4:** AL$^{\star\star}$

the reason for the possible non-residuality of the automaton produced by AL$^\star$ is that the reverse statement of Lemma 4.22 does not hold for AFAs. As we perform no basis reduction in the construction of $\mathfrak{A}^{P'}(\mathcal{T})$, compatibility of the table and the automaton guarantees residuality of the automaton.

**Lemma 4.24.** *If the AFA $\mathfrak{A}^{P'}(\mathcal{T})$ constructed in Line 10 is compatible with $\mathcal{T}$, then automaton $\mathfrak{A}^{P'}(\mathcal{T})$ is residual.*

*Proof.* Consider some $u \in U$. As $P' = \mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$, we have $r_u \in P'$ and thus $L(\mathfrak{A}^{P'}_{r_u}(\mathcal{T})) \subseteq u^{-1}L(\mathfrak{A}^{P'}(\mathcal{T}))$ by Lemma 4.22. It remains to prove the inclusion in the other direction. Iterating over the length of $u$ one can show that for every configuration of the AFA $\tau(Q_0, u) \equiv r_u \wedge R_u$, where $R_u$ is some expression.

By construction, every monomial of $Q_0 = b^P(r_\lambda)$ contains $r_\lambda$. Therefore, $Q_0 \equiv r_\lambda \wedge R_\lambda$ for some expression $R_\lambda$. Hence, $\tau(Q_0, \lambda) = Q_0 \equiv r_\lambda \wedge R_\lambda$.

As $U$ is prefix-closed, every prefix of $u$ is also in $U$. If $u = u'a$, every monomial of $\tau(u', a)$ contains $r_{u'a} = r_u \in P'$ by the induction hypothesis. Therefore,

$\tau(u', a) \equiv r_u \wedge R'_u$, where $R'_u$ is an expression. Thus, for an appropriate expression $R_u$ we get

$$\tau(Q_0, u) = \tau(\tau(Q_0, u'), a) \equiv \tau(r_{u'} \wedge R_{u'}, a) \equiv (r_u \wedge R'_u) \wedge R_{u'} \equiv r_u \wedge R_u \ .$$

Therefore, $L(\mathfrak{A}^{P'}_{r_u}(\mathcal{T})) \supseteq u^{-1}L(\mathfrak{A}^{P'}(\mathcal{T}))$. □

Computing the large residual automaton $\mathfrak{A}^{P'}(\mathcal{T})$ in Line 10 upon the trivial basis $P'$ allows us to test the smaller automaton $\mathfrak{A}^{P}(\mathcal{T})$ for residuality via the following lemma. If $\mathfrak{A}^{P'}(\mathcal{T})$ passes the equivalency test it certificates the residuality of $\mathfrak{A}^{P}(\mathcal{T})$. Otherwise, the construction directly gives us a counterexample that helps $\mathfrak{A}^{P'}(\mathcal{T})$ to pass the equivalence test the next time.

**Lemma 4.25.** *If the two AFAs $\mathfrak{A}^{P}(\mathcal{T})$ and $\mathfrak{A}^{P'}(\mathcal{T})$ constructed in Line 8, respectively 10 satisfy the condition $L(\mathfrak{A}^{P}(\mathcal{T})) = L = L(\mathfrak{A}^{P'}(\mathcal{T}))$ then $\mathfrak{A}^{P}(\mathcal{T})$ is residual.*

*Proof.* Assume $L(\mathfrak{A}^{P}(\mathcal{T})) = L = L(\mathfrak{A}^{P'}(\mathcal{T}))$. Lemma 4.24 states that $\mathfrak{A}^{P'}(\mathcal{T})$ is residual. Consider a state $q = r_u$ of $\mathfrak{A}^{P}(\mathcal{T})$ with corresponding state $q'$ of $\mathfrak{A}^{P'}(\mathcal{T})$. As $r_u \in P \subseteq \mathrm{Rows}_{\mathrm{high}}(\mathcal{T}) = P'$, there is always such a corresponding state. Let $a \in \Sigma$ be any alphabet symbol. For every monomial $M' \sqsubset \tau(q', a)$, there is a monomial $M \sqsubset \tau(q, a)$ such that every literal of $M$ is in $M'$ (with the corresponding $v$ we have $M = M^P(v)$ and $M' = M^{P'}(v)$ and $M^{P'}(v)$ may consist of states not in $P$). Hence, $[\![\tau(q, w)]\!] \geq [\![\tau(q', w)]\!]$. From Lemma 4.24 one gets $u^{-1}L = u^{-1}L(\mathfrak{A}^{P'}(\mathcal{T})) \subseteq L(\mathfrak{A}^{P'}_{q'}(\mathcal{T}))$ and from Lemma 4.22 $L(\mathfrak{A}^{P}_{q}(\mathcal{T})) \subseteq u^{-1}L(\mathfrak{A}^{P}(\mathcal{T})) = u^{-1}L$. Thus, we get $u^{-1}L \subseteq L(\mathfrak{A}^{P'}_{q'}(\mathcal{T})) \subseteq L(\mathfrak{A}^{P}_{q}(\mathcal{T})) \subseteq u^{-1}L$ and $u^{-1}L = u^{-1}L(\mathfrak{A}^{P}(\mathcal{T})) = L(\mathfrak{A}^{P}_{q}(\mathcal{T}))$. Therefore, the automaton $\mathfrak{A}^{P}(\mathcal{T})$ is residual, too. □

A basis $P$ is called *optimal* for a regular language $L$ if its size is minimal over all bases $P$ for all tables $\mathcal{T}$ for $L$ such that $\mathfrak{A}^{P}(\mathcal{T})$ is an RAFA. The *reverse* of $L$ contains all strings $a_1 \dots a_k \in \Sigma^*$ such that $a_k \dots a_1$ is in $L$. Now we are ready to state the main result of this chapter.

**Theorem 4.26.** *For every regular language $L$, the algorithm* AL$^{\star\star}$ *always generates an RAFA $\mathfrak{A}^{P}$ such that $L(\mathfrak{A}^{P}) = L$. Moreover, if the basis $P$ is optimal then $\mathfrak{A}^{P}$ has the minimal number of states over all RAFAs for $L$.*
*The algorithm terminates after at most $\kappa_L$ equivalence queries and $\kappa_L\hat{\kappa}_L(1+|\Sigma|)\ell$ membership queries, where $\kappa_L$ and $\hat{\kappa}_L$ denote the number of states of the minimal DFA for $L$, respectively the reverse of $L$ and $\ell$ is the size of the longest counterexample obtained from the equivalence oracle. It runs in time bounded by a polynomial in $\kappa_L$, $\hat{\kappa}_L$, $|\Sigma|$, and $\ell$.*

Recall that, by definition of the concept class of regular languages, for every $d \in \mathbb{N}$ and every $L \in \mathcal{C}_d$, $\kappa_L \leq d$ and $\hat{\kappa}_L \leq d$.

*Proof.* Lemma 4.25 implies that the output of $\mathtt{AL}^{\star\star}$ is always residual. The number of states of the final hypothesis equals the size of the basis. Thus, an optimal basis leads to a minimal number of states. The table $\mathcal{T}$ cannot have more different rows than $\kappa_L$, the number of states of the minimal DFA for $L$ (compare Lemma 5 of [15]).

**Claim 4.27.** *$r_\lambda[v] = [\![\tau(Q_0, v)]\!]\,[\lambda]$ for every $v \in V$.*

*Proof.* Choose $\varphi = b^P(r_\lambda)$. By construction we have $r_\lambda = [\![b^P(r_\lambda)]\!]$. Now we can apply Lemma 4.21. □

**Claim 4.28.** *If $c \in \Sigma^*$ is classified incorrectly by the AFA then there exists a suffix $v$ of $c$ such that the corresponding column $v \notin V$ is different from all columns in $V$.*

*Proof.* The claim above shows that every $w \in V$ is classified correctly by $\mathfrak{A}_A^P(\mathcal{T})$ as well as by $\mathfrak{A}_A^{P'}(\mathcal{T})$. So, for every counterexample $c \in \Sigma^*$ we know that $c$ is not classified correctly by the current AFA, but will be classified correctly by every future AFA, which will be constructed from a table $\mathcal{T}' = (T', U', V')$ with $c$ in $V'$. Hence, every counterexample yields a change of $\tau$ at least for $\mathfrak{A}_A^{P'}(\mathcal{T})$. This can only be the case if either one of the new columns (added when seeing the counterexample) differs from all of the old columns, or if a new row is added to $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$. However, a new row is only added if the table has become non-$P$-closed. Therefore, a column that differs from every old column must have been added before. □

Thus, the maximal number of different columns is bounded by the minimal number of states of a DFA for the reserve language of $L$ denoted by $\hat{\kappa}_L$. Note that $\hat{\kappa}_L \leq 2^{\kappa_L}$. Thus both, $\mathfrak{A}_A^P(\mathcal{T})$ and $\mathfrak{A}_A^{P'}(\mathcal{T})$, must be equivalent to the unknown language $L$ after a finite number of counterexamples. Thus, $\mathtt{AL}^{\star\star}$ terminates.

By construction, $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$ does not contain a row more than once. So, $|U|$ is bounded by $\kappa_L$ and $\mathrm{Rows}(\mathcal{T})$ by $(1 + |\Sigma|)\,\kappa_L$. $V$ is bounded by the number of equivalence queries multiplied by the length of the counterexamples. Therefore, $|V| \leq \ell\,\hat{\kappa}_L$.

The size of the final table is thus at most $\kappa_L \hat{\kappa}_L (1 + |\Sigma|)\ell$, and also the number of membership queries. The total running time of $\mathtt{AL}^{\star\star}$ is polynomial in the size of the final table. □

### 4.4.1 Approximating the Minimum Basis

Assume $\mathcal{T} = (T, U, V)$ is a table for a regular language. Analogously to $\mathtt{AL}^{\star}$, $\mathtt{AL}^{\star\star}$ constructs a minimal basis, because computing a minimum basis is $\mathcal{NP}$-hard (see [7]). In order to guarantee that the basis (and hence the set of states) used by the algorithm is small enough, we give an approximation algorithm for this

problem. In the optimization problem MIN-SET-COVER, one is given a groundset $\mathcal{X}$ and a set $\mathcal{S}$ of subsets of $\mathcal{X}$ and searches the smallest $S \subseteq \mathcal{S}$ with $\bigcup_{s \in S} s = \mathcal{X}$ (see e. g. [65]). If $\mathcal{M}^P := \left\{ \left[\!\left[ M^P(v) \right]\!\right] : v \in V \right\}$ for $P \subseteq \mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$, we obtain the following lemma.

**Lemma 4.29.** *For every $P$ it holds: $\mathcal{M}^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})} = \mathcal{M}^P$ if and only if $P$ is a basis of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$.*

*Proof.* Assume that $P$ is a basis of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$, but $\mathcal{M}^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})} \neq \mathcal{M}^P$. By construction, for every $m = \left[\!\left[ M^P(v) \right]\!\right] \in \mathcal{M}^P$ there must be some $m' = \left[\!\left[ M^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})}(v) \right]\!\right] \in \mathcal{M}^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})}$ with $m' \leq m$. By assumption, there must be such a pair $m$, $m'$ with $m' < m$. Now consider $v, v' \in V$ such that $m = \left[\!\left[ M^P(v) \right]\!\right]$, $m' = \left[\!\left[ M^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})}(v) \right]\!\right]$ and $m'[v'] < m[v']$. There must be a row $r_u \in \mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$ with $r_u[v] = +$ and $r_u[v'] = -$. Note that $r_u \in M^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})}(v)$. From $\left[\!\left[ M^P(v) \right]\!\right][v'] = +$ we know that $P$ cannot contain such a row $r_u$. Thus, every monomial over $P$ that evaluates to $+$ at position $v$ must evaluate to $+$ at position $v'$. But then, $r_u \in \mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$ cannot be composed from $P$ by an MDNF. Thus, $P$ cannot be a basis of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$.

Now assume that $P$ is not a basis of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$. So there is some $u \in U$ such that $r_u$ cannot be expressed by an MDNF over $P$. Thus, there is a $v \in V$ with $u[v] = +$, but $\left[\!\left[ M^P(v) \right]\!\right] > r_u = \left[\!\left[ M^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})}(v) \right]\!\right]$. This implies $\mathcal{M}^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})} \neq \mathcal{M}^P$. □

We will now reduce the problem of finding a basis of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$ to the problem of finding a solution to a SET-COVER instance.

**Lemma 4.30.** *Let $\mathcal{X} = \{(v, i) : v, i \in V \wedge \left[\!\left[ M^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})}(v) \right]\!\right][i] = -\}$ be the groundset and $\mathcal{S} = \{m_u : u \in U\}$ with subsets $m_u = \{(v, i) \in \mathcal{X} : r_u \geq \left[\!\left[ M^{\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})}(v) \right]\!\right]$ and $r_u[i] = -\}$ be an instance of SET-COVER. The set $P$ is a basis of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$ if and only if there exists a feasible solution $\mathcal{C}$ of the set cover instance above such that $P = \{r_u : m_u \in \mathcal{C}\}$.*

*Proof.* Every vector of $\mathcal{M}^P$ can be composed by the vectors of $P$ by intersection, so requiring these compositions does not increase $P$. Now we apply the lemma above. □

We can now use the well known algorithm for the optimization problem MIN-SET-COVER due to [37] that on input $(\mathcal{X}, \mathcal{S})$ produces a feasible solution $S \subseteq \mathcal{S}$ with $|S| \leq (\ln(|\mathcal{X}|) + 1)|S^*|$ in polynomial time, where $S^*$ is an optimal solution to the instance. We get the following result.

**Theorem 4.31.** *There exists a polynomial time algorithm that for a given table $\mathcal{T} = (T, U, V)$ returns a basis $P$ of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$ with $|P| \leq (2\ln(|V|) + 1) \cdot |P^*|$, where $P^*$ is a minimum basis of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$.*

It is important to note here that $P^*$ is a minimum basis of $\mathrm{Rows}_{\mathrm{high}}(\mathcal{T})$ and does not necessarily correspond to an *optimal* basis. One can indeed construct tables $\mathcal{T}$ such that *no* basis $P \subseteq \mathrm{Rows}(\mathcal{T})$ is optimal.

### 4.4.2 Experimental Results

We have run $\mathtt{L}^\star$, $\mathtt{NL}^\star$, and $\mathtt{AL}^{\star\star}$ on random AFA targets. The first distribution of these random AFAs (RAT1) has been generated similar to the experiments of Angluin et al. [7] as told by Fisman [33].

- Every AFA has 7 states over an alphabet of size 3.
- The number of accepting states is chosen randomly between 1 and 3.
- The initial state and every transition are either disjunctions or conjunctions each.
- Each such formula consists of 1 up to 3 variables.
- Trivial AFAs are discarded.

For the equivalence oracle we have used the probabilistic (non-error free) equivalence oracle (REQ) described in [7] and also implemented an exact version (EEQ). Whenever REQ returned "equivalent", this has been verified by EEQ. In almost every run of $\mathtt{L}^\star$, $\mathtt{NL}^\star$, and $\mathtt{AL}^{\star\star}$, at least one wrong answer given by REQ has shown up. Thus, the following experiments have been obtained by using the exact algorithm EEQ. However, in about 50% of all non-trivial RAT1 instances EEQ has required so much computational power that the computation could not be finished. This problem is unlikely to be fixed by a more efficient implementation of EEQ, because AFA-equivalence is $\mathcal{PSPACE}$-hard (NFA-equivalence is already $\mathcal{PSPACE}$-complete [59]).

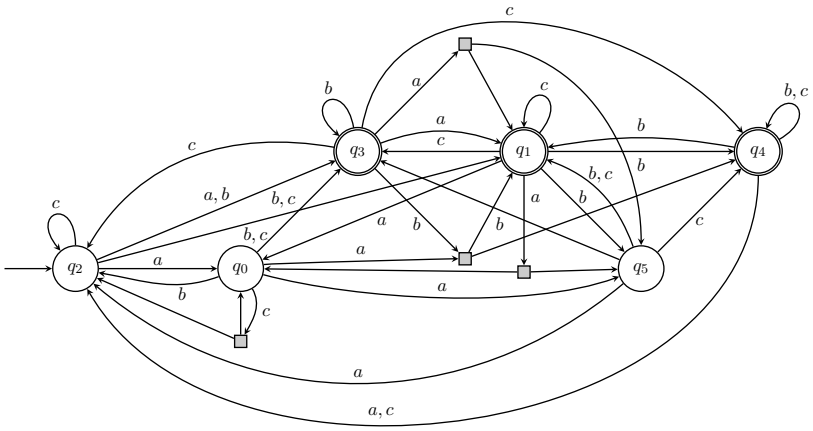


Figure 4.10: An example of a RAT2 instance.

Therefore, to reduce the computational complexity of the instances we have generated a different set of random AFA targets (RAT2) obtained as follows.

- Every AFA has 6 states over an alphabet of size 3.
- Every state is accepting with probability 1/2.
- With probability 1/3, there is exactly one initial state. Otherwise, the initial configuration is a disjunction of two different random states.
- Every transition is an MDNF formula, consisting of two monomials. Each monomial is a conjunction of random states. With probability 2/3, such a monomial is of size 1, otherwise of size 2.
- Trivial AFAs are discarded.

Figure 4.10 shows such a randomly generated target AFA. There were still about 24% non-trivial RAT2 instances we had to abort.

Figure 4.11 summarizes our experimental results with EEQ for RAT2 comparing the sizes of the automata generated by `L*`, `NL*` and `AL**`. Note that the target instances randomly generated may not be residual, while the AFAs output by `AL**` are always residual.

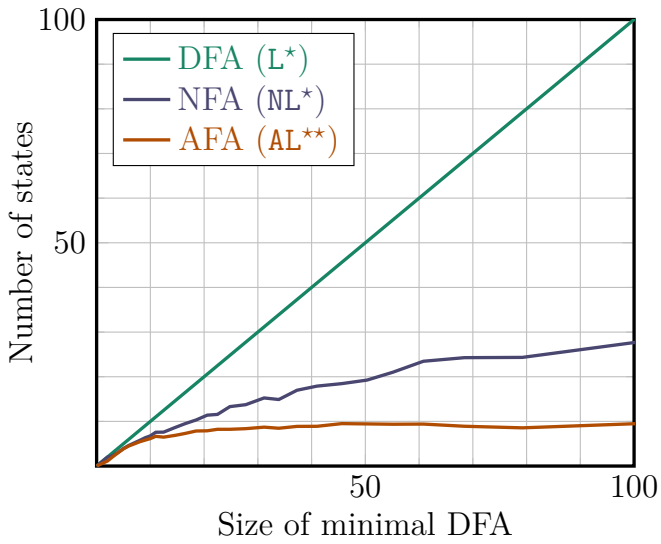


Figure 4.11: Comparison of the size of automata learned by `L*`, `NL*` and `AL**` for random regular languages generated by AFAs.

## 4.5 The Size of Residual AFAs

Angluin et al. [7] have shown that RAFAs may be exponentially more succinct than RNFAs and RUFAs and double exponentially more succinct than DFAs. We strengthen these results by proving that RAFAs may be exponentially more succinct than every equivalent *non-residual* NFAs or UFAs. Furthermore, there exists an RAFA that is double exponentially more succinct than the minimal DFA and uses only 2 nondeterministic (i. e. $\vee$) transitions and only a linear number

of universal (i. e. $\wedge$) transitions. Thus, the restriction to residual automata still allows a very compact representation. On the other hand, we give an example where the residuality of an automata demands an exponentially larger state set.

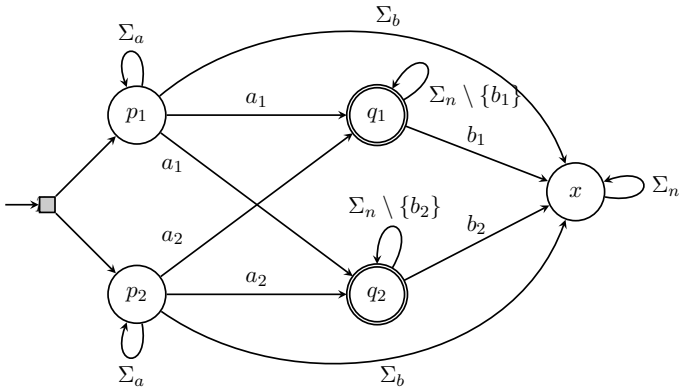


Figure 4.12: The residual AFA for the language $A_n$ of Theorem 4.32 with $n = 2$. The corresponding alphabet is $\Sigma_n = \Sigma_a \cup \Sigma_b$ with $\Sigma_a = \{a_1, a_2\}$ and $\Sigma_b = \{b_1, b_2\}$, the initial configuration is $Q_0 = p_1 \wedge p_2$, and the set of accepting states is $F = \{q_1, q_2\}$.

**Theorem 4.32.** *For every even $n \in \mathbb{N}$, there exists a language $A_n$ that can be accepted by a* residual *AFA with $2n + 1$ states and every NFA or UFA for $A_n$ needs at least $\binom{n}{n/2}$ states.*

*Proof.* The alphabet $\Sigma_n$ for $A_n$ consists of disjoint subsets $\Sigma_a = \{a_1, a_2, \ldots, a_n\}$ and $\Sigma_b = \{b_1, b_2, \ldots, b_n\}$. The language is defined as

$$A_n = \{w_1 w_2 : w_1 \in \Sigma_a^*, w_2 \in \Sigma_n^*, w_1 \textit{ contains all } \text{symbols from } \Sigma_a, \\ w_2 \text{ does } \textit{not contain all } \text{symbols from } \Sigma_b\} \ .$$

We construct a residual AFA with states $\{p_1, \ldots, p_n,\ q_1, \ldots, q_n, x\}$ that is sketched for $n = 2$ in Fig. 4.12. A general construction of AFAs $\mathfrak{A}^n$ for $A_n$ is given below:

- $Q^n = \{p_1, p_2, \ldots, p_n, q_1, q_2, \ldots, q_n, x\}$, $Q_0 = \bigwedge_{i=1}^n p_i$, $F = \{q_1, q_2, \ldots q_n\}$
- $\tau(p_i, a_i) = p_i \vee q_1 \vee q_2 \vee \ldots \vee q_n$
- $\tau(p_i, \sigma) = p_i$ for $\sigma \in \Sigma_a \setminus \{a_i\}$, $\tau(p_i, \sigma) = x$ for $\sigma \in \Sigma_b$, $\tau(q_i, b_i) = x$
- $\tau(q_i, \sigma) = q_i$ for all $\sigma \in \Sigma_n \setminus \{b_i\}$, $\tau(x, \sigma) = x$ for all $\sigma \in \Sigma_n$

Residuality follows from the following strings $u(q)$ for $q \in Q^n$ such that $L(\mathfrak{A}_q) = u(q)^{-1} A_n$:

$$u(x) = a_1 a_2 \ldots a_n\ b_1 b_2 \ldots b_n,$$
$$u(q_i) = a_1 a_2 \ldots a_n b_1 b_2 \ldots b_{i-1} b_{i+1} \ldots b_n,$$
$$u(p_i) = a_1 a_2 \ldots a_{i-1} a_{i+1} \ldots a_n.$$

In order to prove the lower bound for the size of NFAs for $A_n$, we use permutations on $\Sigma_a$. For a permutation $\pi$ on $\Sigma_a$ we define $S_0(\pi) = \{\pi(1), \pi(2), \ldots, \pi(n/2)\}$,

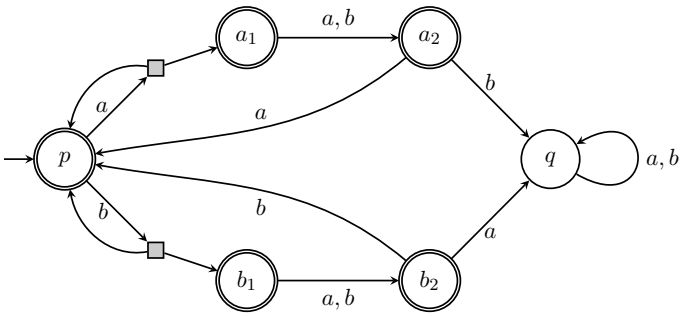


Figure 4.13: The (non-residual) AFA for the language $B_n$ of Theorem 4.33 with $n = 2$.

$S_1(\pi) = \{\pi(n/2+1), \ldots, \pi(n)\}$ and $w(\pi) = \pi(1)\pi(2)\ldots\pi(n)$. The string $w(\pi)$ contains each letter of $\Sigma_a$ exactly once and hence belongs to $A_n$.

Let $\mathfrak{A} = (Q, Q_0, \tau, F)$ be an NFA for $A_n$. For $w(\pi)$, consider an accepting computation generating the sequence of states $q_\pi^1, q_\pi^2, \ldots, q_\pi^n$ with $q_\pi^n \in F$. There are $\binom{n}{n/2}$ many pairs of permutations $\pi, \pi'$ with $S_0(\pi) \neq S_0(\pi')$. If $\mathfrak{A}$ has less than that many states there must exist two such permutations $\pi$ and $\pi'$ with $q_\pi^{n/2} = q_{\pi'}^{n/2}$. Hence,

$$q_\pi^1, \ldots, q_\pi^{n/2},\ q_{\pi'}^{n/2+1}, \ldots q_{\pi'}^n$$

is an accepting run of

$$w(\pi, \pi') := \pi(1)\pi(2)\ldots\pi(n/2)\ \pi'(n/2+1)\ldots\pi'(n)\ .$$

But, as $S_0(\pi) \neq S_0(\pi')$, there is some symbol $\sigma \in S_1(\pi)$ with $\sigma \notin S_1(\pi')$. Hence, $\sigma$ does not occur in $w(\pi, \pi')$ and thus $w \notin A_n$. Thus, $\mathfrak{A}$ does not recognize $A_n$ correctly.

The lower bound proof for UFAs is dual by taking permutations on $\Sigma_b$. Now a string $w(\pi)$ starts with $a_1 \ldots a_n$ to fulfill the first conditions and then continues with the permutation $\pi$ of the letters in $\Sigma_b$. All these strings do not belong to $A_n$, but omitting one letter $b_i$ in the second part puts the input into the language. □

**Theorem 4.33.** *For every $n \in \mathbb{N}$ there exists a language $B_n$ over a binary alphabet that can be accepted by a (non-residual) AFA with $2n+2$ states, but every residual AFA for $B_n$ requires at least $2^n$ states.*

*Proof.* We start with an auxiliary lemma. For an AFA $\mathfrak{A} = (Q, Q_0, F, \tau)$ and $\varphi \in \mathcal{F}(Q)$ let $\mathfrak{A}_\varphi = (Q, \varphi, F, \tau)$ denote the AFA starting with the initial configuration $\varphi$.

**Lemma 4.34.** *Let $L$ be a regular language and $\mathfrak{A}$ an AFA accepting $L$. For every $w \in \Sigma^*$, there is a formula $\varphi_w \in \mathcal{F}(Q)$ such that $L(\mathfrak{A}_{\varphi_w}) = w^{-1}L$.*

*Proof.* Suppose that this is wrong and there exists a $\widehat{w} \in \Sigma^*$ that for every $\varphi \in \mathcal{F}(Q)$, $L(\mathfrak{A}_\varphi) \neq \widehat{w}^{-1}L$. Hence, for every $\varphi \in \mathcal{F}(Q)$, there is a string $v_\varphi$ such that $v_\varphi \in L(\mathfrak{A}_\varphi) \triangle \widehat{w}^{-1}L$. This means that $\widehat{w}v_\varphi$ is wrongly classified by $\mathfrak{A}$. □

Let $\Sigma = \{a, b\}$ and consider $B_n = \{w^*w' \ : \ w \in \Sigma^n, w' \text{ is a prefix of } w\}$ (based on the construction of [63]). Let us construct a (non-residual) AFA for $B_n$:

- $Q = \{p, q, a_1, \ldots, a_n, b_1, \ldots, b_n\}$, $Q_0 = \{p\}$,
- $F = \{p, a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n\}$
- $\tau(p, a) = p \wedge a_1$, $\tau(p, b) = p \wedge b_1$
- $\tau(a_i, \sigma) = a_{i+1}$ for $i < n$ and $\sigma \in \Sigma$
- $\tau(b_i, \sigma) = b_{i+1}$ for $i < n$ and $\sigma \in \Sigma$
- $\tau(a_n, a) = p$, $\tau(b_n, b) = p$, $\tau(a_n, b) = q$, $\tau(b_n, a) = q$
- $\tau(q, \sigma) = q$ for all $\sigma \in \Sigma$

For $n = 2$, the non-residual AFA $\mathfrak{A} = (Q, Q_0, \tau, F)$ for $B_n$ is sketched in Fig. 4.13. It should not be too difficult to convince oneself that this AFA does the job.

To prove that every residual AFA accepting $B_n$ has at least $2^n$ states, let $S = \{u \ : \ u \in \{a, b\}^*, |u| \leq n\}$ be the set of strings of maximal length $n$. For $w = w_1 w_2 \ldots w_m$ in $B_n$, with $w_i \in \Sigma$ and $m > n$, we have that $w^{-1}B_n = (w_{m-n+1} w_{m-n+2} \ldots w_m)^{-1} B_n$ by the construction of $B_n$. Hence, for each residual language $L'$ of $B_n$, there is a string $u \in S$ such that $L' = u^{-1}B_n$. For $u, u' \in S$ with $u \neq u'$, we have either $u^{-1}B_n \subsetneq u'^{-1}B_n$, or $u'^{-1}B_n \subsetneq u^{-1}B_n$, or $u^{-1}B_n \cap u'^{-1}B_n = \emptyset$, due to the construction of $B_n$. Hence, there is a bijection between $\mathrm{RES}(B_n)$ and $S$.

Let $\mathfrak{A}$ be a residual AFA for $B_n$ with states $Q$. As $\mathfrak{A}$ is residual, every state $q \in Q$ corresponds to a residual language and thus to a string $u_q \in S$. Now consider a string $v \in \Sigma^n$. We have $(v^{-1}B_n) \cap \Sigma^n = \{v\}$. In order to correctly recognize $B_n$, one can see that there is a configuration $\varphi_v \in \mathcal{F}(Q)$ such that $L(\mathfrak{A}_{\varphi_v}) = v^{-1}B_n$ (see Lemma 4.34 above). Without loss of generality, suppose that $\varphi_v = \bigvee_{i=1}^k M_i$ and $M_i \subseteq Q$. Remember that for all residual languages, they are either disjoint or one of the languages is a subset of the other. Hence, for every $M_i$, either $L(\mathfrak{A}_{M_i}) = \emptyset$ (and it thus can be removed from $\varphi_v$) or there is a state $q_i \sqsubset M_i$ such that $L(\mathfrak{A}_{M_i}) = L(\mathfrak{A}_{q_i})$. Now $\varphi_v$ can be represented as a conjunction of states, i. e. $\varphi_v \equiv \bigvee_{i=1}^k q_i$. But, as $L(\mathfrak{A}_{\varphi_v}) = v^{-1}B_n$, we conclude that there is a single state $q_v \in Q$ such that $\varphi_v \equiv q_v$ (as the language of every other state is either disjoint or a proper superset). As all of these states need to be disjoint, we have $|Q| \geq |\Sigma|^n = 2^n$. □

A closer look at the constructions of succinct automata for $B_n$ reveals that the resulting AFAs are in fact UFAs. Dually, $\overline{B}_n = \Sigma^* \setminus B_n$ can be accepted by an NFA with the same number of states $2n + 2$. Thus, we obtain families of languages $B_n$ and $\overline{B}_n$ for $n = 1, 2, \ldots$, such that every residual AFA for $B_n$, respectively $\overline{B}_n$, is exponentially larger than the corresponding minimal UFA, respectively NFA.

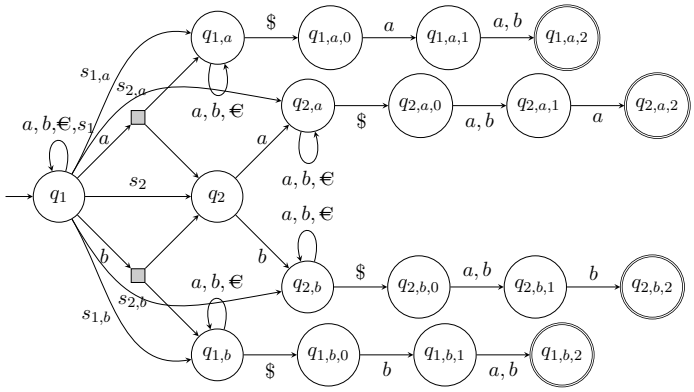


Figure 4.14: The AFA $\mathfrak{A}_n$ for $n = 2$.

As it has already been noted in [7], RAFAs may be double exponentially smaller than the minimal DFAs. We give a more precise bound inspired by a language defined in [17].

**Theorem 4.35.** *For every $n \in \mathbb{N}$ there exists a language $C_n$ such that the minimal DFA for $C_n$ needs at least $2^{2^n}$ states and there is a* residual *AFA with $2n^2 + 5n$ states for $C_n$.*

*Proof.* Consider the alphabet

$$\Sigma^n = \{a, b, €, \$\} \cup \{s_i : 1 < i \leq n\} \cup \{s_{i,\sigma} : 1 \leq i \leq n \wedge \sigma \in \{a, b\}\}.$$

For sake of simplicity, let $\mathcal{I} = \{i, (i, \sigma) : 1 \leq i \leq n, \sigma \in \{a, b\}\}$ be the indices of the alphabet symbols $s_i$ (respectively $s_{i,\sigma}$). The AFA $\mathfrak{A}_n = (Q, Q_0, \tau, F)$ is constructed as follows.

- $Q = \{q_i, q_{i,\sigma}, q_{i,\sigma,j} : 1 \leq i \leq n, \sigma \in \{a, b\}, 0 \leq j \leq n\}$, $Q_0 = q_1$
- $F = \{q_{i,\sigma,n} : 1 \leq i \leq n \wedge \sigma \in \{a, b\}\}$
- $\tau(q_1, €) = q_1$, $\tau(q_1, s_I) = q_I$ for all indices $I \in \mathcal{I}$
- $\tau(q_1, a) = (q_{1,a} \wedge q_2) \vee q_1$, $\tau(q_1, b) = (q_{1,b} \wedge q_2) \vee q_1$
- $\tau(q_i, a) = q_{i,a} \wedge q_{i+1}$, $\tau(q_i, b) = q_{i,b} \wedge q_{i+1}$ for $1 < i < n$
- $\tau(q_n, a) = q_{n,a}$, $\tau(q_n, b) = q_{n,b}$
- $\tau(q_{i,\sigma}, a) = \tau(q_{i,\sigma}, b) = \tau(q_{i,\sigma}, €) = q_{i,\sigma}$ for $1 \leq i \leq n$ and $\sigma \in \{a, b\}$
- $\tau(q_{i,\sigma}, \$) = q_{i,\sigma,0}$ for $1 \leq i \leq n$ and $\sigma \in \{a, b\}$
- $\tau(q_{i,\sigma,i-1}, \sigma) = q_{i,\sigma,i}$ for $1 \leq i \leq n$ and $\sigma \in \{a, b\}$
- $\tau(q_{i,\sigma,j}, a) = \tau(q_{i,\sigma,j}, b) = q_{i,\sigma,j+1}$ for $1 \leq i \leq n$, $j \neq i - 1$, and $\sigma \in \{a, b\}$
- $\tau(q, \sigma) = \bot$ for every $q \in Q$ and $\sigma \in \Sigma^n$ such that $\tau(q, \sigma)$ has not been defined above, where $\bot$ is the empty MDNF.

The corresponding automaton $\mathfrak{A}_2$ is shown in Fig. 4.14.

Note that $\mathfrak{A}_n$ has $n + 2n + 2n(n+1) = 2n^2 + 5n$ states and its transitions are of polynomial size. It accepts the language

$$C_n = \{uwv\$w : u, v \in \{a, b, €\}^* \wedge w \in \{a, b\}^n\} \cup \{us_I v_I : \ u \in \{0, 1, €\}^* \wedge I \in \mathcal{I} \wedge v_I \in L((\mathfrak{A}_n)_{q_I})\}.$$

This language is inspired by [17]. It remains to show that $\mathfrak{A}_n$ is residual, and that $C_n$ has at least $2^{2^n}$ different Nerode classes.

1. For every index $I \in \mathcal{I}$ it holds $L\left((\mathfrak{A}_n)_{q_I}\right) = (s_I)^{-1}C_n$, because $\tau(Q_0, s_I) = q_I$. For $q_{i,\sigma,j}$ with $1 \leq i \leq n$ and $\sigma \in \{a, b\}$, $0 \leq j \leq n$, consider the set $\{w_1, \ldots, w_{2^{n-1}}\} = \{a, b\}^{i-1}\sigma\{a, b\}^{n-i}$. Since $L\left((\mathfrak{A}_n)_{q_{i,\sigma,j}}\right) = (w_1€w_2€\ldots€w_{2^{n-1}}\$\sigma^j)^{-1}C_n$ the AFA $\mathfrak{A}_n$ is residual.
2. For every subset $W = \{w_1, \ldots, w_\ell\}$ of $\{a, b\}^n$ , $(w_1€w_2€\ldots€w_\ell\$)^{-1}C_n = W$. Thus, the number of different Nerode classes of $C_n$ is at least $2^{2^n}$.

□

The tables below are showing the results presented in this section. Here

| $\mathfrak{A}_1$ | $\mathfrak{A}_2$ |
|---|---|
| $k_1(n)$ | $k_2(n)$ |

has the following meaning: For every $n$ there exists a language $L_n$ with a $k_1(n)$ state automata of type $\mathfrak{A}_1$ and every automaton of type $\mathfrak{A}_2$ for $L_n$ needs at least $k_2(n)$ states.

| RAFA | NFA/UFA |
|---|---|
| $2n+1$ | $\binom{n}{n/2}$ |

| NFA/UFA | RAFA |
|---|---|
| $2n+2$ | $2^n$ |

| RAFA | DFA |
|---|---|
| $2n^2+5n$ | $2^{2^n}$ |

In this chapter the conjecture that the algorithm $\texttt{AL}^\star$ generates residual AFAs only has been disproved. A modified algorithm $\texttt{AL}^{\star\star}$ that achieves this property has been designed. Experiments with randomly generated automata show that $\texttt{AL}^{\star\star}$ typically generates automata that are significantly more succinct than equivalent minimal DFAs and RNFAs.

# 5 Learning Density Levels of Distributions

*Density level sets*, or *density levels*, are high probability areas of probability distributions. Estimating density level sets has been studied widely from a statistical point of view [36, 48, 50, 60]. These works from statistical research area develop density level set estimators and estimate their convergence, but algorithmic learning is not considered. Especially, convergence of the estimators in the limit is proven, but not the computational and sample complexity with respect to accuracy requirements. On the other hand, Kearns et al. [40] have considered learning probability distributions in the spirit of PAC learning. Their framework is restricted to discrete distributions that are generated by specific kinds of circuits. When it comes to more natural distributions, Ashtiani et al. [10] have shown a new bound on the sample complexity for algorithmically learning mixtures of Gaussian distributions. For learning a much wider class of natural distributions, Ben-David and Lindenbaum [11] have developed an approach, in the following called BL framework, to identify high-density areas of distributions. The authors consider the problem as unsupervised learning, where the density level is the unknown concept to be learned. They have defined requirements on the accuracy based on the PAC framework and obtained lower and upper bounds for the sample complexity. These bounds are polynomial in the VC dimension of the class of density level sets and the inverse of the error bounds. For learning without time constraints, this implies learnability from a finite number of examples if and only if the VC dimension of the density level sets is finite.

Let $P$ be a probability distribution with probability density function $\mu_P$. For $s \in \mathbb{R}_+$, the density level $\mathrm{Lev}(\mu_P, s) := \{x \in \mathcal{X} : \mu_P(x) \geq s\}$ describes a region of high probability. These levels are the targets to be learned. This kind of question arises if one wants to identify high-risk groups in a population, as well as in social studies, marketing analysis, pattern recognition, and computer vision [11].

Ben-David and Lindenbaum have defined a consistency criterion for learning algorithms that is sufficient for an algorithm to be a successful learner. However, it looks difficult to determine whether this criterion is satisfied. So far we could apply this setting only in the one-dimensional case [28]. Even worse, the application of their consistency criterion requires density level sets around the target to be part of the concept class. To be specific, the levels $\mathrm{Lev}(\mu_P, s - \rho)$ and $\mathrm{Lev}(\mu_P, s + \rho)$ must occur in the concept class, where $\rho > 0$ denotes an arbitrary error parameter. In consequence, since learning must work for every

choice of the error parameter $\rho > 0$, the concept class must cover all density levels $\mathrm{Lev}(\mu_P, s)$ for every $s \in \mathbb{R}_+$. This removes an advantage of density level learning: the learnability of a specific level even if the probability distribution is too complicated to be learned at other levels.

We build on the technique of empirical excess mass maximization used in statistics, which has been developed by Hartigan [36] and Müller and Sawitzki [48] independently. It is shown that this measure can be made compatible with the BL framework. As a consequence, a wide range of already existing density level estimators are proven to be successful learning algorithms in the BL framework. Additionally, we can improve the upper sample bound by a constant factor.

## 5.1 The Learning Model

In this chapter we fix a dimension $d$ and write $\mathcal{C} = \mathcal{C}_d$, $\mathcal{X} = \mathcal{X}_d$, and $\mathcal{H} = \mathcal{H}_d$. Let $\mathcal{M} = (\mathcal{X}, \mathcal{B})$ be a measure space consisting of a set $\mathcal{X}$ and a $\sigma$-algebra $\mathcal{B}$ on $\mathcal{X}$ with the Lebesgue measure $\lambda$, i. e. for all $X \in \mathcal{B}$, $\lambda(X) = \int_X 1 \, \mathrm{d}x$.

For a function $r : \mathcal{X} \to \mathbb{R}_+$ and some threshold $s \in \mathbb{R}_+$, the $s$-level of $r$ is the set

$$\mathrm{Lev}(r, s) := \{x \in \mathcal{X} : r(x) \geq s\} \ .$$

Additionally, define

$$\mathrm{Lev}^{>}(r, s) \ := \ \{x \in \mathcal{X} : r(x) > s\} \ .$$

Let $\mathcal{R}$ denote the set of all functions $r : \mathcal{X} \to \mathbb{R}_+$ that satisfy $\mathrm{Lev}(r, s) \in \mathcal{B}$ for every $s \in \mathbb{R}_+$.

**Definition 5.1.** *For $s, \varepsilon, \rho \in \mathbb{R}_+$ and $r \in \mathcal{R}$, a measurable set $h \in \mathcal{B}$ is $(\varepsilon, \rho)$-close to the set* $\mathrm{Lev}(r, s)$ *if*

$$\lambda(h \setminus \mathrm{Lev}(r, s - \rho)) \ + \ \lambda(\mathrm{Lev}^{>}(r, s + \rho) \setminus h) \ \leq \ \varepsilon \ .$$

For an illustration of $(\varepsilon, \rho)$-closeness see Fig. 5.1.

Now let $P$ be a probability distribution on $\mathcal{M}$ with probability density function $\mu_P : \mathcal{X} \to \mathbb{R}_0^+$. Learning the entire density function $\mu_P$ of an unknown distribution $P$ from a sample drawn according to $P$ may be impossible depending on the properties of $P$ (see Section 5.2 for a discussion). Thus, instead we want to solve the easier problem of identifying high-probability regions of $P$ – the $s$-levels of $\mu_P$ for some fixed $s \in \mathbb{R}_+$.

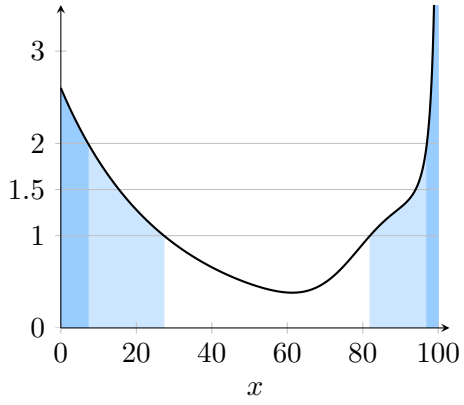


Figure 5.1: To be $(0, 0.5)$-close to $\mathrm{Lev}(r, 1.5)$ a hypothesis must include the deep blue, but exclude the white regions. The light blue regions do not matter. For $(\varepsilon, 0.5)$-closeness, a small percentage $\varepsilon$ of misclassification is allowed within these mandatory, respectively forbidden regions.

Let $\mathcal{P}$ be a family of probability distributions on $\mathcal{M}$. A concept class $\mathcal{C} \subseteq \mathcal{B}$ is *compatible* with the $s$-levels of $\mathcal{P}$ if

$$\mathcal{P} \subseteq \mathcal{D}(\mathcal{C}, s) := \{P : P \text{ is a probability distribution on } \mathcal{M}, \\ \mu_P \in \mathcal{R}, \text{ and } \mathrm{Lev}(\mu_P, s) \in \mathcal{C}\} \ .$$

A learner for the density levels of $\mathcal{P}$ with a compatible concept class $\mathcal{C}$ for some level $s \in \mathbb{R}_+$ is told the parameter $s$ and then given random samples from $\mathcal{X}$ distributed according to $P \in \mathcal{P}$.

**Definition 5.2.** *A learning algorithm $\mathcal{A}$ for density levels with hypothesis space $\mathcal{H} \subseteq \mathcal{B}$ is $(m, \varepsilon, \delta, \rho)$-successful for $\mathcal{C}$ and $s \in \mathbb{R}_+$ if for every $P \in \mathcal{P} \subseteq \mathcal{D}(\mathcal{C}, s)$ after drawing $m$ random samples according to $P$, with probability at least $(1 - \delta)$ it outputs a hypothesis $h \in \mathcal{H}$ that is $(\varepsilon, \rho)$-close to $\mathrm{Lev}(\mu_P, s)$ according to the Lebesgue measure $\lambda$.*

As in the previous chapters, we require the sample complexity of a learning algorithm to be bounded by a polynomial in the inverse error parameters $(\varepsilon \cdot \rho)^{-1}$ and $\delta^{-1}$, and in the dimension $d$ of the concept class $\mathcal{C}$. Here, we use the VC dimension $d = \text{VC-dim}(\mathcal{C})$. For an *efficient* learning algorithm, the running time also must be bounded by such a polynomial.

As an example, consider a univariate probability distribution whose density function has at most $k$ local maxima. For such a distribution, the concept class of its density levels is the $k$-fold union of intervals. This concept class has VC dimension $2k$. Now let us consider an example in the bivariate case. The density levels of bivariate Gaussian distributions form ellipses. The VC dimension of the concept class of ellipses is at least 5, because the five points at the corners

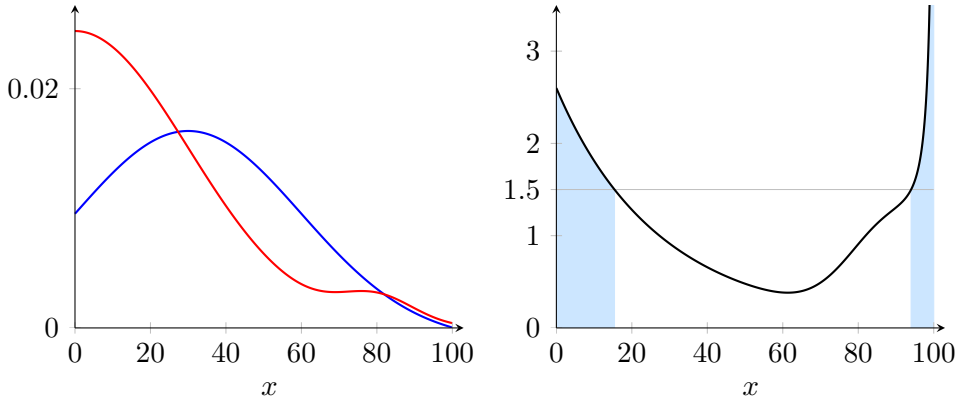


Figure 5.2: *Left:* The probability density function of the known distribution of age $\mu_D$ over the basic population (blue) and the probability density function $\mu_P$ over infected individuals (red), which is used for sampling.
*Right:* The conditional infection risk $r$ and the corresponding high-risk density level set $\text{Lev}(r, 1.5) = [0; 15.41] \cup [94; 100]$ that should be identified by the learner. High density levels occur where $P$ exceeds $D$ significantly. For high-density regions of $D$, very high density of $P$ is required (young people), whereas a low density of $P$ is already sufficient within rare areas according to $D$ (old people).

of a regular pentagon can be shattered by ellipses. This argument cannot be extended to six points, thus the VC dimension of ellipses is 5 [2].

Now that we have seen a basic model for learning density levels, we describe the BL framework, a generalization of this model. Instead of the Lebesgue measure $\lambda$, an arbitrary measure $D$ with density function $\mu_D \in \mathcal{R}$ is used. Further, we are now interested in the $s$-levels of $r = \mu_P/\mu_D$ instead of $\mu_P = \mu_P/\mu_\lambda$. An application that uses a probability measure $D$ may be the following. Let $D$ be the distribution of age ($\mathcal{X} = \{0, 1, \ldots, 100\}$) within a population and $P$ the distribution of age restricted to people that suffer from a certain disease. The value $r(x)$ quantifies whether the group of $x$-years old people has a smaller or larger portion of infected people compared to its proportion within the whole population.

In this setting, the measure $D$ is known to the learner and samples are drawn according to $P$. If we were interested to learn the density levels of $\mu_P$, we would not need the generalization that introduces the measure $D$. In this generalized setting, we want to find areas of high probability compared to the expected probability within the population, i. e. the density levels of $\mu_P/\mu_D$. See Fig. 5.2 for an illustration.

**Definition 5.3.** *For $s, \varepsilon, \rho \in \mathbb{R}_+$ and $r \in \mathcal{R}$, a measurable set $h \in \mathcal{B}$ is $(\varepsilon, \rho)$-close to the set* $\mathrm{Lev}(r, s)$ *with respect to the measure $D$ if*

$$D(h \setminus \mathrm{Lev}(r, s-\rho)) \;+\; D(\mathrm{Lev}^{>}(r, s+\rho) \setminus h) \;\leq\; \varepsilon \;.$$

In this more general setting, a concept class $\mathcal{C} \subseteq \mathcal{B}$ is *compatible* with the $s$-levels of $\mathcal{P}$ with respect to $D$ if

$$\mathcal{P} \subseteq \mathcal{D}_D(\mathcal{C}, s) := \{P : P \text{ is a probability distribution on } \mathcal{M}, \\ \mu_P/\mu_D \in \mathcal{R}, \text{ and } \mathrm{Lev}(\mu_P/\mu_D, s) \in \mathcal{C}\} \;.$$

Further, let

$$\mathcal{D}_D^{>}(\mathcal{C}, s) := \{P : P \text{ is a probability distribution on } \mathcal{M}, \\ \mu_P/\mu_D \in \mathcal{R}, \text{ and } \mathrm{Lev}^{>}(\mu_P/\mu_D, s) \in \mathcal{C}\} \;.$$

$\mathcal{C}$ is *$\rho$-compatible* to $\mathcal{P}$ if

$$\mathcal{P} \subseteq \mathcal{D}_D^{\pm\rho}(\mathcal{C}, s) := \mathcal{D}_D(\mathcal{C}, s-\rho) \cap \mathcal{D}_D(\mathcal{C}, s) \cap \mathcal{D}_D^{>}(\mathcal{C}, s+\rho) \;.$$

We assume that the learner for density levels has complete knowledge about $D$. The learner is told the parameter $s$ and then given random samples from $\mathcal{X}$ distributed according to $P \in \mathcal{P}$.

**Definition 5.4.** *Let $D$ be a fixed measure. A learning algorithm $\mathcal{A}$ for density levels with hypothesis space $\mathcal{H} \subseteq \mathcal{B}$ is $(m, \varepsilon, \delta, \rho)$-successful for $\mathcal{C}$ and $s \in \mathbb{R}_+$ if for every every unknown distribution $P \in \mathcal{P} \subseteq \mathcal{D}_D(\mathcal{C}, s)$ after drawing $m$ random samples according to $P$, with probability at least $(1-\delta)$ $\mathcal{A}$ outputs a hypothesis $h \in \mathcal{H}$ that is $(\varepsilon, \rho)$-close to* $\mathrm{Lev}(\mu_P/\mu_D, s)$ *according to $D$.*

As in the specific scenario, in the general setting we also require the sample complexity of a learning algorithm to be bounded by a polynomial in the inverse error parameters $(\varepsilon \cdot \rho)^{-1}$ and $\delta^{-1}$, and in the VC dimension of $\mathcal{C}$.

## 5.2 Learning Density Levels Versus the Complete Density Function

The learning techniques for density level set estimation discussed in the literature can be divided into two different approaches: plug-in and direct. To our knowledge, the term "plug-in estimator" has been introduced by Tsybakov [60] initially. If the plug-in method is used, an estimation $g$ of the entire probability density function $\mu_P$ is computed first. Afterwards, the $s$-level $\mathrm{Lev}(\mu_P, s)$ is estimated by $\mathrm{Lev}(g, s)$. On the other hand, the direct method denotes estimation of density levels without estimating the probability density function first.

The BL framework, which forms the basis of the research in this chapter, is a direct method for learning density levels. The main advantage of the direct

method is that only the complexity of the density level to be learned is relevant. Thus, density levels can be estimated even if the density function is arbitrarily complicated at other levels and cannot be estimated therefore.

However, despite mentioning of this advantage by Ben-David and Lindenbaum, the learnability within their framework depends on the VC dimension of every density level (to be more precise, the levels whose VC dimension has to be taken into account depend on the choice of the error parameter $\rho$), but not only on the VC dimension of the density level to be learned. Thus, the BL framework does not fully provide this advantage. This is a consequence of the consistency criterion, which we will get rid off in Section 5.4 yielding learnability of density levels independent of the complexity of other levels.

Given two density functions $f : \mathcal{X} \to \mathbb{R}_+$ and $g : \mathcal{X} \to \mathbb{R}_+$, their distance with respect to a measure $D$ is defined by

$$L_D^1(f,g) := \int_{\mathcal{X}} |f(x) - g(x)| \cdot \mu_D(x) \, \mathrm{d}x \ .$$

By Claim 1 of Ben-David and Lindenbaum [11], for two functions $f$, $g$ with $L_D^1(\mu_P, g) \leq \varepsilon$, and every $s \in \mathbb{R}_+$, the density level $\mathrm{Lev}(g,s)$ is $(\varepsilon,\rho)$-close to the level $\mathrm{Lev}(f,s)$ with respect to $D$. Thus, successful approximation of a probability density function $\mu_P$ implies density level learnability of $\mu_P$. It is not clear whether the other direction holds as well, i. e. whether density level learning of $\mu_P$ can be used to find a function $g$ that satisfies $L_D^1(\mu_P, g) \leq \varepsilon$.

Polonik [52] has shown that if density levels are estimated by excess mass maximization (see Section 5.4), the density function can be approximated as well. In Section 5.4, we show that excess mass maximization can be used to obtain a density level learner in the BL framework. However, the framework is more general and allows other methods of learning as well, e. g. consistent learning algorithms (see Section 5.3). The hypotheses obtained by excess mass maximization are monotone with respect to the $s$-level, which is exploited by Polonik. This kind of monotonicity is not guaranteed in the BL framework. In [11] it is claimed:

> *As it turns out, most of the results of this paper can be readily extended to apply to learning density functions in the $L_1$ norm.*

We have tried to do this, but it is not at all obvious to us how this can be proven formally since we have to face a lot of technical difficulties. In the remainder of this section for Lebesgue measure $D = \lambda$ an algorithm is designed that, with probability at least $1 - \delta$, computes an approximation $g$ of $\mu_P \,/\, \mu_D = \mu_P$ satisfying $L^1(\mu_P, g) := L_\lambda^1(\mu_P, g) \leq \varepsilon$ by repeatedly calling a density level learner for $\mu_P$. Our algorithm can be generalized to work with arbitrary measures $D$ on $(\mathcal{X}, \mathcal{B})$.

The approximation function $g$ may not be a probability density function. However, this can be fixed easily by defining a function $h : \mathcal{X} \to \mathbb{R}_+$ with

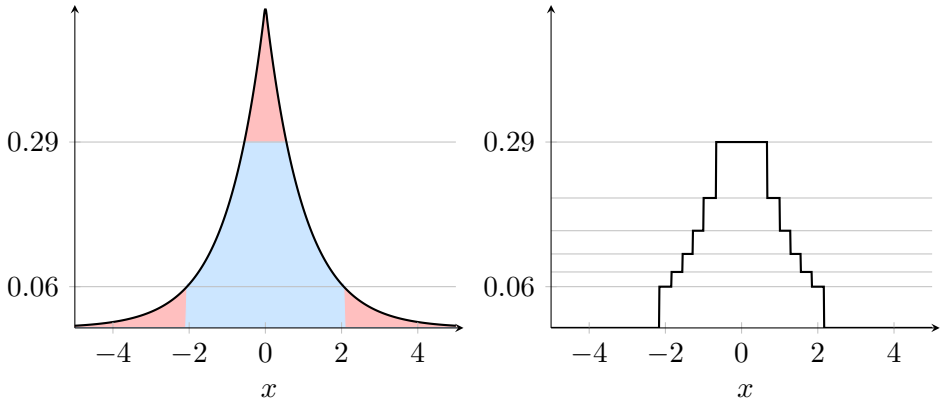


Figure 5.3: Approximation of the probability density function $\mu_P$ by appropriately selected density levels.
*Left:* The learning range of levels $[0.06, 0.29]$ is marked blue. The area of the density function $\mu_P$ that is not approximated due to the restriction of the learning range is marked red. The error r-error $(\mu_P, 0.06, 0.29)$ is the weight of this red area.
*Right:* Approximation of $\mu_P$ within the range $[0.06, 0.29]$ by density levels. The distance of the lower levels has to be smaller than the distance of the higher levels.

$h(x) = g(x)/\left(\int_{\mathcal{X}} g(x)\,\mathrm{d}x\right)$ for $x \in \mathcal{X}$. If $L^1(\mu_P, g) \leq \varepsilon$ for $0 < \varepsilon < 1$, then $h$ is a probability density function that satisfies $L^1(\mu_P, h) \leq \varepsilon/(1-\varepsilon)$.

**Observation.** *Lower $s$-levels of $\mu_P$ may include a wider area of $\mathcal{X}$ than higher levels, because for $s \in \mathbb{R}_+$, $\lambda(\mathrm{Lev}(\mu_P, s)) \leq s^{-1}$.*

The approximation algorithm can only learn a finite number of levels. By the observation, the distance between the levels has to be smaller for lower levels. As a result, computing an approximation $g$ requires not only an upper bound $s_{\max}$, but also a lower bound $s_{\min} > 0$ for the range $s$ of the $s$-levels learned. This is illustrated in the right subfigure of Fig. 5.3.

The approximation algorithm for $\mu_P$ works as follows. It searches the minimum and the maximum density level needed for a good approximation incrementally. For that purpose, it calls a subroutine that approximates $\mu_P$ within a given range of levels $[s_{\min}, s_{\max}]$. The range is extended until the error bound guaranteed by the subroutine is small enough.

We start by introducing the subroutine `ApproxRange` as Algorithm 5. For the function $g : \mathcal{X} \to \mathbb{R}_+$, a set $U \subseteq \mathcal{X}$, and a value $s \in \mathbb{R}_+$, we write $g[U] \leftarrow s$ to

denote that $s$ is assigned to $g(u)$ for every $u \in U$. The limitation of the learning range of the algorithm yields an error

$$\text{r-error}\,(\mu_P, s_{\min}, s_{\max}) := \int_{\mathcal{X}\setminus \mathrm{Lev}(\mu_P, s_{\min})} \mu_P(x)\,\mathrm{d}x + \int_{\mathrm{Lev}(\mu_P, s_{\max})} (\mu_P(x) - s_{\max})\,\mathrm{d}x \ .$$

This error is illustrated in the left subfigure of Fig. 5.3. Using the r-error, we will bound the approximation error of `ApproxRange` by $L^1\,(\mu_P, g) \leq \varepsilon +$ r-error $(\mu_P, s_{\min}, s_{\max})$ to guarantee that a good hypothesis is found if it exists.

However, the density function $\mu_P$ must be known to compute the r-error. Thus, this error bound does not help to decide whether the range of density levels needs to be extended and `ApproxRange` has to be called again. For that purpose, we establish a second error bound $L^1\,(\mu_P, g) \leq 2\varepsilon + 1 - \int_{\mathcal{X}} g(x)\,\mathrm{d}x$, depending on the approximation $g$ returned by the subroutine instead of r-error $(\mu_P, s_{\min}, s_{\max})$.

**Input:** error parameter $\varepsilon$, confidence parameter $\delta$, learning range of levels $[s_{\min}, s_{\max}]$, density level learner for $f$ `Learn`$(\cdot,\cdot,\cdot,\cdot)$
**Output:** approximation $g$ of function $\mu_P$

$g[\mathcal{X}] \leftarrow 0$;
$s_0 \leftarrow 0$;
$s_1 \leftarrow s_{\min} \,/\, 2$;
$i \leftarrow 1$;
**while** $s_{i-1} < s_{\max}$ **do**
  $\delta_i \leftarrow \delta/(2i^2)$;
  $\rho_{s_i} \leftarrow s_i\,\varepsilon/5$;
  $\varepsilon_{s_i} \leftarrow s_i^{-1}\,\varepsilon/(8i^2)$;
  $h[s_i] \leftarrow$ `Learn`$(s_i, \varepsilon_{s_i}, \delta_i, \rho_{s_i})$;
  $g[h[s_i]] \leftarrow s_i$;
  $s_{i+1} \leftarrow s_i \cdot (1 + \varepsilon/5)$;
  $i \leftarrow i + 1$;
**return** $g$;

**Algorithm 5:** `ApproxRange`$(\varepsilon, \delta, s_{\min}, s_{\max}, \texttt{Learn})$

**Lemma 5.5.** *Consider a probability density function $\mu_P$ such that every level $s \in \mathbb{R}_+$ of $\mu_P$ is density level learnable. For every $0 < s_{\min} \leq s_{\max} \in \mathbb{R}$ and every $\varepsilon > 0$ and $\delta > 0$, with probability at least $1 - \delta$,* `ApproxRange` *computes an approximation $g$ that satisfies*

$$L^1\,(\mu_P, g) \leq \min\left\{\varepsilon + \text{r-error}\,(\mu_P, s_{\min}, s_{\max}), 2\varepsilon + 1 - \int_{\mathcal{X}} g(x)\,\mathrm{d}x\right\}$$

*in time $\mathcal{O}(\varepsilon^{-1}\ \log(s_{\max}\,/\,s_{\min})\ T)$, where $T$ is the worst case running time of the density level learner. The density level learner is called repeatedly with different parameters $\varepsilon_{s_i}$, $\delta_i$, $\rho_{s_i}$. Both $(\varepsilon_{s_i} \cdot \rho_{s_i})^{-1}$ and $\delta_i^{-1}$ are always bounded by a polynomial in $\varepsilon$, $\delta$, and $\log(s_{\min}\,/\,s_{\max})$.*

*Proof.* Consider a run of `ApproxRange` on the probability density function $\mu_P$ with parameters $\varepsilon$, $\delta$, $s_{\min}$, and $s_{\max}$. There are two different scenarios.

**Case 1:** At least one time when the density level learner is called in Line 9 of `ApproxRange`, a hypothesis $h[s_i]$ is returned that is not $(\varepsilon_{s_i}, \rho_{s_i})$-close to $\mathrm{Lev}(\mu_P, s_i)$.

Let $\delta_i$ denote the values assigned to the corresponding variables during the run of the algorithm. The probability that Case 1 actually occurs is bounded by

$$\sum_{i=1}^{\infty} 2\,\delta_i = \sum_{i=1}^{\infty} \delta \,/\, (2i^2) = \delta\,\pi^2 \,/\, 12 < \delta \ .$$

**Case 2:** Every time the density level learner is called in Line 9 of `ApproxRange`, a hypothesis $h[s_i]$ is returned that is actually $(\varepsilon_{s_i}, \rho_{s_i})$-close to $\mathrm{Lev}(\mu_P, s_i)$.

For arbitrary $k \in \mathbb{Z}$, let $s_k = s_{\min}/2 \cdot (1 + \varepsilon/5)^{k-1}$ and $\rho_{s_k} = s_k \cdot \varepsilon/5$. Further let $\ell$ denote the number of iterations of the while loop in Line 5 during the run of `ApproxRange`. For $k \in [\ell]$ the values $s_k$ and $\rho_{s_k}$ as defined here actually occur during the run of the algorithm, but not for $k \notin [\ell]$. Note that $s_\ell \geq s_{\max}$.

Let $g : \mathcal{X} \to \{0\} \cup \bigcup_{k=-\infty}^{\infty} \{s_k\}$ denote the function returned at the end of the run of of `ApproxRange`. In the following we define three functions $g_1$, $g_2$, and $g_3$ in order to show a bound for $L^1(\mu_P, g)$ using the triangle inequation

$$L^1(\mu_P, g) \leq L^1(g, g_1) + L^1(g_1, g_2) + L^1(g_2, g_3) + L^1(g_3, \mu_P) \ .$$

An example of the functions defined below is visualized in Fig. 5.4.

The first function to be defined is $g_1$ that eliminates the error $\varepsilon$ of the density level learner from $g$. Let $g_1 : \mathcal{X} \to \{0\} \cup \bigcup_{k=-\infty}^{\infty} \{s_k\}$ denote the function returned by the following virtual run of `ApproxRange`. The run of the algorithm is modified such that each time a hypothesis $h[s_i]$ is constructed in Line 5 of the while loop, it is changed to

$$h'[s_i] := (h[s_i] \cup \mathrm{Lev}(\mu_P, s_i + \rho_{s_i})) \cap \mathrm{Lev}(\mu_P, s_i - \rho_{s_i}) \ .$$

This transforms $h[s_i]$ into the $(0, \rho_{s_i})$-close hypothesis $h'[s_i]$ with minimum distance $\lambda(h[s_i] \triangle h'[s_i])$ over all $(0, \rho_{s_i})$-close hypotheses, where $\triangle$ denotes the symmetric difference of sets.

The function $g_3$ eliminates all errors of $g$, except the stairstepping that is established by discrete density level learning. Let the function $g_3 : \mathcal{X} \to \{0\} \cup \bigcup_{k=-\infty}^{\infty} \{s_k\}$ be defined by

$$g_3(x) := \begin{cases} \max\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) \geq s_k\} & \text{if } \mu_P(x) > 0 \\ 0 & \text{if } \mu_P(x) = 0 \ . \end{cases} \tag{5.1}$$

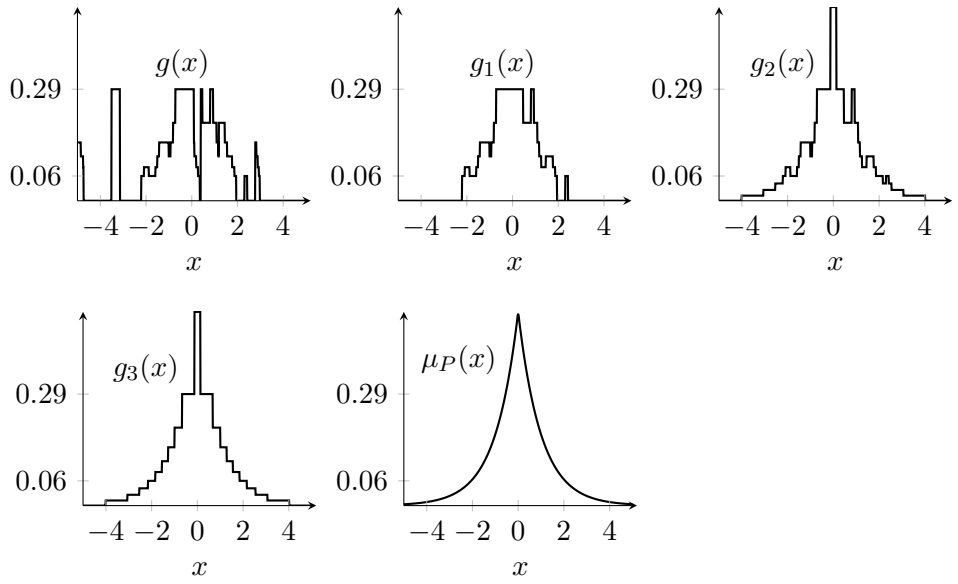


Figure 5.4: Construction of $g_1$, $g_2$, and $g_3$ for an example run of `ApproxRange` on $\mu_P$ that returned $g$. The error $\varepsilon_{s_i}$ of the density level learner is chosen quite large to improve its visibility.

The last approximation function to be defined, $g_2$, removes the r-error from $g_1$. The value of $g_1(x)$ within regions that are cut by $s_1$ (or $s_\ell$, respectively) are changed to a reasonable approximation $g_3(x)$ of $\mu_P(x)$. Define $g_2 : \mathcal{X} \to \{0\} \cup \bigcup_{k=-\infty}^{\infty} \{s_k\}$ by

$$g_2(x) := \begin{cases} g_1(x) & \text{if } g_1(x) \neq 0 \ \wedge \ (\mu_P(x) < s_\ell \ \vee \ g_1(x) \neq s_\ell) \\ g_3(x) & \text{if } g_1(x) = 0 \ \vee \ (\mu_P(x) \geq s_\ell \ \wedge \ g_1(x) = s_\ell) \end{cases} \tag{5.2}$$

**Claim 5.6.** *For the functions $\mu_P$ and $g_1$, $g_2$, $g_3$ with approximation range $[s_{\min}, s_{\max}]$ and error parameter $\varepsilon$ as defined above,*

1. $L^1(\mu_P, g_3) \leq \varepsilon/5$,
2. $L^1(g_3, g_2) \leq \varepsilon/2$, *and*
3. $\int_{{g_1}^{-1}(0)} |\mu_P(x) - g_1(x)| \, \mathrm{d}x \leq \int_{\mathcal{X} \setminus \mathrm{Lev}(\mu_P, s_{\min})} \mu_P(x) \, \mathrm{d}x$.

The proof of this claim will be given at the end of this section. By 1. and 2. of Claim 5.6 we conclude

$$\begin{aligned} & \int_{\mathcal{X} \setminus {g_1}^{-1}(0) \setminus ({g_1}^{-1}(s_\ell) \cap \mathrm{Lev}(\mu_P, s_\ell))} |\mu_P(x) - g_1(x)| \, \mathrm{d}x \\ \overset{(5.2)}{=} & \int_{\mathcal{X} \setminus {g_1}^{-1}(0) \setminus ({g_1}^{-1}(s_\ell) \cap \mathrm{Lev}(\mu_P, s_\ell))} |\mu_P(x) - g_2(x)| \, \mathrm{d}x \\ \leq & \int_{\mathcal{X}} |\mu_P(x) - g_2(x)| \, \mathrm{d}x \ \leq \ \varepsilon/5 + \varepsilon/2 \end{aligned}$$

and, by 3. of the same claim,

$$\begin{aligned}
&\int_{{g_1}^{-1}(0)\cup({g_1}^{-1}(s_\ell)\cap \mathrm{Lev}(\mu_P,s_\ell))} |\mu_P(x) - g_1(x)| \,\mathrm{d}x \\
\leq &\int_{\mathcal{X}\setminus \mathrm{Lev}(\mu_P,s_{\min})} \mu_P(x)\,\mathrm{d}x + \int_{\mathrm{Lev}(\mu_P,s_{\max})} |\mu_P(x) - s_{\max}|\,\mathrm{d}x \\
= &\text{ r-error}\,(\mu_P, s_{\min}, s_{\max}) \ .
\end{aligned}$$

By combining both inequations,

$$\begin{aligned}
L^1(\mu_P, g_1) = &\int_{\mathcal{X}\setminus {g_1}^{-1}(0)\setminus({g_1}^{-1}(s_\ell)\cap \mathrm{Lev}(\mu_P,s_\ell))} |\mu_P(x) - g_1(x)| \,\mathrm{d}x \\
&+ \int_{{g_1}^{-1}(0)\cup({g_1}^{-1}(s_\ell)\cap \mathrm{Lev}(\mu_P,s_\ell))} |\mu_P(x) - g_1(x)| \,\mathrm{d}x \\
&\leq \varepsilon/5 + \varepsilon/2 + \text{r-error}\,(\mu_P, s_{\min}, s_{\max}) \ .
\end{aligned} \tag{5.3}$$

Note that for every $x \in {g_1}^{-1}(0)\cup({g_1}^{-1}(s_\ell)\cap \mathrm{Lev}\,(\mu_P, s_\ell))$ it holds $\mu_P(x) \geq g_1(x)$. Hence,

$$\begin{aligned}
\int_{\mathcal{X}} g_1(x)\,\mathrm{d}x = &\int_{\mathcal{X}} \mu_P(x)\,\mathrm{d}x \\
&- \int_{\mathcal{X}\setminus {g_1}^{-1}(0)\setminus({g_1}^{-1}(s_\ell)\cap \mathrm{Lev}(\mu_P,s_\ell))} (\mu_P(x) - g_1(x))\,\mathrm{d}x \\
&- \int_{{g_1}^{-1}(0)\cup({g_1}^{-1}(s_\ell)\cap \mathrm{Lev}(\mu_P,s_\ell))} (\mu_P(x) - g_1(x))\,\mathrm{d}x \\
\leq &\int_{\mathcal{X}} \mu_P(x)\,\mathrm{d}x \\
&+ \int_{\mathcal{X}\setminus {g_1}^{-1}(0)\setminus({g_1}^{-1}(s_\ell)\cap \mathrm{Lev}(\mu_P,s_\ell))} |\mu_P(x) - g_1(x)|\,\mathrm{d}x \\
&- \int_{{g_1}^{-1}(0)\cup({g_1}^{-1}(s_\ell)\cap \mathrm{Lev}(\mu_P,s_\ell))} |\mu_P(x) - g_1(x)|\,\mathrm{d}x \\
&\leq 1 + \varepsilon/5 + \varepsilon/2 - (L^1(\mu_P, g_1) - \varepsilon/5 - \varepsilon/2) \\
&= 1 + 1.4\cdot\varepsilon - L^1(\mu_P, g_1) \ .
\end{aligned} \tag{5.4}$$

**Claim 5.7.** *For the functions $g$ and $g_1$ and error parameter $\varepsilon$ as defined above,*

$$L^1(g, g_1) < \varepsilon/4 \ .$$

The proof of this claim will be given at the end of this section as well. By Claim 5.7 and (5.3) we conclude

$$L^1(\mu_P, g) \leq L^1(\mu_P, g_1) + L^1(g, g_1) < \varepsilon + \text{r-error}\,(\mu_P, s_{\min}, s_{\max}) \ .$$

By Claim 5.7 and (5.4) we conclude

$$\int_{\mathcal{X}} g(x)\,\mathrm{d}x \leq \int_{\mathcal{X}} g_1(x)\,\mathrm{d}x + L^1(g, g_1)$$

$$\begin{aligned} &\leq 1 + 1.4\varepsilon - (L^1(\mu_P, g) - L^1(g, g_1)) + L^1(g, g_1) \\ &< 1 + 2\varepsilon - L^1(\mu_P, g) \end{aligned}$$

and thus

$$L^1(\mu_P, g) \leq 2\varepsilon + 1 - \int_{\mathcal{X}} g(x)\,\mathrm{d}x \ .$$

Now consider the running time of `ApproxRange`. We show a bound for the number $\ell$ of iterations of the while loop in Line 5. Recall $s_{\ell-1} < s_{\max} \leq s_\ell$. By

$$s_{\ell-1} = s_{\min}/2 \cdot (1 + \varepsilon/5)^{\ell-2}$$

we have

$$\ell = \ln(2s_{\ell-1} \,/\, s_{\min}) / \ln(1 + \varepsilon/5) + 2 < \ln(2s_{\max} \,/\, s_{\min}) \cdot 6\,\varepsilon^{-1} + 2 \ .$$

Consequently, the algorithm runs in time $\mathcal{O}(\varepsilon^{-1} \ \log(s_{\max} \,/\, s_{\min})\ T)$, where $T$ is the worst case running time of the density level learner. The parameters used to call the density level learner are bounded by

$$\varepsilon_{s_i} \cdot \rho_{s_i} \geq 40^{-1}\ \varepsilon^2\ (\ln(2s_{\max}\, s_{\min}^{-1}) \cdot 6\,\varepsilon^{-1} + 2)^{-2} \in \Omega(\varepsilon^4 \ \log^{-2}(s_{\max} \,/\, s_{\min}))$$

and

$$\delta_i \geq 4^{-1}\ \delta\ (\ln(2s_{\max}\, s_{\min}^{-1}) \cdot 6\,\varepsilon^{-1} + 2)^{-2} \in \Omega(\varepsilon^2\ \delta\ \log^{-2}(s_{\max} \,/\, s_{\min})) \ .$$

This completes the proof of Lemma 5.5. □

The final approximation algorithm `ApproxDist` calls `ApproxRange` incrementally with a growing region of $s$-levels $[s_{\min}, s_{\max}]$. It is presented as Algorithm 6.

**Input:** error parameter $\varepsilon$, confidence parameter $\delta$,
density level learner for $f$ $\texttt{Learn}(\cdot,\cdot,\cdot,\cdot)$
**Output:** approximation $g$ of function $\mu_P$

```
1  i ← 1;
2  s_min ← 0.5;
3  s_max ← 2;
4  while true do
5  |  g ← ApproxRange(ε / 4, 2^{-i} δ, s_min, s_max, Learn);
6  |  if |1 − ∫_X g(x) dx| ≤ ε/2 then
7  |  |  return g;
8  |  s_min ← s_min/2;
9  |  s_max ← 2 s_max;
10 |  i ← i + 1;
```

**Algorithm 6:** `ApproxDist`$(\varepsilon, \delta, \texttt{Learn})$

**Definition 5.8.** *For a density function $\mu_P$ and an error parameter $\varepsilon$, the* $\varepsilon$-variation of $\mu_P$

$$\beta(\mu_P, \varepsilon) := \min\{s \in [1, \infty) : \text{r-error}\left(\mu_P, s^{-1}, s\right) \leq \varepsilon\}$$

*is the minimum value for $s_{\max}$ and ${s_{\min}}^{-1}$ such that the range error made by* `ApproxRange` *is at most $\varepsilon$.*

**Lemma 5.9.** *Consider a probability density function $\mu_P$ such that every level $s \in \mathbb{R}_+$ of $\mu_P$ is density level learnable and, for every $\varepsilon > 0$, the $\varepsilon$-variation of $\mu_P$ is finite. For every $\varepsilon > 0$ and $\delta > 0$, with probability at least $1 - \delta$,* `ApproxDist` *computes an approximation $g$ that satisfies $L^1(\mu_P, g) \leq \varepsilon$. If the density level learner for $\mu_P$ is efficient, the expected running time of* `ApproxDist` *is bounded by a polynomial in $\varepsilon^{-1}$, $\delta^{-1}$, and $\log(\beta(\mu_P, \varepsilon/4))$.*

*Proof.* Consider the termination condition in Line 6 of `ApproxDist` at an arbitrary step of the while loop. Let $i$, $s_{\min}$, $s_{\max}$ denote the values of the corresponding variables. Assume that `ApproxRange` has just returned a hypothesis $g$ that satisfies $L^1(\mu_P, g) \leq \min\{\varepsilon/4 + \text{r-error}(\mu_P, s_{\min}, s_{\max}), \varepsilon/2 + 1 - \int_{\mathcal{X}} g(x)\,\mathrm{d}x\}$. Due to Lemma 5.5 this happens with probability at least $1 - 2^{-i}\,\delta$.

If $L^1(\mu_P, g) < \varepsilon/2$, which is especially true if $s_{\max} = s_{\min}^{-1} \geq \beta(\mu_P, \varepsilon/4)$, the termination condition is satisfied, because

$$\left|1 - \int_{\mathcal{X}} g(x)\,\mathrm{d}x\right| = \left|\int_X (\mu_P(x) - g(x)\,\mathrm{d}x\right| \leq L^1(\mu_P, g)\ .$$

If, on the other hand, $L^1(\mu_P, g) > \epsilon$, we know

$$\int_{\mathcal{X}} g(x)\,\mathrm{d}x \leq \varepsilon/2 + 1 - L^1(\mu_P, g) < 1 - \varepsilon/2$$

and consequently $|1 - \int_{\mathcal{X}} g(x)\,\mathrm{d}x| > \varepsilon/2$.

The probability that `ApproxRange` returns a function $g$ that does not satisfy $L^1(\mu_P, g) \leq \min\{\varepsilon/4 + \text{r-error}(\mu_P, s_{\min}, s_{\max}), \varepsilon/2 + 1 - \int_{\mathcal{X}} g(x)\,\mathrm{d}x\}$ at least once is bounded by $\sum_{i=1}^{\infty} 2^{-i}\,\delta = \delta$.

Assume that the density level learner is efficient. Let $\ell = \lceil \log(\beta(\mu_P, \varepsilon/4)) \rceil$ denote the number of steps of the while loop until $s_{\max} = s_{\min}^{-1} \geq \beta(\mu_P, \varepsilon/4)$ holds. Beginning from the $\ell$-th iteration of the while loop, the algorithm will terminate with probability $1 - 2^{-i}\,\delta$ every time the while loop is executed, where $i$ is the iteration count. Thus, for $k \in \mathbb{N}$, the probability $p_{k+\ell}$ that the $(k+\ell)$-th iteration is ever executed, is bounded by

$$p_{k+\ell} \leq 2^{-k\,\ell}\, 2^{-(k^2+k)/2}\ .$$

If this iteration is executed, the parameters used for the call of `ApproxRange` are $(\varepsilon/4, 2^{-k-\ell}\,\delta, 2^{-k-\ell}, 2^{k+\ell})$. By Lemma 5.5 and efficient density level learning,

this call runs in time $T_{k+\ell}$ bounded by a polynomial in $1/\varepsilon$, $1/\delta$, and $k+\ell$. This yields an expected running time $T'$ for `ApproxDist` of

$$\begin{aligned} T' &\leq \ell \cdot T_\ell + \sum_{k=1}^{\infty} \left(2^{k/2}\right)^{-k} \cdot \operatorname{poly}(1/\varepsilon, 1/\delta, k+\ell) \\ &\leq \ell \cdot T_\ell + \operatorname{poly}(1/\varepsilon, 1/\delta) \cdot \sum_{k=1}^{\infty} \left(2^{k/2} / \operatorname{poly}(1/\varepsilon, 1/\delta, k+\ell)\right)^{-k} \\ &\leq \ell \cdot T_\ell + \operatorname{poly}(1/\varepsilon, 1/\delta) \\ &= \operatorname{poly}(1/\varepsilon, 1/\delta, \log(\beta(\mu_P, \varepsilon/4))) \ . \end{aligned}$$

□

**Theorem 5.10.** *If a probability density function $\mu_P$ has a finite $\varepsilon$-variation and every level $s \in \mathbb{R}_+$ of $\mu_P$ is density level learnable, an approximation $g$ of $\mu_P$ that satisfies $L^1(\mu_P, g) \leq \varepsilon$ can be computed.*

*If in addition the $\varepsilon$-variation $\beta(\mu_P, \varepsilon)$ is bounded by an exponential function in $\varepsilon^{-1}$, the approximation can be computed with sample complexity of the learner bounded by a polynomial in $\varepsilon^{-1}$ and $\delta^{-1}$.*

*Furthermore if the density level learner is efficient for every $s$-level of $\mu_P$, the computation of the approximation $g$ is efficient as well, i. e. the time complexity is bounded by a polynomial in $\varepsilon^{-1}$ and $\delta^{-1}$.*

Recall that, according to Claim 1 of Ben-David and Lindenbaum [11], the other direction also is true.

As an example, we estimate $\beta(\mu_P, \varepsilon)$ for a Gaussian distribution $P = \mathcal{N}(0, \sigma^2)$. Since $\mu_P(x) \leq (2\pi)^{-1/2} \sigma^{-1}$ for every $x \in \mathbb{R}_+$,

$$\beta(\mu_P, \varepsilon) = \max\left\{\mu_P\left(\sigma \cdot \Phi^{-1}(1-\varepsilon/2)\right)^{-1}, (2\pi)^{-1/2}\sigma^{-1}\right\} \ ,$$

where $\Phi$ is the cumulative distribution function of $\mathcal{N}(0,1)$. By

$$\Phi^{-1}(x) = \sqrt{2}\ \operatorname{erf}^{-1}(2x-1) \ ,$$

where $\operatorname{erf}(x) := 2/\sqrt{\pi} \int_0^x \exp(-t^2)\ \mathrm{d}t$ denotes the Gauss error function,

$$\beta(\mu_P, \varepsilon) = \max\left\{\left(1/\sqrt{2\pi\sigma^2}\exp\left(-\operatorname{erf}^{-2}(1-\varepsilon)\right)\right)^{-1}, (2\pi)^{-1/2}\sigma^{-1}\right\} \ .$$

Let $\operatorname{erfc}^{-1}(1-x) := \operatorname{erf}^{-1}(x)$ denote the inverse complementary error function. By Theorem 2 of Chang et al. [19], $\operatorname{erfc}(x) \geq \sqrt{e/(2\pi)}\exp(-2x^2)$ for every $x \geq 0$. Thus

$$\operatorname{erfc}^{-1}(x) \leq \sqrt{-\ln\left(x\sqrt{2\pi/e}\right)/2}$$

and finally

$$\beta\left(\mu_P, \varepsilon\right) \leq \max\left\{e^{1/2}\,\sigma\,\varepsilon^{-1}, (2\pi)^{-1/2}\,\sigma^{-1}\right\} \ .$$

We finish this section with the remaining proofs of Claim 5.6 and 5.7.

*Proof of Claim 5.6.* We prove the three statements of the claim separately. We start with statement 1. By (5.1), for all $k \in \mathbb{Z}$ and all $x \in {g_3}^{-1}(s_k)$

$$g_3(x) = s_k \leq \mu_P(x) < s_{k+1} = s_k + s_k\,\varepsilon/5$$

and thus

$$0 \leq \mu_P(x) - g_3(x) < s_k + s_k\,\varepsilon/5 - g_3(x) = g_3(x)\,\varepsilon/5 \leq \mu_P(x)\,\varepsilon/5 \ .$$

Hence

$$\begin{aligned} &\int_{\mathrm{Lev}(\mu_P,s_k)\setminus\mathrm{Lev}(\mu_P,s_{k+1})} |\mu_P(x) - g_3(x)|\,\mathrm{d}x \\ \leq\ &\int_{\mathrm{Lev}(\mu_P,s_k)\setminus\mathrm{Lev}(\mu_P,s_{k+1})} \mu_P(x)\,\varepsilon/5\,\mathrm{d}x \end{aligned}$$

for every $k \in \mathbb{Z}$.

Since $\mu_P$ is a probability density function, we know that $\int_{\mathcal{X}} \mu_P(x)\,\mathrm{d}x = 1$ and consequently

$$\begin{aligned} &\int_{\mathcal{X}} |\mu_P(x) - g_3(x)|\,\mathrm{d}x \\ =\ &\sum_{k=-\infty}^{\infty} \int_{\mathrm{Lev}(\mu_P,s_k)\setminus\mathrm{Lev}(\mu_P,s_{k+1})} |\mu_P(x) - g_3(x)|\,\mathrm{d}x \\ \leq\ &\varepsilon/5 \cdot \sum_{k=-\infty}^{\infty} \int_{\mathrm{Lev}(\mu_P,s_k)\setminus\mathrm{Lev}(\mu_P,s_{k+1})} \mu_P(x)\,\mathrm{d}x \\ =\ &\varepsilon/5 \cdot \int_{\mathcal{X}} \mu_P(x)\,\mathrm{d}x \\ \leq\ &\varepsilon/5 \ . \end{aligned}$$

Now we prove statement 2., starting with an additional claim.

**Claim 5.11.**

1. *For every $x \in \mathcal{X}$, $g_2(x) = 0 \Leftrightarrow \mu_P(x) = 0$.*
2. *For every $x \in \mathcal{X} \setminus {\mu_P}^{-1}(0)$,*

$$\begin{aligned} &\max\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) \geq s_k + \rho_{s_k}\} \\ \leq\ &g_2(x) < \min\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) < s_k - \rho_{s_k}\} \ . \end{aligned}$$

*Proof.* For every $x \in {\mu_P}^{-1}(0)$, $g_1(x) = 0$ because $s_1\,(1-\varepsilon/5) > 0$. This implies $g_2(x) = g_3(x) = 0$ for every $x \in {\mu_P}^{-1}(0)$. On the other hand, for $x \in {\mu_P}^{-1}(0)$ there are two cases. For $g_1(x) = 0$, $g_2(x) = g_3(x) \neq 0$. For $g_1(x) \neq 0$, $g_2(x)$ equals either $g_3(x) \neq 0$ or $g_1(x) \neq 0$. This finishes the proof of the first statement.

Now fix some $x \in \mathcal{X} \setminus {\mu_P}^{-1}(0)$. If $g_1(x) = 0$ or $\mu_P(x) \geq s_\ell \wedge g_1(x) = s_\ell$, then $g_2(x) = g_3(x) = \max\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) \geq s_k\}$. Thus, $g_2(x)$ satisfies the inequation

$$\max\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) \geq s_k + \rho_{s_k}\} \leq g_2(x) < \min\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) < s_k - \rho_{s_k}\}\ .$$

Assume $x \in \mathcal{X} \setminus {g_1}^{-1}(0) \setminus (\mathrm{Lev}\,(\mu_P, s_\ell) \cap {g_1}^{-1}(s_\ell))$. By definition of $g_2$ in (5.2), $g_2(x) = g_1(x)$ for such $x$.

- Assume $g_1(x) \geq \min\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) < s_k - \rho_{s_k}\}$. This implies that for some $k \in [\ell]$ with $s_k - \rho_{s_k} > \mu_P(x)$ and the corresponding density level $s_k$, the density level learner must have returned a hypothesis $h[s_k]$ that contains $x$ and is not $(0, \rho_{s_k})$-close for that reason. This contradicts the definition of $g_1$. Hence, $g_2(x) = g_1(x) < \min\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) < s_k - \rho_{s_k}\}$.
- Assume $g_1(x) < \max\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) \geq s_k + \rho_{s_k}\}$. Recall $g_1(x) > 0$ and if $g_1(x) = s_\ell$ then $\mu_P(x) < s_\ell$. Thus, at least for the maximum value of $k \in \{2, \ldots, \ell\}$ with $s_k + \rho_{s_k} \leq \mu_P(x)$ and the corresponding density level $s_k$, the density level learner must have returned a hypothesis $h[s_k]$ that does not contain $x$ and is not $(0, \rho_{s_k})$-close for that reason. This contradicts the definition of $g_1$. Hence, $g_2(x) = g_1(x) \leq \max\{s_k : k \in \mathbb{Z} \wedge \mu_P(x) \geq s_k + \rho_{s_k}\}$.

□

Fix any $k \in \mathbb{Z}$. By Claim 5.11 for every $x \in \mathcal{X}$ with $\mu_P(x) \geq s_k = s_{k-1} + \rho_{s_{k-1}}$ we have $g_2(x) \geq s_{k-1}$.

By

$$\begin{aligned} s_k &= s_{k+2} \cdot (1+\varepsilon/5)^{-2} \\ &= s_{k+2} \cdot \Big((1+\varepsilon/5)^2(1-\varepsilon/5)\Big)^{-1}\,(1-\varepsilon/5) \\ &= s_{k+2} \cdot \Big(1+\varepsilon/5-\varepsilon^2/25-\varepsilon^3/125\Big)^{-1}\,(1-\varepsilon/5) \\ &< s_{k+2} \cdot (1-\varepsilon/5) \\ &= s_{k+2} - \rho_{s_{k+2}} \end{aligned}$$

and Claim 5.11 for every $x \in \mathcal{X}$ with $\mu_P(x) \leq s_k < s_{k+2} - \rho_{s_{k+2}}$ we have $g_2(x) < s_{k+2}$. Consequently, for every $x \in \mathcal{X}$ with $\mu_P(x) \leq s_k$, it holds $g_2(x) \leq s_{k+1}$.

We conclude that for all $k \in \mathbb{Z}$ and for all $x \in {g_3}^{-1}(s_k)$ it holds $s_{k-1} \leq g_2(x) \leq s_{k+2}$, because $g_3(x) = s_k \leq \mu_P(x) < s_{k+1}$ by (5.1).

Since for every $k \in \mathbb{Z}$

$$s_{k+2} - s_k = s_k \cdot (1 + \varepsilon/5)^2 - s_k = s_k \cdot 2\varepsilon/5 + s_k \cdot \varepsilon^2/25 < s_k \cdot \varepsilon/2$$

and

$$s_k - s_{k-1} = s_{k-1} + \rho_{s_{k-1}} - s_{k-1} = \rho_{s_{k-1}} < \rho_{s_k} = s_k \cdot \varepsilon/5 < s_k \cdot \varepsilon/2$$

we conclude

$$|g_3(x) - g_2(x)| \leq g_3(x) \cdot \varepsilon/2$$

for every $x \in \mathcal{X} \setminus {\mu_P}^{-1}(0)$. Also note that ${\mu_P}^{-1}(0) = {g_3}^{-1}(0) = {g_2}^{-1}(0)$.

We conclude

$$L^1(g_3, g_2) = \int_{\mathcal{X}} |g_3(x) - g_2(x)| \, \mathrm{d}x \leq \varepsilon/2 + \int_{\mathcal{X}} g_3(x) \, \mathrm{d}x \leq \varepsilon/2$$

since $g_3(x) \leq \mu_P(x)$ for every $x \in \mathcal{X}$ and $\int_{\mathcal{X}} \mu_P(x) \, \mathrm{d}x = 1$.

We finish the proof of Claim 5.6 by proving statement Claim 3 now. Obviously, $\int_{{g_1}^{-1}(0)} |\mu_P(x) - g_1(x)| \, \mathrm{d}x = \int_{{g_1}^{-1}(0)} \mu_P(x) \, \mathrm{d}x$. Fix some arbitrary $x \in {g_1}^{-1}(0)$. By definition of $g_1$, we conclude $\mu_P(x) \leq s_{\min}/2 + \rho_{s_{\min}/2} = (1 + \varepsilon/5)s_{\min}/2 < s_{\min}$. Thus, $x \in \mathcal{X} \setminus \mathrm{Lev}(\mu_P, s_{\min})$ and finally ${g_1}^{-1}(0) \subseteq \mathcal{X} \setminus \mathrm{Lev}(\mu_P, s_{\min})$. □

*Proof of Claim 5.7.* Consider $x \in \mathcal{X}$ with $g(x) \neq g_1(x)$. If $g_1(x) < g(x)$ then there exists some $k \in [\ell]$ such that $g(x) = s_k$ and $g_1(x) < s_k$. In the other case, namely $g_1(x) > g(x)$, there exists some $k \in [\ell]$ such that $g_1(x) = s_k$ and $g(x) < s_k$. We conclude

$$\begin{aligned}
\int_{\mathcal{X}} |g(x) - g_1(x)| \, \mathrm{d}x &= \int_{\{x \in \mathcal{X} : g(x) \neq g_1(x)\}} |g(x) - g_1(x)| \, \mathrm{d}x \\
&= \sum_{k=1}^{\ell} \left( \int_{g^{-1}(s_k) \setminus \mathrm{Lev}(g_1, s_k)} |g(x) - g_1(x)| \, \mathrm{d}x \right. \\
&\quad \left. + \int_{{g_1}^{-1}(s_k) \setminus \mathrm{Lev}(g, s_k)} |g(x) - g_1(x)| \, \mathrm{d}x \right) .
\end{aligned}$$

Now consider an arbitrary $k \in [\ell]$. We show bounds for both summands separately.

First summand:

$$\begin{aligned}\int_{g^{-1}(s_k)\setminus \mathrm{Lev}(g_1,s_k)} |g(x)-g_1(x)|\ \mathrm{d}x &\leq \int_{g^{-1}(s_k)\setminus \mathrm{Lev}(g_1,s_k)} g(x)\ \mathrm{d}x \\ &= \int_{g^{-1}(s_k)\setminus \mathrm{Lev}(g_1,s_k)} s_k\ \mathrm{d}x \\ &= s_k \cdot \lambda\left(g^{-1}(s_k) \setminus \mathrm{Lev}\left(g_1, s_k\right)\right) \\ &\leq s_k \cdot \lambda(h[s_k] \setminus \mathrm{Lev}\left(\mu_P, s_k - \rho_{s_k}\right))\ ,\end{aligned}$$

where $h[s_k]$ is the hypothesis for $s_k$ returned by the density level learner during the construction of $g$.

Second summand:

$$\begin{aligned}\int_{{g_1}^{-1}(s_k)\setminus \mathrm{Lev}(g,s_k)} |g(x)-g_1(x)|\ \mathrm{d}x &\leq \int_{{g_1}^{-1}(s_k)\setminus \mathrm{Lev}(g,s_k)} g_1(x)\ \mathrm{d}x \\ &= \int_{{g_1}^{-1}(s_k)\setminus \mathrm{Lev}(g,s_k)} s_k\ \mathrm{d}x \\ &= s_k \cdot \lambda\left({g_1}^{-1}(s_k) \setminus \mathrm{Lev}\left(g, s_k\right)\right) \\ &\leq s_k \cdot \lambda(\mathrm{Lev}^{>}\left(\mu_P, s_k + \rho_{s_k}\right) \setminus h[s_k])\ .\end{aligned}$$

Since the density level learner is called with error parameter $\varepsilon_{s_k}$, it holds that $\lambda(h[s_k] \setminus \mathrm{Lev}\left(\mu_P, s_k - \rho_{s_k}\right)) + \lambda(\mathrm{Lev}^{>}\left(\mu_P, s_k + \rho_{s_k}\right) \setminus h[s_k]) \leq \varepsilon_{s_k}$. We conclude

$$\begin{aligned}\int_{\mathcal{X}} |g(x)-g_1(x)|\ \mathrm{d}x &\leq \sum_{k=1}^{\ell} s_k \cdot \varepsilon_{s_k} = \sum_{k=1}^{\ell} \varepsilon/(8k^2) \\ &< \sum_{k=1}^{\infty} \varepsilon/(8k^2) = \varepsilon\,\pi^2/48 < \varepsilon/4\ .\end{aligned}$$

□

## 5.3 Consistent Density Level Learners

Ben-David and Lindenbaum [11] have shown that a concept class $\mathcal{C}$, respectively the distribution family $\mathcal{P}$ that $\mathcal{C}$ is $\rho$-compatible with (for every $\rho > 0$), can be density level learned if the VC dimension of $\mathcal{C}$ is finite. They introduce a sufficient (but not necessary) criterion, called the *consistency criterion*, which guarantees that a density learner is successful. This is analogous to PAC learning, where a learner is successful if it always returns a hypothesis that is consistent to the sample the learner has received. In this section the results of Ben-David and Lindenbaum regarding the consistency criterion for density level learning are presented.

For a sample $X \in \mathcal{X}^m$ with $m \in \mathbb{N}$ and a measurable set $C \in \mathcal{B}$ let $\nu(X, C) := |\{i \in [m] : X_i \in C\}|/m$ denote the proportion of elements of $X$ that belong to $C$.

**Definition 5.12.** *A sample $X \in \mathcal{X}^m$ with $m \in \mathbb{N}$ is an $\varepsilon$-approximation of a distribution $P$ on $\mathcal{X}$ relative to $\mathcal{C} \subseteq \mathcal{B}$ if*

$$\forall\, C \in \mathcal{C} :\ |\nu(X, C) - P(C)| < \varepsilon \ .$$

Thus for a given set $\mathcal{C} \subseteq \mathcal{B}$, a sample $X$ generated independently at random according to $P$ is expected to be an $\varepsilon$-approximation of $P$ relative to $\mathcal{C}$. To shorten the notation let us define

$$N(d, \varepsilon, \delta) \ := \ \left\lceil \frac{16}{\varepsilon^2} \left( d \ln \frac{16\, d}{\varepsilon^2} + \ln \frac{4}{\delta} \right) \right\rceil \ .$$

**Lemma 5.13** (Vapnik and Chervonenkis [62])**.** *Consider a probability distribution $P$ and a set $\mathcal{C} \subseteq \mathcal{B}$ and choose $\varepsilon, \delta > 0$. If $d = \text{VC-dim}(\mathcal{C})$ is finite, then with probability at least $1 - \delta$ a sample $X$ of length at least $N(d, \varepsilon, \delta)$ that is generated independently at random according to $P$ is an $\varepsilon$-approximation of $P$ relative to $\mathcal{C}$.*

For a concept class $\mathcal{C}$ and a hypothesis space $\mathcal{H}$ define the set of all error regions by

$$\mathcal{C}^*_{\mathcal{H}} \ := \ \{c \setminus h : c \in \mathcal{C} \wedge h \in \mathcal{H}\} \ \cup \ \{h \setminus c : c \in \mathcal{C} \wedge h \in \mathcal{H}\} \ .$$

To shorten the notation, we write $\mathcal{C}^*$ for $\mathcal{C}^*_{\mathcal{C}}$.

**Definition 5.14.** *Let $D$ be a measure, $\mathcal{C}^*_{\mathcal{H}} \subseteq \mathcal{B}$ a set of error regions, $s \in \mathbb{R}_+$ a level, and $\eta \in \mathbb{R}_+$ an error bound. A hypothesis $h \in \mathcal{H}$ is $(s, \eta)$-consistent with a sample $X \in \mathcal{X}^m$ relative to $\mathcal{C}^*_{\mathcal{H}}$ and $D$ if for every $e \in \mathcal{C}^*_{\mathcal{H}}$ holds:*

$$\begin{aligned} e \cap h = \emptyset \quad &\Longrightarrow \quad \nu(X, e) < s\, D(e) + \eta \ , \\ e \subseteq h \quad &\Longrightarrow \quad \nu(X, e) > s\, D(e) - \eta \ . \end{aligned}$$

The idea of this definition is as follows. At first note that for every set $e \in \mathcal{B}$ and the expected value $\mathrm{E}[\nu(X, e)]$ it holds

$$\begin{aligned} e \subseteq \mathrm{Lev}(\mu_P/\mu_D, s) \quad &\Longrightarrow \quad \mathrm{E}[\nu(X, e)] \geq s \cdot D(e) \ , \\ e \cap \mathrm{Lev}(\mu_P/\mu_D, s) = \emptyset \quad &\Longrightarrow \quad \mathrm{E}[\nu(X, e)] < s \cdot D(e) \ . \end{aligned}$$

Let $c = \mathrm{Lev}(\mu_P/\mu_D, s) \in \mathcal{C}$ denote the target concept and $h \in \mathcal{H}$ the hypothesis. If the error region $c \setminus h$ is large, we expect that for $e = c \setminus h \in \mathcal{C}^*_{\mathcal{H}}$ and sufficient small $\eta$ the first condition of the consistency criterion is violated. If, on the other hand, the error region $h \setminus c$ is large, analogously we expect the second condition to be violated.

To be more precise, a violation of consistency can only be guaranteed with high probability if a region $e = \mathrm{Lev}(\mu_P/\mu_D, s + \rho) \setminus h$ or $e = h \setminus \mathrm{Lev}(\mu_P/\mu_D, s - \rho)$ is

considered. The reason for that is statistical noise and the relaxation parameter $\eta$, which, on the other hand, is necessary to guarantee closeness of good hypothesis with high probability despite statistical noise. Hence, $\mathcal{C}$ must be chosen in such a way that $\mathcal{C}^*$ contains these regions $e$. This is the case if $\mathcal{C}$ is $\rho$-compatible with the $s$-levels of $\mathcal{P}$ (instead of simple compatibility only). Since $\rho$ is a variable parameter, it seems necessary either to restrict the range of $\rho$ or to require compatibility of $\mathcal{C}$ with every $s$-level of $\mathcal{P}$.

Finally, to restrict the range of $\nu(X, e)$, $X$ needs to be an $\varepsilon$-approximation of $P$ according to $\mathcal{C}^*$. In order to estimate the probability of this event, an estimation of the VC dimension of $\mathcal{C}^*$ is necessary.

**Claim 5.15** ([11]). *Let $\mathcal{C} \subseteq \mathcal{B}$ denote a concept class with finite VC dimension. Then* $\text{VC-dim}(\mathcal{C}^*) \leq 2\,\text{VC-dim}(\mathcal{C}) \cdot \log(\text{VC-dim}(\mathcal{C}))$.

With the tools of $\varepsilon$-approximation and $(s, \eta)$-consistency, Ben-David and Lindenbaum were able to conclude the following result.

**Lemma 5.16** ([11]). *Let $\mathcal{C}$ be a concept class, $d = \text{VC-dim}(\mathcal{C})$ and $D$ a measure. Let $\mathcal{A}$ be a learning algorithm that draws a sample $X \in \mathcal{X}^m$ of length $m \in \mathbb{N}$ according to a probability distribution $P$ and returns, whenever possible, a hypothesis $h \in \mathcal{H} = \mathcal{C}$ that is $(s, \varepsilon\rho/4)$-consistent with $X$ relative to $\mathcal{C}^*$, where $s$, $\varepsilon$, $\delta$, and $\rho$ are parameters given to $\mathcal{A}$.*

*For every $\varepsilon, \delta, \rho > 0$, every $s \in \mathbb{R}_+$, and every probability distribution $P \in \mathcal{P} \subseteq \mathcal{D}_D^{\pm\rho}(\mathcal{C}, s)$, $\mathcal{A}$ is $(m, \varepsilon, \rho, \delta)$-successful for $\mathcal{C}$ and $s$ if $m \geq N(2d \log d, \varepsilon\rho/4, \delta)$.*

Ben-David and Lindenbaum have restricted the measure $D$ to probability measures. However, it is easy to see that their results hold for arbitrary measures $D$ as well, as long as $D$ is a measure on $(\mathcal{X}, \mathcal{B})$. They also have shown that the finiteness of the VC dimension is a necessary condition for density level learning.

**Lemma 5.17** ([11]). *Let $\mathcal{C}$ be a concept class with infinite VC dimension and $D$ a measure.*

*There exists no learning algorithm $\mathcal{A}$ and function $m : (0, 1]^3 \to \mathbb{N}$ such that $\mathcal{A}$ is $(m(\varepsilon, \rho, \delta), \varepsilon, \rho, \delta)$-successful for every $\varepsilon, \delta, \rho > 0$, every $s \in \mathbb{R}_+$, and every probability distribution $P \in \mathcal{P} \subseteq \mathcal{D}_D^{\pm\rho}(\mathcal{C}, s)$.*

## 5.4 Empirical Excess Mass Maximization

Empirical excess mass maximization has been introduced by Hartigan [36] and Müller and Sawitzki [48]. Algorithms for density level estimation by excess mass maximization have been designed such as for the concept class of $k$-fold unions of intervals in $\mathbb{R}$ by Müller and Sawitzki [48], the class of convex polygons in $\mathbb{R}^2$ by Hartigan [36], and the class of closed ellipsoids in $\mathbb{R}^d$ by Nolan [50]. However, the quality of estimation has been studied by terms of the convergence rates, but not in a computational learning theory setting, i. e. with respect to the

sample and time complexity bounds depending on the estimation error. As a consequence, for each concept class the convergence rate of the estimator is proven separately. This is not necessary if excess mass maximization gets embedded into a framework for density level learning. In this section we will show that such algorithms can indeed be used in the BL framework. Note that this does not apply to convex polygons considered by Hartigan, which provably cannot be learned in the BL framework, because that concept class has an infinite VC dimension.

**Definition 5.18.** *Given a probability distribution $P$, a measure $D$, and a threshold value $s \in \mathbb{R}_+$, the excess mass function for a hypothesis $h \in \mathcal{H}$ is defined by*

$$E_{P,D}(h,s) := P(h) - s\,\mu_D(h) \ .$$

Obviously, for $h = \mathrm{Lev}(\mu_P/\mu_D, s)$ we get $E_{P,D}(h,s) \geq 0$. Furthermore, if $\mu_P/\mu_D(x) = s$ for all $x \in h$ then $E_{P,s}(h) = 0$. Thus, $E_{P,D}(h,s)$ measures how much $\mu_P/\mu_D$ exceeds the $s$-level within the hypothesis $h$. Replacing $P(h)$ by the relative frequency leads to the definition of the *empirical* excess mass.

**Definition 5.19.** *Given a finite sample set $X$, a measure $D$, and a threshold value $s \in \mathbb{R}_+$, the empirical excess mass function for a hypothesis $h \in \mathcal{H}$ is given by*

$$H_{X,D}(h,s) := \nu(X,h) - s\,\mu(h) \ .$$

*For $\mathcal{C} \subseteq \mathcal{B}$, a hypothesis $h \in \mathcal{H}$ is $(H_{X,D}, s)$-maximal relative to $\mathcal{C}$ if*

$$\forall\, C \in \mathcal{C} : \ H_{X,D}(h,s) \geq H_{X,D}(C,s) \ .$$

Empirical excess mass maximization means to find a $(H_{X,D}, s)$-maximal hypothesis. Since $D$ is countably additive, $E_{P,D}(h,s)$ becomes maximal for $h$ if $\mathrm{Lev}^{>}(\mu_P/\mu_D, s) \subseteq h \subseteq \mathrm{Lev}(\mu_P/\mu_D, s)$.

In the following we will show that for every $\mathcal{C} \subseteq \mathcal{B}$ with finite VC dimension and $\mathcal{H} = \mathcal{C}$, every empirical excess mass maximization algorithm is a successful density level learner according to Definition 5.4. Note that on the other hand, the negative result for concept classes with infinite VC dimension stated in Lemma 5.17 is independent of the learning algorithm and holds for empirical excess mass maximization algorithms obviously as well.

First, it will be shown that every hypothesis $h$ generated by empirical excess mass maximization is $(\varepsilon, \rho)$-close to $\mathrm{Lev}(\mu_P/\mu_D, s)$ if a suitable sample set $X$ has been used.

**Lemma 5.20.** *Let $\mathcal{C} \subseteq \mathcal{B}$ denote a concept class, $D$ a measure, $s \in \mathbb{R}_+$ a threshold value, and $\varepsilon, \rho > 0$ error parameters. Consider a probability distribution $P \in \mathcal{P}$ where $\mathcal{C}$ is compatible with the $s$-levels of $\mathcal{P}$ with respect to $D$. Further let $X \in \mathcal{X}^m$ be an $(\varepsilon\rho/2)$-approximation of $P$ relative to $\mathcal{C}^*$ and choose a hypothesis $h \in \mathcal{C}$ that is $(H_{X,D}, s)$-maximal relative to $\mathcal{C}$. Then $h$ is $(\varepsilon, \rho)$-close to $\mathrm{Lev}(\mu_P/\mu_D, s)$.*

*Proof.* Let $r = \mu_P/\mu_D$. From $h$ being $(H_{X,D}, s)$-maximal relative to $\mathcal{C}$ follows $H_{X,D}(\mathrm{Lev}(r,s),s) \leq H_{X,D}(h,s)$. Furthermore, from

$$\begin{aligned} H_{X,D}(\mathrm{Lev}(r,s),s) = {} & H_{X,D}(h,s) \\ & + H_{X,D}(\mathrm{Lev}(r,s) \setminus h, s) \\ & - H_{X,D}(h \setminus \mathrm{Lev}(r,s), s) \end{aligned}$$

we conclude

$$H_{X,D}(\mathrm{Lev}(r,s) \setminus h, s) \;\leq\; H_{X,D}(h \setminus \mathrm{Lev}(r,s), s) \;. \tag{5.5}$$

The sample $X$ is an $(\varepsilon\rho/2)$-approximation of $P$ relative to $\mathcal{C}^*$. Therefore

$$\begin{aligned} \nu(X, \mathrm{Lev}(r,s) \setminus h) &> P(\mathrm{Lev}(r,s) \setminus h) - \frac{\varepsilon\rho}{2} \\ &\geq s \cdot D((\mathrm{Lev}(r,s) \setminus \mathrm{Lev}^{>}(r, s+\rho)) \setminus h) \\ &\quad + (s+\rho)\, D(\mathrm{Lev}^{>}(r,s+\rho) \setminus h) \\ &\quad - \frac{\varepsilon\rho}{2} \\ &= s \cdot D(\mathrm{Lev}(r,s) \setminus h) + \rho \cdot D(\mathrm{Lev}^{>}(r,s+\rho) \setminus h) - \frac{\varepsilon\rho}{2} \end{aligned}$$

and thus

$$H_{X,D}(\mathrm{Lev}(r,s) \setminus h, s) \;>\; \rho \cdot D(\mathrm{Lev}^{>}(r,s+\rho) \setminus h) - \frac{\varepsilon\rho}{2} \;. \tag{5.6}$$

Again $X$ being an $(\varepsilon\rho/2)$-approximation of $P$ relative to $\mathcal{C}^*$ implies

$$\begin{aligned} \nu(X, h \setminus \mathrm{Lev}(r,s)) &< P(h \setminus \mathrm{Lev}(r,s)) + \frac{\varepsilon\rho}{2} \\ &\leq (s-\rho)\, D(h \setminus \mathrm{Lev}(r, s-\rho)) \\ &\quad + s \cdot D((h \cap \mathrm{Lev}(r,s-\rho)) \setminus \mathrm{Lev}(r,s)) \\ &\quad + \frac{\varepsilon\rho}{2} \\ &= s \cdot D(h \setminus \mathrm{Lev}(r,s)) - \rho \cdot D(h \setminus \mathrm{Lev}(r,s-\rho)) + \frac{\varepsilon\rho}{2} \end{aligned}$$

and thus

$$H_{X,D}(h \setminus \mathrm{Lev}(r,s), s) \;<\; \frac{\varepsilon\rho}{2} - \rho \cdot D(h \setminus \mathrm{Lev}(r, s-\rho)) \;. \tag{5.7}$$

By putting (5.5), (5.6), and (5.7) together we get

$$\begin{aligned} \frac{\varepsilon\rho}{2} - \rho \cdot D(h \setminus \mathrm{Lev}(r,s-\rho)) &> H_{X,D}(h \setminus \mathrm{Lev}(r,s), s) \\ &\geq H_{X,D}(\mathrm{Lev}(r,s) \setminus h, s) \\ &> \rho \cdot D(\mathrm{Lev}^{>}(r,s+\rho) \setminus h) - \frac{\varepsilon\rho}{2} \end{aligned}$$

and finally $D(h \setminus \mathrm{Lev}(r, s-\rho)) \ + \ D(\mathrm{Lev}^{>}(r, s+\rho) \setminus h) < \varepsilon$. □

With the help of Lemma 5.13 and Claim 5.15, Lemma 5.20 yields the following theorem.

**Theorem 5.21.** *Let $\mathcal{C}$ be a concept class, $d = \text{VC-dim}(\mathcal{C})$ and $D$ a measure. Let $\mathcal{A}$ be a learning algorithm that draws a sample $X \in \mathcal{X}^m$ of length $m \in \mathbb{N}$ according to an unknown probability distribution $P \in \mathcal{P}$ and always returns a hypothesis $h \in \mathcal{H} = \mathcal{C}$ that is $(H_{X,D}, s)$-maximal relative to $\mathcal{C}$, where $s$ is a parameter given to $\mathcal{A}$.*

*For every $\varepsilon, \delta, \rho > 0$, every $s \in \mathbb{R}_+$, and every family of probability distributions $\mathcal{P} \subseteq \mathcal{D}_D(\mathcal{C}, s)$, $\mathcal{A}$ is $(m, \varepsilon, \rho, \delta)$-successful for $\mathcal{C}$ and $s$ if $m \geq N(2d \log d, \varepsilon\rho/2, \delta)$.*

In contrast to Lemma 5.16, a density learner that maximizes the empirical excess mass has only to deal with the concept class $\mathcal{C}$ that is compatible to the actual $s$-level to be learned. There is no need to consider the neighborhood levels at $s - \rho$ and $s + \rho$, neither to bother with $\mathcal{C}^*$ when the learning algorithm is designed. In fact, here the set of all error regions $\mathcal{C}^*$ occurs inside the correctness proof only, yielding a minimum sample size of $N(2d \log d, \varepsilon\rho/2, \delta)$ (instead of $N(d, \varepsilon\rho/2, \delta)$). Also note that the upper bound for the number of samples $m$ needed for successful density level learning has decreased from $m \geq N(2d \log d, \varepsilon\rho/4, \delta)$ in Lemma 5.16 to $m \geq N(2d \log d, \varepsilon\rho/2, \delta)$ here.

Empirical excess mass maximization has already been studied for learning density levels. For specific concept classes algorithms that maximize the empirical excess mass are already known (see the beginning of this section). Theorem 5.21 can be applied to these algorithms directly, yielding density learners for the BL framework instantly.

As an example, consider a generalization of $k$-fold mixtures of Gaussian distributions, consisting of univariate probability distributions whose density functions have at most $k$ local maxima each. All of its density levels are covered by the concept class of $k$-fold unions of intervals. This concept class has VC dimension $2k$. Using the algorithm that maximizes the empirical excess mass for this class [48], the density function of such distributions can be approximated by `ApproxDist`.

For the class of $k$-fold mixtures of Gaussian distributions, Ashtiani et al. [10] have given new bounds on the sample complexity for learning the density function, making use of specific properties of these mixtures. In the univariate case, $\tilde{\Theta}(k\,\varepsilon^{-2})$ examples are sufficient. The sample complexity of our approach also depends on these parameters polynomially. However, our approach also works for the generalized class described above. This class includes $k$-fold mixtures of uniform distributions on intervals, triangular distributions, Gaussian distributions, and many others.

Consider the bivariate case. The density levels of a single Gaussian distribution form ellipses with VC dimension 5. The empirical excess mass of closed ellipses

can be maximized by the algorithm of Nolan [50]. Therefore the density function of bivariate Gaussian distributions can be approximated by our method as well. However, density levels of $k$-fold mixtures of bivariate Gaussians may have a complicated pattern, which cannot be described by the $k$-fold union of ellipses. It is left for future work to analyze the concept class of these density levels and design an algorithm that maximizes the empirical excess mass.

# 6 Conclusions and Discussion

In this thesis, different learning problems been studied. We have solved two open problems, regarding learning DNF formulas and regular languages. Further, we have studied the identification of high probability areas in probability distributions.

Concerning DNF formulas, we have presented an efficient and proper learning algorithm for $k$-term DNF formulas that learns from positive examples alone that are drawn at random according to an unknown $q$-bounded distribution. Thereby we have answered an open question of Pitt and Valiant [51], who have asked whether restrictions of the probability distributions used for sampling yield learnability of $k$-term DNF formulas from positive examples alone. Our result is even stronger since our algorithm learns strongly without false positives. The impossibility to learn poly($d$)-term DNF formulas weakly without false positives from positive examples alone independent of running time constraints and the hypothesis class indicates that our sample complexity bound, which grows exponentially in $k$, cannot be improved substantially.

The question about whether $\log(d)$-term DNF formulas can be learned properly and efficiently from positive examples alone is still open for further research. The positive result for monotone $\log(d)$-term DNF formulas by Sakai and Maruoka [54] indicates that efficient and proper learning could be possible. As well does the negative result, which has been proven for poly($d$)-term DNF formulas only.

Another open problem is the extension of our results to other or more general classes of distributions. Flammini et al. [34] have claimed that their proper learning algorithm for $k$-term DNF formulas from positive and negative samples can be adapted to the class of product distributions as well. Sakai and Maruoka [55] have introduced a generalization of $q$-bounded distributions and product distributions, called smooth distributions. They have shown that monotone $k$-term DNF formulas can be learned properly and efficiently even from examples drawn according to unknown smooth distributions. Further research is needed to figure out whether $k$-term DNF formulas can be learned efficiently and properly from positive examples alone under product or even smooth distributions.

When it comes to regular languages, we have shown that the $\mathtt{AL}^\star$ algorithm designed by Angluin et al. [7] may return non-residual AFAs. This disproves a conjecture postulated by the authors. We have fixed that issue and designed a new algorithm $\mathtt{AL}^{\star\star}$ that always returns residual AFAs. This algorithm fulfills the same asymptotic complexity bounds as $\mathtt{AL}^\star$. Moreover, even the absolute query complexity is almost the same. We have performed experiments with

random regular languages, and for more than 98% of the non-trivial instances, $\texttt{AL}^{\star\star}$ needs only one additional equivalence-query compared to $\texttt{AL}^{\star}$ in order to verify the residuality. In these cases, the number of membership queries keeps the same. Besides the fact that $\texttt{AL}^{\star\star}$ always returns residual AFAs, it has almost the same properties as $\texttt{AL}^{\star}$. It generates an AFA that has always less states than (or the same number of) the equivalent canonical RNFA. Typically, $\texttt{AL}^{\star\star}$ generates automata which are significantly more succinct than DFAs and RNFAs. This observation fits to the theoretical analysis, which shows that (residual) AFAs can be exponentially more succinct than NFAs and even double exponentially more succinct than the minimal DFA.

However, it is still open to find a definition of canonical AFAs that, like canonical NFAs or the canonical UFAs, are not only residual, but also unique. While residuality always adds a natural meaning to the automaton's states, uniqueness of the minimal residual AFA is not guaranteed. We have encountered regular languages for which both, the canonical NFA and the canonical UFA, are minimal residual AFAs as well. Naturally, these two minimal residual AFAs differ. It is not obvious how a natural definition of the canonical AFA should look like.

An ambitious open problem is learning of context-free grammars (CFG), or pushdown automata respectively, from membership and equivalence queries. Learning CFGs has been solved by Angluin [4] for CFGs that have at most $k$ non-terminal symbols at the right-hand side of each production rule, and with even more powerful non-terminal membership queries. To our best knowledge, this problem is still open for general CFGs or if only ordinary membership queries can be asked (see also the survey by Lee [44]).

Finally we have considered the identification of high-density areas in probability distributions via algorithmic learning. We have clarified the relation between learning density levels and the estimation of the entire probability distribution function. Afterwards, we have shown that algorithms that maximize the empirical excess mass can be applied directly to learn density levels in the framework of Ben-David and Lindenbaum – a framework that transforms the spirit of PAC learning to density levels. Before, the design of learning algorithms in Ben-David's and Lindenbaum's framework required a lot of effort. On the other hand, empirical excess mass maximization algorithms are known to be successful learners without further proof now.

Nonetheless, the following open questions about empirical excess mass maximization arise for further research: Which concept classes are meaningful to describe density levels of probability distributions, especially in the multivariate case? What is their VC dimension? For bivariate Gaussian distributions, every density level is a ellipse, yielding a concept class with VC dimension 5. However, the geometry of density levels corresponding to mixtures of multivariate Gaussian distributions becomes complicated, not to mention mixtures of other distributions. If the structure of density levels is clarified, the question about whether the empirical excess mass can be maximized efficiently arises subsequently. And,

concerning existing excess mass maximizers as well, with which measures do these algorithms work? Are they restricted to the Lebesgue measure?

Recently, there have been astounding progress in the area of practical machine learning. For example, consider the artificial intelligence AlphaZero [58], which can beat the best human players of the board games Go, chess, and shogi after a few hours of training only. For training, only the rules were given to AlphaZero, which has learned the strategy by playing against itself repeatedly. Another example of great advances is lung cancer screening, where a newly developed learning algorithm surpasses expert specialists in computed tomography findings [9]. However, these successes originate essentially from practical approaches without a well-founded theoretical basis. For the further targeted development, it seems to be helpful to understand why and how these methods work. Thus, research in learning theory must be promoted, especially to close the gap between theory and practice and to improve the understanding of the most recently practical progress.

# Bibliography


[1] Hasan Abasi, Nader H. Bshouty, and Hanna Mazzawi. On exact learning monotone DNF from membership queries. In Peter Auer, Alexander Clark, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory*, pages 111–124, Cham, 2014. Springer International Publishing.

[2] Yohji Akama and Kei Irie. VC dimension of ellipsoids. *arXiv e-prints*, arXiv:1109.4347, 2011.

[3] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *J. Comput. Syst. Sci.*, 18(2):155–193, 1979.

[4] Dana Angluin. Learning $k$-bounded context-free grammars. Technical Report RR-557, Yale University, 1987.

[5] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[6] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr 1988.

[7] Dana Angluin, Sarah Eisenstat, and Dana Fisman. Learning regular languages via alternating automata. In *Proc. 24. IJCAI*, pages 3308–3314, 2015.

[8] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[9] Diego Ardila, Atilla P. Kiraly, Sujeeth Bharadwaj, Bokyung Choi, Joshua J. Reicher, Lily Peng, Daniel Tse, Mozziyar Etemadi, Wenxing Ye, Greg Corrado, David P. Naidich, and Shravya Shetty. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature Medicine*, 2019.

[10] Hassan Ashtiani, Shai Ben-David, Nicholas Harvey, Christopher Liaw, Abbas Mehrabian, and Yaniv Plan. Nearly tight sample complexity bounds for learning mixtures of gaussians via sample compression schemes. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3412–3421. Curran Associates, Inc., 2018.

[11] Shai Ben-David and Michael Lindenbaum. Learning distributions by their density levels: A paradigm for learning without a teacher. *Journal of Computer and System Sciences*, 55(1):171–182, 1997.

[12] Sebastian Berndt, Maciej Liśkiewicz, Matthias Lutter, and Rüdiger Reischuk. Learning residual alternating automata. *AAAI Conference on Artificial Intelligence*, 2017.

[13] P. Bhattacharyya and G. Nagaraja. Learning a class of regular languages in the probably approximately correct learnability framework of valiant. In *IEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives*, pages 2/1–2/16, 1993.

[14] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik–Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[15] Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of NFA. In *Proc. 21. IJCAI*, pages 1004–1009, 2009.

[16] Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker, Daniel Neider, and David R Piegdon. libalf: The automata learning framework. In *International Conference on Computer Aided Verification*, pages 360–364. Springer, 2010.

[17] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, January 1981.

[18] Ashok K. Chandra and George Markowsky. On the number of prime implicants. *Discrete Mathematics*, 24(1):7–11, 1978.

[19] S. Chang, P. C. Cosman, and L. B. Milstein. Chernoff-type bounds for the gaussian error function. *IEEE Transactions on Communications*, 59(11):2939–2944, 2011.

[20] Yu-Fang Chen, Chiao Hsieh, Ondřej Lengál, Tsung-Ju Lii, Ming-Hsien Tsai, Bow-Yaw Wang, and Farn Wang. PAC learning-based verification and model synthesis. In *Proceedings of the 38th International Conference on Software Engineering*, pages 714–724. ACM, 2016.

[21] Anindya De, Ilias Diakonikolas, and Rocco A Servedio. Learning from satisfying assignments. In *Proc. 26th SODA*, pages 478–497. SIAM, 2014.

[22] Colin De La Higuera. A bibliographical study of grammatical inference. *Pattern recognition*, 38(9):1332–1348, 2005.

[23] Scott E. Decatur. PAC learning with constant-partition classification noise and applications to decision tree induction. In *Proc. ICML*, pages 83–91, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[24] François Denis. PAC learning from positive statistical queries. In Michael M. Richter, Carl H. Smith, Rolf Wiehagen, and Thomas Zeugmann, editors, *Proc. ALT*, pages 112–126, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[25] François Denis. PAC learning from positive statistical queries. In *Proc. 9th ALT*, pages 112–126. Springer, 1998.

[26] François Denis, Aurélien Lemay, and Alain Terlutte. Residual finite state automata. In *Proc. 18. STACS*, LNCS 2010, pages 144–157. Springer, 2001.

[27] François Denis, Aurélien Lemay, and Alain Terlutte. Learning regular languages using RFSAs. *Theoretical Computer Science*, 313(2):267–294, 2004.

[28] Matthias Ernst. Algorithmic learning of distributions (in german). Master's thesis, Institut für Theoretische Informatik, Universität zu Lübeck, Lübeck, Germany, 2013.

[29] Matthias Ernst, Maciej Liśkiewicz, and Rüdiger Reischuk. Algorithmic learning for steganography: Proper learning of $k$-term DNF formulas from positive samples. In *Proc. 26th ISAAC*, pages 151–162. Springer, 2015.

[30] Vitaly Feldman. A general characterization of the statistical query complexity. In Satyen Kale and Ohad Shamir, editors, *Proc. COLT*, volume 65 of *Proceedings of Machine Learning Research*, pages 785–830, Amsterdam, Netherlands, 2017. PMLR.

[31] Lu Feng, Marta Kwiatkowska, and David Parker. Compositional verification of probabilistic systems using learning. In *Quantitative Evaluation of Systems (QEST), 2010 Seventh International Conference on the*, pages 133–142. IEEE, 2010.

[32] Lu Feng, Marta Kwiatkowska, and David Parker. Automated learning of probabilistic assumptions for compositional reasoning. In *International Conference on Fundamental Approaches to Software Engineering*, pages 2–17. Springer, 2011.

[33] Dana Fisman. . personal communication, 2017.

[34] Michele Flammini, Alberto Marchetti-Spaccamela, and Luděk Kučera. Learning DNF formulae under classes of probability distributions. In *Proc. 5th COLT*, pages 85–92. ACM, 1992.

[35] E Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[36] J. A. Hartigan. Estimation of a convex density contour in two dimensions. *Journal of the American Statistical Association*, 82(397):267–270, 1987.

[37] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[38] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, November 1998.

[39] Michael Kearns, Ming Li, and Leslie Valiant. Learning Boolean formulas. *J. ACM*, 41(6):1298–1328, 1994.

[40] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In *Proc. STOC*, pages 273–282, New York, NY, USA, 1994. ACM, ACM.

[41] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[42] Carsten Kern. *Learning communicating and nondeterministic automata.* PhD thesis, RWTH, Fachgruppe Informatik, 2009.

[43] Adam R Klivans and Rocco Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *J. Comput. System Sci.*, 68(2):303–318, 2004.

[44] Lillian Lee. Learning of context-free languages: A survey of the literature. Technical Report TR-12-96, Harvard Computer Science Group, 1996.

[45] Maciej Liśkiewicz, Rüdiger Reischuk, and Ulrich Wölfel. Grey-box steganography. *Theor. Comput. Sc.*, 505:27–41, 2013.

[46] Oded Maler and Amir Pnueli. On the learnability of infinitary regular sets. *Information and Computation*, 118(2):316–326, 1995.

[47] C. McMullen and J. Shearer. Prime implicants, minimum covers, and the complexity of logic simplification. *IEEE Transactions on Computers*, C-35(8):761–762, 1986.

[48] Dietrich Werner Müller and Günther Sawitzki. Excess mass estimates and tests for multimodality. *Journal of the American Statistical Association*, 86(415):738–746, 1991.

[49] Balaubramaniam Kausik Natarajan. On learning boolean functions. In *Proc. 19th STOC*, pages 296–304. ACM, 1987.

[50] Deborah Nolan. The excess-mass ellipsoid. *Journal of Multivariate Analysis*, 39(2):348–371, 1991.

[51] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.

[52] Wolfgang Polonik. Density estimation under qualitative assumptions in higher dimensions. *Journal of Multivariate Analysis*, 55(1):61–81, 1995.

[53] Liva Ralaivola, François Denis, and Christophe Nicolas Magnan. CN = CPCN. In *Proc. ICML*, pages 721–728, New York, NY, USA, 2006. ACM.

[54] Y. Sakai and A. Maruoka. Learning monotone log-term DNF formulas under the uniform distribution. *Theory of Comput. Systems*, 33(1):17–33, 2000.

[55] Yoshifumi SAKAI and Akira MARUOKA. Learning $k$-term monotone Boolean formulae. *Interdisciplinary Information Sciences*, 3(2):71–80, 1997.

[56] Yoshifumi Sakai, Eiji Takimoto, and Akira Maruoka. Proper learning algorithm for functions of $k$ terms under smooth distributions. *Inform. and Comput.*, 152(2):188–204, 1999.

[57] Haim Shvaytser. A necessary condition for learning from positive examples. *Machine Learning*, 5(1):101–113, 1990.

[58] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[59] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC, pages 1–9, New York, NY, USA, 1973. ACM.

[60] Alexandre B. Tsybakov. On nonparametric estimation of density level sets. *The Annals of Statistics*, 25(3):948–969, 1997.

[61] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[62] Vladimir Naumovich Vapnik and Alexey Jakovlevich Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16(2):264–280, 1971.

[63] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, LNCS 1043, pages 238–266. Springer, 1995.

[64] Karsten Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proc. 3rd COLT*, pages 314–326, San Francisco, 1990. Morgan Kaufmann Publishers Inc.

[65] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[66] T. Yokomori. Machine intelligence 13. chapter Learning Non-deterministic Finite Automata from Queries and Counterexamples, pages 169–189. Oxford University Press, Inc., New York, NY, USA, 1995.

# Curriculum Vitae

Matthias Lutter received his Abitur certificate at the Gymnasium Winsen in 2007. In the same year, he began to study computer science at the University of Lübeck. He received his B.Sc. degree in 2011 and his M.Sc. degree in 2013 from the University of Lübeck, both as the best student of the corresponding year. His master's thesis addresses learning of high-density level areas of probability distributions.

Since 2013 Matthias Lutter is a PhD student at the Institute for Theoretical Computer Science of the University of Lübeck. He has held tutorials mainly in the area of cryptography and IT security. His research area has been computational learning theory with applications to steganography, automata theory, and differential privacy.