Interpretable Anomaly Detection via Discrete Optimization

Simon Lutz[1,4], **Florian Wittbold**[3], **Simon Dierl**[1], **Benedikt Böing**[1], **Falk Howar**[1,2], **Barbara König**[3], **Emmanuel Müller**[1], **Daniel Neider**[1,4]

[1]TU Dortmund University, Germany
[2]Fraunhofer ISST, Germany
[3]University of Duisburg-Essen, Germany
[4]Center for Trustworthy Data Science and Security, UA Ruhr, Germany
{simon.dierl, falk.howar, simon.lutz, daniel.neider}@tu-dortmund.de
{benedikt.boeing, emmanuel.mueller }@cs.tu-dortmund.de
{barbara_koenig, florian.wittbold}@uni-due.de

Abstract

Anomaly detection is essential in many application domains, such as cyber security, law enforcement, medicine, and fraud protection. However, the decision-making of current deep learning approaches is notoriously hard to understand, which often limits their practical applicability. To overcome this limitation, we propose a framework for learning inherently interpretable anomaly detectors from sequential data. More specifically, we consider the task of learning a deterministic finite automaton (DFA) from a given multi-set of unlabeled sequences. We show that this problem is computationally hard and develop two learning algorithms based on constraint optimization. Moreover, we introduce novel regularization schemes for our optimization problems that improve the overall interpretability of our DFAs. Using a prototype implementation, we demonstrate that our approach shows promising results in terms of accuracy and F1 score.

1 Introduction

Anomaly detection (i.e., identifying patterns in data that do not conform to expected behavior [Chandola *et al.*, 2009]) has evolved into a vibrant subfield of machine learning and data science. Its many application domains include cyber security, law enforcement, medicine, and fraud detection, to name but a few. Essentially, anomaly detection is applied whenever the correct functioning of a complex system is essential for safety or financial reasons.

The recent advances in machine learning have prompted the development of a host of deep learning techniques for anomaly detection (see the section on related work for a brief overview). Many of these operate in an unsupervised setting, where the data is unlabeled (i.e., what does and does not constitute an anomaly is unknown). To make this task tractable, one usually requires further auxiliary information, such as an upper and lower bound on the expected number of anomalies in the data. Another approach is to fine-tune specific hyper-parameters of the learning algorithm to prevent it from producing a degenerate solution (i.e., one that classifies all or no data as anomalies).

While deep learning methods for anomaly detection have shown excellent performance, their inherent complexity and black-box nature make their decisions often hard to understand. However, a lack of interpretability is often a severe obstacle to employing anomaly detection in practice. Consider, for instance, a monitor for vital signs in an intensive care unit. If an anomaly is detected, it is imperative to understand the reason to initiate the proper treatment quickly.

This paper focuses on unsupervised learning of inherently interpretable anomaly detectors for sequential data. More specifically, we consider the task of learning a deterministic finite automaton (DFA) from a given multi-set of unlabeled sequences, which accepts the anomalies and rejects the normal data. This specific choice of model is motivated by Shvo *et al.* [2021], who have demonstrated that DFAs are inherently interpretable models for classifying sequential data. Moreover, the authors have shown that DFAs can match the performance of deep LSTM networks on various tasks.

We consider two unsupervised learning setups. In the first setting, we are given a finite multi-set $\mathcal{S}$ of unlabeled sequences and two natural numbers $\ell, u \in \mathbb{N}$ with $\ell \leq u \leq |\mathcal{S}|$. The task is then to learn a minimal DFA that accepts at least ℓ and at most u sequences from $\mathcal{S}$. Minimality refers to a minimal number of states and is a common requirement in automata learning [Biermann and Feldman, 1972; Heule and Verwer, 2010; Leucker and Neider, 2012; Neider *et al.*, 2021]. Here, we use it to ensure high interpretability in the sense of Occam's razor (i.e., smaller models are generally easier to understand than larger ones [Shvo *et al.*, 2021; Roy *et al.*, 2020]). The parameters ℓ and u, on the other hand, serve as an estimate for the lower and upper number of anomalies in the data set and are used to prevent degenerate DFAs (i.e., DFAs that accept or reject all sequences). To give more intuition, assume that we are given n sequences and know that 10-20% of the sequences are typically anomalies. Then our aim is to learn a minimal automaton that accepts between $\ell = 0.1 \cdot n$ and $u = 0.2 \cdot n$ of the sequences. We operate under the assumption that it is presumably easier to separate regular sequences and outliers via a regular language, rather than mix them. Hence by looking for an automaton that is as compact as possible, the classification of anomalies is performed automatically:

The second setting alleviates the user's burden to specify both ℓ and u. Instead, it assumes a multi-set $\mathcal{S}$, a size n of the resulting DFA, but only one bound to be given, say $\ell \in \mathbb{N}$. In this setting, the task is to learn a DFA of size n that accepts the smallest number $k \geq \ell$ of sequences from $\mathcal{S}$. In other words, ℓ serves as a lower bound on the assumed number of anomalies in the given data set.

While this setting only requires one of the bounds to be known or specified, it requires the user to specify the size n of the resulting DFA as input. Generally, the number of states n should be chosen carefully in this setting as numbers too high could hinder interpretability while, if n is too small, the resulting DFAs may not be able to separate anomalies from normal sequences.

Our contributions are fourfold. First, we show that these learning problems are computationally hard. In fact, the first problem is NP-complete and the second one lies within the class NPO. These results are in line with the classical learning of DFAs from positive and negative data, which is known to be NP-complete as well [Gold, 1978].

Second, we develop two learning algorithms, one for each setting. Since both settings are computationally hard, our algorithms follow a common approach in automata learning and reduce the tasks into a series of constraint optimization problems. These problems can then be solved by highly-optimized mixed-integer programming solvers (Gurobi in our case [Gurobi Optimization, LLC, 2022]).

Third, we propose novel regularization terms to enhance the interpretability of the learned DFAs. In particular, we show how to augment our constraint optimization problems to maximize the number of self-loop and parallel edges. This approach is orthogonal to the original encoding and can, in principle, also be applied to the classical passive learning algorithms for finite-state machines.

Fourth, we evaluate our two algorithms empirically on an benchmark based on the ALFRED data set [Shridhar *et al.*, 2020]. We examine both the runtime and the anomaly detection performance for different configuration options and uncertainty w.r.t. the anomaly frequency.

Related Work

A large number of different anomaly detection methods can be found in the literature. For instance, the survey by Chandola *et al.* [2009] distinguishes between methods that are classification-based, clustering-based, based on nearest neighbors, statistical, information-theoretic, and spectral. The algorithms developed in this paper are learning-based.

Within the learning-based methods, deep learning has evolved as a powerful technique. Examples include variational autoencoders [Xu *et al.*, 2018; Li *et al.*, 2021], adversarial neural networks [Li *et al.*, 2019; Geiger *et al.*, 2020], and the paradigm of "one-class classification" [Ruff *et al.*, 2018]. Another way to categorize learning-based methods is their setup. Here, one typically distinguishes between supervised learning (i.e., the data is labeled as normal or anomalous), semi-supervised learning (i.e., the combination of a small amount of labeled data with a large amount of unlabeled data), and unsupervised learning (i.e., the data is not labeled). Since obtaining labeled data for anomaly detection is often difficult or dangerous, this paper operates in an unsupervised setting.

A substantial drawback of deep neural networks is their black-box nature and high complexity, which makes their decision-making intransparent and hard to understand. Hence, interpretable machine learning has evolved recently (see Molnar [2022] for an introduction). A key distinction in this field (among others) is whether one explains a complex model post-hoc or trains an inherently interpretable one. We follow the second paradigm, although inherently interpretable models can have a performance penalty.

Recently, various papers have argued for deterministic finite automata (DFAs) as a powerful yet interpretable model for sequence classification [Hammerschmidt *et al.*, 2016; Shvo *et al.*, 2021]. We support this proposition but want to point out that Shvo *et al.*'s work is fundamentally different from ours: Shvo *et al.* consider a supervised setup, whereas we consider an unsupervised one. In fact, not much research has been devoted to unsupervised automata learning so far (see Carmel and Markovitch [1995] for a notable exception), let alone to interpretable anomaly detection.

Automata learning has a long history, dating back to the 1970s [Biermann and Feldman, 1972; Trakhtenbrot and Barzdin, 1973]. One typically distinguishes between active learning and passive learning. In active learning [Angluin, 1987], the learning algorithm aims to learn a minimal DFA (in terms of the number of states) by querying an information source called the teacher. In passive learning [Biermann and Feldman, 1972], on the other hand, the learning algorithm seeks to learn a minimal DFA from a given set of labeled data. Although our learning setup is unsupervised, it resembles the passive one.

Gold [1978] showed that passive learning, as defined above, is computationally hard (i.e., the corresponding decision problem is NP-complete). Thus, learning algorithms that use constraint solving have become the de facto standard [Heule and Verwer, 2010; Neider, 2012; Neider *et al.*, 2021]. Since our learning task is also computationally hard, our two algorithms follow a similar approach and translate the learning task into a sequence of optimization problems in mixed integer linear programming.

2 Preliminaries

In this paper we address the task of detecting anomalies in sequential data which we view as a multi-set of unlabeled sequences ranging over a finite set of symbols. While this is a restriction in general, it is a common approach in data mining to use methods such as Symbolic aggregate approximation (SAX) [Lin *et al.*, 2007] to discretize continuous data. In order to emphasize this restriction linguistically we revert to a standard notation of automata theory by referring to a sequence $w = a_1 \dots a_n$ as a *finite word*. Moreover, we call the nonempty, finite set of *symbols* over which these words can range an *alphabet* Σ. The sequence without any symbols, also referred to as *empty word*, is denoted by ϵ. Furthermore, denote the set of all words over an alphabet Σ as Σ^*. In the remainder of this paper we will refer to the multi-set of sequential data $\mathcal{S} = \{w_1, \ldots, w_n\}$ as a *sample*. Since a word w can be contained multiple times in a sample $\mathcal{S}$, we denote the number of occurrences of w in $\mathcal{S}$ as $\mathcal{S}(w)$.

From a given sample we learn a *deterministic finite automata (DFA)* as an anomaly detector. DFAs are commonly known in computer science, easy to understand even for non-experts, and inherently interpretable [Shvo *et al.*, 2021]. Formally, a DFA is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ where Q is a finite set of states, Σ is a finite set of (input) symbols, $q_I \in Q$ is the *initial state*, $\delta : Q \times \Sigma \to Q$ is the *state-transition function*, and $F \subseteq Q$ is a set of accepting states. The *size* of a DFA is defined to be the number of its states $|Q|$. A *run* on a word $w = a_1 \ldots a_n$ is a sequence of states $q_0 \ldots q_n$ such that $q_0 = q_I$ and $q_i = \delta(q_{i-1}, a_i)$ for $i \in \{1, \ldots, n\}$. We call a run *accepting* if $q_n \in F$, otherwise *rejecting*. In order to indicate whether a word w is accepted or rejected by a DFA $\mathcal{A}$ we define the indicator function

$$\mathcal{A}(w) \coloneqq \begin{cases} 1 & \text{if } \mathcal{A} \text{ accepts } w, \\ 0 & \text{otherwise} \end{cases}$$

The *language* of a DFA $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of all words accepted by $\mathcal{A}$.

As mentioned in the introduction we reduce the task of learning a DFA into a series of *mixed-integer linear programming (MILP)* problems. Let *Var* be a finite set of real variables. An MILP problem consists of two parts, a linear function over the variables, referred to as *objective function* obj, and a conjunction of *linear constraints* Φ on these variables. The solution to such a MILP problem is an assignment $Var \to \mathbb{R}$, referred to as (feasable) *model*, such that the value of the objective function obj is optimal (i.e. minimal/maximal respectively) while satisfying Φ (i.e. all constraints).

3 Problem Formulation

In this paper we address unsupervised anomaly detection for time-series in a setting where an interpretable and/or queryable detector is required. In particular, we learn a deterministic finite automaton detecting the anomalies in a set of sequences.

As mentioned in the introduction we solve this problem in two unsupervised learning setups which require different amount of prior knowledge about the distribution of normal and anomalous sequences in the sample $\mathcal{S}$. This knowledge could be gained from previous experiments with similar data sets or empirical testing to name but a few examples. The first setup, which we refer to as *Two-Bound DFA Learning*, is motivated by the fact that even though the precise number of anomalies in a sample is unknown in most cases, often a rough estimate on the proportion of anomalies is known. With this estimate a lower and an upper bound on the number of anomalies in the sample can be computed with high certainty. We assume these bounds to be given as two natural numbers $\ell, u \in \mathbb{N}$ with $\ell \leq u \leq |\mathcal{S}|$. Then the task is to learn a minimal DFA which accepts at least ℓ and at most u sequences from $\mathcal{S}$. We formally state this problem as:

Problem 1 (Two-Bound DFA Learning Problem)**.** *Given a multi-set of words $\mathcal{S} = \{w_1, \ldots, w_n\}$ and two natural numbers $\ell, u \in \mathbb{N}$ with $\ell \leq u \leq |\mathcal{S}|$, construct a DFA $\mathcal{A}$ which accepts at least ℓ and at most u words from $\mathcal{S}$.*

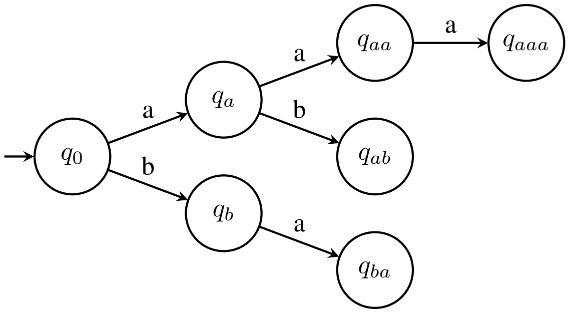

Figure 1: Prefix tree for the sample (aa, ab, ba, aaa)

Notice that one may not always find a solution for this problem. This is illustrated using the following example.

Example 1. *Consider the bounds $\ell = u = 1$ and the sample $\mathcal{S}$ which only contains the word $'a'$ twice. Being deterministic, every DFA has to either accept both copies of the word $'a'$ or reject them both. Hence there does not exist a DFA fulfilling the bounds in this case.*

Although there may not always be a solution for Problem 1 it is decidable. Towards decidability we start by representing the sample as a *prefix tree* [De la Higuera, 2010]. A prefix tree for a sample $\mathcal{S}$ is a partial DFA (i.e. some transitions are unspecified) without final states such that after reading a word $w \in \mathcal{S}$ the DFA is in a unique state q_w. An example of a prefix tree is displayed in Figure 1. We complete this partial DFA by adding an additional state to which we target all unspecified transitions. To decide whether there exists a DFA accepting at least ℓ and at most u words we iterate over all combinations of final states and check the number of accepted words in each case. Since there is a unique state for each word in $\mathcal{S}$ at least one of these DFAs fulfills the bounds or we can conclude that none exists.

However, a DFA constructed this way suffers from two main issues making it unsuitable as an interpretable anomaly detector. On the one hand it overfits the sample $\mathcal{S}$ and thus poorly generalizes to unseen data. On the other hand it becomes rather large, hindering interpretability in the sense of Occam's razor. To overcome these issues we propose an algorithm which not only constructs a DFA fulfilling the given bounds, but also constructs a DFA of minimal size (if one exists). However, by requiring minimality the problem becomes computationally hard.

More precisely, we can show that the problem whether there exists a DFA with n states for Problem 1 is NP-complete. We begin by showing that the problem is in NP, as formalized next.

Theorem 1. *Given a multi-set of words $\mathcal{S}$, two natural numbers $\ell, u \in \mathbb{N}$ with $\ell \leq u \leq |\mathcal{S}|$, and a natural number n (given in unary), a non-deterministic Turing machine can compute in polynomial time whether there exists a DFA $\mathcal{A}$ with n states which accepts at least ℓ and at most u words from $\mathcal{S}$ (i.e., the problem lies in NP).*

Proof of Theorem 1. Let $\mathcal{S}, \ell, u$, and n be given. As n is unary, a non-deterministic Turing machine can guess an automaton $\mathcal{A}$ with n states. The number of accepted words from $\mathcal{S}$ can then be computed in polynomial time by simulation. Finally, checking whether this number is at least ℓ and at most u is also possible in polynomial time, showing that the problem lies in NP. □

Next, we we show NP-hardness of Problem 1, as formalized below.

Theorem 2. *Given a multi-set of words $\mathcal{S}$, two natural numbers $\ell, u \in \mathbb{N}$ with $\ell \leq u \leq |\mathcal{S}|$, and a natural number n (given in unary), the problem of finding a DFA $\mathcal{A}$ with n states which accepts at least ℓ and at most u words from $\mathcal{S}$, is* NP*-hard.*

To proof Theorem 2, we use the NP-completeness of the following problem (see [Garey and Johnson, 1979]).

Problem 2. *Given finite disjoint sets of words $P, N \subseteq \Sigma^*$ and a unary number $k \in \mathbb{N}_0$, does there exist a deterministic finite automaton $\mathcal{A}$ with k states such that $\mathcal{A}$ accepts all words in P and rejects all words in N.*

As detailed in the proof below, NP-hardness (and thus NP-completeness) of our problem follows by reduction from Problem 2. This reduction makes use of the multi-set structure of the samples to encode positive and negative words from Problem 2 in different multiplicities in the multi-set, which can be distinguished by Problem 1.

Proof of Theorem 2. We show this by a many-one reduction from Problem 2. Let finite disjoint sets $P, N \subseteq \Sigma^*$ and a unary number $k \in \mathbb{N}_0$ be given. We construct an instance of our problem as follows:

- $n := k$;
- $\mathcal{S}(w) := \begin{cases} |N| + 1 & \text{if } w \in P, \\ 1 & \text{if } w \in N, \\ 0 & \text{otherwise,} \end{cases}$

  for $w \in \Sigma^*$, i.e., the multi-set contains exactly $(|N| + 1)$-times all words in P and once all words in N;
- $\ell = u := |P|(|N| + 1)$.

This construction can be done in polynomial time.

Furthermore, if there exists a DFA $\mathcal{A}$ with n states solving Problem 1 for $\mathcal{S}, \ell$, and u as above, i.e., accepting exactly $|P|(|N| + 1)$ words from $\mathcal{S}$, we have that

$$0 \equiv \sum_{w \in P \cup N} \mathcal{S}(w) \cdot \mathcal{A}(w) \equiv \sum_{w \in N} \mathcal{A}(w) \pmod{|N| + 1}$$

Recall that $\mathcal{S}(w)$ denotes the number of occurrences of w in $\mathcal{S}$ and $\mathcal{A}(w)$ indicates whether w is accepted by $\mathcal{A}$. This shows that $\mathcal{A}$ rejects all words in N, and, thus

$$|P|(|N| + 1) = \sum_{w \in P} \mathcal{S}(w) \cdot \mathcal{A}(w) \Rightarrow |P| = \sum_{w \in P} \mathcal{A}(w),$$

showing that $\mathcal{A}$ accepts all words in P whence $\mathcal{A}$ is also a solution to the instance (P, N, k) of Problem 2.

Similarly, if there exists a DFA $\mathcal{A}$ solving the instance (P, N, k) of Problem 2, we have that

$$\sum_{w \in \Sigma^*} \mathcal{S}(w) \cdot \mathcal{A}(w) = (|N| + 1) \sum_{w \in P} \mathcal{A}(w) = |P|(|N| + 1),$$

whence $\mathcal{A}$ is also a solution with n states to the instance $(\mathcal{S}, \ell, u)$ of Problem 1.

All in all, this shows the reduction from Problem 2, and thus NP-hardness of finding a solution of Problem 1 with n states. □

In the first setup, we require the user to provide a lower and an upper bound on the number of anomalies in a given sample. However this requirement may be to strong. So in the second setup, which we refer to as *Single-Bound DFA Learning*, we reduce the amount of prior knowledge compared to the first case by alleviating the user's burden to specify both ℓ and u. Instead, we assume to be given a sample $\mathcal{S}$ and only one parameter, say ℓ (the case in which u is given is analogously). In this case the task is to construct a minimal DFA which accepts at least ℓ words from $\mathcal{S}$. In contrast to Problem 1, this problem always has a trivial solution since the DFA which accepts all words in S only has size 1 and fulfills the bound. However, this DFA underfits the sample $\mathcal{S}$ and thus does not capture the underlying structure. To reduce underfitting, we apply a technique commonly applied in automata learning. We construct a DFA of a fixed size which accepts the smallest number $k \geq l$ of words from the sample. By providing this size as an additional parameter n the user can regularize the trade-off between avoiding underfitting (larger) and interpretability (smaller). We propose an algorithm solving this problem which we formally state as:

Problem 3 (Single-Bound-Learning-Problem)**.** *Given a multi-set of words $\mathcal{S} = \{w_1, \ldots, w_n\}$ and two natural numbers $\ell, n \in \mathbb{N}$ with $\ell \leq |\mathcal{S}|$, construct a DFA $\mathcal{A}$ of size n which accepts the smallest number $k \geq l$ of words from $\mathcal{S}$.*

This problem is the optimization version of Problem 1, thus it is also computationally hard. In fact, from Problem 1 being NP-complete it immediately follows that Problem 3 lies within the complexity class NPO (i.e. the class of optimization problems whose decision variant lies in NP)

4 Learning via Discrete Optimization

Following a common approach in automata learning we learn a DFA in both setups by reducing the tasks to a set of constraint optimization problems and solve them using state-of-the-art mixed-integer programming solvers (Gurobi in our case [Gurobi Optimization, LLC, 2022]).

4.1 Two-Bound DFA Learning

Recall that in the first setup we are given a sample $\mathcal{S}$ and two bounds $\ell, u \in \mathbb{N}$ with $\ell \leq u \leq |\mathcal{S}|$. In order to learn a minimal DFA which fulfills these bounds, we apply a technique commonly used in automata learning to ensure minimality. The idea is to encode the problem for an automaton of fixed size n such that the encoding has two key properties:

- There exists a feasible model if and only if there exists a DFA of size n fulfilling the bounds on the acceptance

- This model contains sufficient information to construct such a DFA

Starting with an automaton of size one and increasing the size whenever there is no feasible model guarantees to produce the minimal solution.

We now describe the MILP model we use to learn a DFA of size n. Since we just check the existence of a suitable DFA in the first setup, we can choose any constant objective function, e.g. $obj = 1$. The set of linear inequalities $\Phi^n_{\mathcal{S},\ell,u} = \Phi^n_{\mathcal{A}} \wedge \Phi_{\mathcal{B}}$ consists of two kinds of constraints: *automata constrains* $\Phi^n_{\mathcal{A}}$, which encode a DFA of size n and the runs on all words from the sample, and *bound constraints* $\Phi_{\mathcal{B}}$ encoding the bounds on the acceptance. Throughout these constraints we bound the introduced variables to only take on integer values between (and including) 0 and 1, thus simulating boolean variables.

Automata constraints $\Phi^n_{\mathcal{A}}$ The automata constraints are motivated by the SAT encoding of Biermann and Feldman [Biermann and Feldman, 1972]. Without loss of generality, the states of the DFA form the set $Q = \{q_0, \dots, q_{n-1}\}$ where q_0 is the initial state. The alphabet Σ of the DFA is the set of all symbols appearing in the sample $\mathcal{S}$. To encode the transitions of the DFA we introduce variables $\delta_{q,a,q'}$ for $q, q' \in Q$ and $a \in \Sigma$. Intuitively, the variable $\delta_{q,a,q'}$ will be set to 1 if and only if the DFA has a transition from state q to state q' on reading a. Furthermore, we introduce variables f_q for $q \in Q$ which indicate whether a state q is a final state. To encode the runs of the DFA we start by computing the set of all prefixes in the sample $Pref(\mathcal{S}) = \{w \mid ww' \in \mathcal{S} \text{ and } w' \in \Sigma^*\}$. We then introduce a third kind of variable: $x_{w,q}$ for all $w \in Pref(\mathcal{S})$ and $q \in Q$. Intuitively, these variables indicate that after reading the prefix w the DFA is in state q.

We now impose constraints on these variables to encode a DFA and its runs. Being deterministic there must be precisely one transition for each state q and symbol a which we can model by the following constraint:

$$\sum_{q' \in Q} \delta_{q,a,q'} = 1 \qquad \forall q \in Q, \forall a \in \Sigma \tag{1}$$

Furthermore, after reading a word w the DFA can only be in one state

$$\sum_{q \in Q} x_{w,q} = 1 \qquad \forall w \in Pref(S) \tag{2}$$

After reading the empty word ϵ the DFA is in the initial state which we defined to be q_0. This is encoded as:

$$x_{\epsilon,q_0} = 1 \tag{3}$$

Finally, we encode a run based on the following implication:

$$x_{w,q} \wedge \delta_{q,a,q'} \rightarrow x_{wa,q'}$$

Intuitively, this implication states that when the DFA is in some state q after reading the word w and there is a transition from q to q' on reading the symbol a, then the DFA is in state q' after reading the word wa. As a constraint in MILP we get:

$$\begin{gathered} x_{w,q} + \delta_{q,a,q'} - 1 \leq x_{wa,q'} \\ \forall q, q' \in Q, a \in \Sigma, \forall wa \in Pref(S) \end{gathered} \tag{4}$$

We define the automata constraints $\Phi^n_{\mathcal{A}}$ to be the conjunction of Equations 1 to 4 which concludes the encoding of a DFA of size n and its runs.

Bound constraints $\Phi_{\mathcal{B}}$ To impose constraints on the number of accepted words we need to track whether a word w is accepted. This is the case if and only if after reading w the DFA is in some state q and this state is final. Intuitively, we could express this case as $x_{w,q} \cdot f_q$, however this is not linear and thus not a valid MILP constraint. Instead, we exploit the fact that for Boolean variables the multiplication $x_{w,q} \cdot f_q$ is equivalent to the formula $x_{w,q} \wedge f_q$. By introducing fresh variables $\alpha_{w,q}$ for $w \in \mathcal{S}$ and $q \in Q$ to store the result the later one can be modeled by the following set of constraints:

$$\begin{aligned} \alpha_{w,q} &\geq x_{w,q} + f_q - 1 && \forall w \in \mathcal{S}, \forall q \in Q \\ \alpha_{w,q} &\leq x_{w,q} && \forall w \in \mathcal{S}, \forall q \in Q \\ \alpha_{w,q} &\leq f_q && \forall w \in \mathcal{S}, \forall q \in Q \end{aligned} \tag{5}$$

Intuitively, the variables $\alpha_{w,q}$ indicate whether a word w is accepted by the DFA (in the state q). Relying on these variables we can encode the bounds on the acceptance as

$$\sum_{w \in S} \sum_{q \in Q} \alpha_{w,q} \geq \ell \tag{6}$$

$$\sum_{w \in S} \sum_{q \in Q} \alpha_{w,q} \leq u \tag{7}$$

Then the bound constraints $\Phi_{\mathcal{B}}$ are the conjunction of Equations 6 and 7.

After introducing the MILP model we employ it to construct the minimal DFA which fulfills the given bounds on the acceptance. The idea is to check feasibility of the MILP problem with constant objective function $obj = 1$ and linear inequalities $\Phi^n_{\mathcal{S},\ell,u}$ for increasing n until either a solution is found or we reach $n = |Pref(\mathcal{S})| + 2$. As argued above when proving decidability of problem 1 the size of the prefix tree is a natural upper bound for the size of the DFA. Therefore, we can conclude that there exists no DFA fulfilling the given bounds on the sample when we exceed this size. In the case where a feasible model exists for some size n we construct the corresponding DFA from this model based on the variables $\delta_{q,a,q'}$ and f_q. This procedure is described by Algorithm 1.

Algorithm 1 Learning with two bounds

```
Input: Sample $\mathcal{S}$, Bounds $\ell, u \in \mathbb{N}$
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    Construct $\Phi^n_{\mathcal{S},\ell,u} = \Phi^n_{\mathcal{A}} \wedge \Phi_{\mathcal{B}}$
    Set $obj = 1$
    if $obj, \Phi^n_{\mathcal{S},\ell,u}$ has a feasible model (say $m$) then
        return Construct DFA $\mathcal{A}$ using $m$
    end if
until $n = |Pref(\mathcal{S})| + 2$
return There exists no DFA fulfilling the given bounds
```

The correctness of this algorithm is established by the following theorem:

Theorem 3. *Given a sample $\mathcal{S}$ and two natural numbers $\ell, u \in \mathbb{N}$ with $\ell \leq u \leq |\mathcal{S}|$, Algorithm 1 terminates and outputs a minimal DFA $\mathcal{A}_{\mathcal{S}}$ which accepts at least ℓ and at most u words from $\mathcal{S}$, if such a DFA exists.*

Proof. We prove Theorem 3 in three steps: First, we explain how we construct a DFA from a feasible model and proof that this automaton is well-defined and solves Problem 1. Afterwards, we show that a feasible model exists for a size n if and only if there exists a DFA of that size fulfilling the bounds. In the end, we establish termination and show that Algorithm 1 finds a DFA fulfilling the bounds if such a DFA exists. By construction this DFA is minimal.

For now let us assume we found a feasible model for some size n. We show that the DFA $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ given by

- $Q = \{q_0, \dots, q_{n-1}\}, q_I = q_0$;
- Σ the symbols present in $\mathcal{S}$;
- $\delta : Q \times \Sigma \to Q, (q, a) \mapsto q'$ for $q \in Q, a \in \Sigma$, and $q' \in Q$ such that $\delta_{q,a,q'} = 1$;
- $F \coloneqq \{q \in Q \mid f_q = 1\}$;

is well-defined and solves Problem 1. First of all, Constraint 1 ensures that the state-transition function δ is well-defined while f_q simulating Boolean variables further ensures that F is well-defined. Next we show that the variables $x_{w,q}$ correspond to the runs of words w from $\mathcal{S}$ in this well-defined DFA. More precisely, we show that $x_{w,q} = 1$ if $\mathcal{A}$ is in state $q \in Q$ after reading $w \in \mathit{Pref}(\mathcal{S})$ by induction over the prefix length $k = |w|$ (where we w.l.o.g. assume $\mathcal{S}$ to be non-empty). Note that Constraint 2 then also ensures the opposite direction, i.e., that $\mathcal{A}$ is in state $q \in Q$ after reading $w \in \mathit{Pref}(\mathcal{S})$ if $x_{w,q} = 1$ (as $x_{w,\tilde{q}} = 1$ for the true state $\tilde{q}$ and thus $x_{w,q'} = 0 \neq 1$ for all other states $q' \neq \tilde{q}$).

Base case. The only prefix of length $k = 0$ obviously being the empty word ε, Constraint 3 gives that $x_{\varepsilon,q_0} = 1$, showing the statement for prefixes of length $k = 0$.

Induction step. Assuming that the statement holds for any prefix of length less or equal to k. If no prefix of length $k+1$ exists in $\mathcal{S}$, the induction is closed and the statement is proven for all prefixes. Assume now that $wa \in \mathit{Pref}(\mathcal{S})$ is a prefix of length $k+1$. Then w is a prefix of length k and thus fulfills for the state q reached after reading w in $\mathcal{A}$ that $x_{w,q} = 1$. Writing $q' \coloneqq \delta(q, a)$, Constraint 4 and the definition of δ then give

$$x_{w,q} + \delta_{q,a,q'} - 1 = 1 \leq x_{wa,q'}$$

whence also $x_{wa,q'} = 1$ as a Boolean variable. As wa was an arbitrary prefix of length $k+1$, this concludes the induction.

While this proofs that the DFA $\mathcal{A}$ is well-defined, the bound constraints $\Phi_{\mathcal{B}}$ ensure that the number of accepted words is above the lower bound ℓ and below the upper bound u. Hence, the DFA $\mathcal{A}$ is well-defined and solves Problem 1

In a second step, we show that our MILP problem has a feasible model for size n if and only if there exists a DFA of size n that accepts at least ℓ and at most u words from $\mathcal{S}$.

$(\Rightarrow)$: Given a feasible model for size n, we construct a DFA $\mathcal{A}$ as explained above. As displayed this DFA $\mathcal{A}$ is well-defined and fulfills the bounds on the acceptance.

$(\Leftarrow)$: Given a DFA $\mathcal{A}$ of size n which accepts at least ℓ and at most u words from $\mathcal{S}$, let the model be given by the natural interpretation of the variables:

$$\delta_{q,a,q'} \coloneqq \begin{cases} 1 & \text{if } \delta(q,a) = q', \\ 0 & \text{otherwise,} \end{cases}$$

$$f_q \coloneqq \begin{cases} 1 & \text{if } q \in F, \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{w,q} \coloneqq \begin{cases} 1 & \text{if } \mathcal{A} \text{ is in state } q \text{ after reading } w, \\ 0 & \text{otherwise,} \end{cases}$$

$$\alpha_{w,q} \coloneqq \begin{cases} 1 & \text{if } \mathcal{A} \text{ is in state } q \text{ after reading } w \text{ and } q \in F, \\ 0 & \text{otherwise,} \end{cases}$$

Being deterministic the DFA $\mathcal{A}$ and thus any model constructed this way obviously fulfills Constraints 1 to 4. Furthermore, the definition of $\alpha_{w,q}$ ensures that the set of constraints 5 is fulfilled and that $\sum_{w\in\mathcal{S}}\sum_{q\in Q}\alpha_{w,q} = \sum_{w\in\mathcal{S}}\sum_{q\in F} x_{w,q}$ corresponds to the amount of words in $\mathcal{S}$ which are accepted by $\mathcal{A}$ whence by assumption both Constraint 6 and Constraint 7 are fulfilled. Hence this model is feasible for our MILP problem for size n.

In order to conclude the proof of Theorem 3 we now show that Algorithm 1 terminates and finds a DFA fulfilling the bounds if such a DFA exists. Termination itself is straight forward. The algorithm iterates over increasing sizes until a feasible model is found or $n = |\mathit{Pref}(\mathcal{S})| + 2$ is reached in which case it concludes that no DFA fulfilling the bounds exists. Since solving the MILP problem for each n is computable, the algorithm thus always terminates. Furthermore, if a feasible model is found, we showed above that the algorithm returns a DFA fulfilling the bounds. On the other hand, as explained in the decidability proof of Problem 1, the size of the prefix tree is a natural upper bound for the minimal DFA fulfilling the bounds. In the prefix tree, the run on each word w in the sample $\mathcal{S}$ leads to a unique state q_w. Therefore, we can check accepting each combination of words from $\mathcal{S}$ by making the corresponding states a final or non-final state respectively. If no such combination fulfills the bounds, we can conclude that no DFA exists which fulfills the bounds. Since the prefix tree can have unspecified transitions, we may need one additional state to which we target all these unspecified transitions in order to construct a well-defined DFA. Hence, there either exists a DFA of size $n = |\mathit{Pref}(\mathcal{S})| + 1$ or none at all. By the equivalence proven above, we thus have that not finding a feasible model until $n = |\mathit{Pref}(\mathcal{S})| + 2$ shows that truly no DFA fulfilling the bounds exists. This concludes the proof of termination and correctness and, thus, of Theorem 3.

We have shown that Algorithm 1 terminates and finds a DFA which fulfills the bounds on the acceptance, if such a DFA exists. We displayed that such a DFA of size n exists if and only if our MILP problem for size n has a feasible model. Furthermore, we have shown how to construct this DFA given a feasible model. □

Finally, let us investigate the number of constraints in our MILP model. This number depends on multiple factors:

- The size n of the automaton to be constructed
- The size of the alphabet Σ over which the words in the sample $\mathcal{S}$ range

- The number of unique words in $\mathcal{S}$
- The size of the prefix tree of $\mathcal{S}$

Let $|\Sigma|$ now denote the size of the alphabet and $p = |Pref(\mathcal{S})|$ denote the size of the prefix tree of $\mathcal{S}$. Note that the number of unique words in $\mathcal{S}$ is a lower bound for the size of the prefix tree. Then, we obtain the following remark.

Remark 1. *The number of constraints in the MILP problem is in $\mathcal{O}(n^2 \cdot |\Sigma| \cdot p)$.*

4.2 Single-Bound DFA Learning

To not clutter this section to much we will only describe the encoding for the case where the lower bound ℓ is given. The case in which the upper bound is given is analogous. Recall: If the lower bound ℓ is given the task is to construct a DFA of a fixed size n that minimizes acceptance above ℓ. Analogously to the first setup we encode the DFA and the runs on all words in the sample using the same set of variables and automata constraints $\Phi^n_{\mathcal{A}}$ as above. Furthermore, we use the same idea to ensure that the number of accepted sequences is larger than the lower bound: We introduce variables $\alpha_{w,q}$ and add Constraints 5 and 6. For the remainder of this section let Φ_ℓ denote the conjunction of these constraints. In contrast to Two-Bound-Learning, we not only want to check the existence of a DFA, but also to find the one which minimizes acceptance. Towards this goal we use the following objective function:

$$obj = \min \sum_{w \in S} \sum_{q \in Q} \alpha_{w,q} \tag{8}$$

which minimizes the number of accepted words from $\mathcal{S}$ Then we employ this MILP model to construct a DFA of a given size n which minimizes acceptance above the lower bound. This procedure is described by Algorithm 2. The correctness of this algorithm is established by the following theorem:

Algorithm 2 Learning with a single bound

```
1: Input: Sample $\mathcal{S}$, Bound $\ell \in \mathbb{N}$, Size $n \in \mathbb{N}$
2: Construct $\Phi^n_{\mathcal{S},\ell} = \Phi^n_{\mathcal{A}} \wedge \Phi_\ell$
3: Set $obj = \min \sum_{w \in S} \sum_{q \in Q} \alpha_{w,q}$
4: Compute optimal model minimizing $obj$ with respect to $\Phi^n_{\mathcal{S},\ell}$, say $m$
5: return Construct DFA $\mathcal{A}$ using $m$
```

Theorem 4. *Given a sample $\mathcal{S}$ and two natural numbers $\ell, n \in \mathbb{N}$ with $\ell \leq |\mathcal{S}|$, Algorithm 2 terminates and outputs a a DFA $\mathcal{A}_\mathcal{S}$ of size n which accepts the smallest number $k \geq l$ of words from $\mathcal{S}$.*

We omit the proof of this theorem which is similar to the proof of Theorem 3.

5 Interpretability

Even though automata are generally regarded as interpretable models [Shvo *et al.*, 2021], they can become quite complicated to grasp if too many different transisitions to different states are possible. Moreover, there may be multiple valid models for the same training data. Therefore we introduce heuristics aimed at reducing their complexity and making them more readable for humans. While [Shvo *et al.*, 2021] introduced similar techniques as regularization terms, we adapt them to improve the interpretability of the resulting models as demonstrated in Figure 2. Note though, that the models obtained by different simplification heuristics need not be equivalent and that thus the modifications may impede or even improve the models accuracy. In essence the following heuristics aim to visually declutter the graphical representation of the resulting model and thus to focus the user's attention. They are implemented by adding a penalty term to the objective function.

- *Sink states*: We favor solutions that have a so-called sink state which can never be left once it is reached. By our design, all words ending in the sink state are classified as normal.

  To introduce a sink state q_1 (i.e., a non-final state with only self-loops) to the automaton, we add the following constraints:

  $$\begin{aligned} \delta_{q_1,a,q_1} &= 1 \qquad\qquad \forall a \in \Sigma \\ f_{q_1} &= 0 \end{aligned}$$

  Moreover we add

  $$\lambda_s \cdot \left(\sum_{q \in Q, a \in \Sigma} 1 - \delta_{q,a,q_1} \right)$$

  to the objective function, which penalizes each transition not targeting the sink state. The parameter $\lambda_l \in \mathbb{R}$ is a weight term to be chosen by the user. Note that we need to have at least two states in our DFA to have a sink-state.

- *Self-loops*: By penalizing transitions to other states we obtain models with a lot of self-loops. By convention, those are omitted in the graphical representation.

  To increase the number of self-loops for the resulting automata, we add

  $$\lambda_l \cdot \left(\sum_{q \in Q, a \in \Sigma} \sum_{q' \in Q \setminus \{q\}} \delta_{q,a,q'} \right)$$

  to the objective function. This term penalizes each transition where the source state is different from the destination state (i.e., not a self-loop). Here, $\lambda_l \in \mathbb{R}$ is a weight term to be chosen by the user.

- *Parallel edges*: Similar to self-loops we prefer solutions where there is only one successor state. Thus, the automata will transition to the same state regardless of the next input $a \in \Sigma$.

  Similar to self-loops, we can increase the number of parallel edges by adding

  $$\lambda_p \cdot \left(\sum_{q \in Q} \sum_{q' \in Q} e_{q,q'} \right)$$

  to the objective function. The boolean variable $e_{q,q'}$ is equal to 1 if and only if there is at least one transition

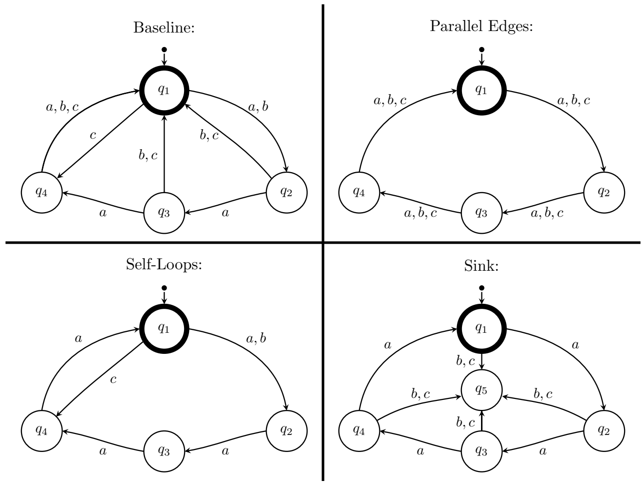

Figure 2: Depiction of four different automatas with different interpretability heuristics based on the same input dataset. Note that, beyond being different to interpret, they also define different models. It is up to the user to decide what type of readability to emphasize as there is no clear best model.

from q to q' and can be computed using the following set of constraints:

$$e_{q,q'} \leq \sum_{a\in\Sigma} \delta_{q,a,q'} \qquad \forall q, q' \in Q$$

$$e_{q,q'} \geq \delta_{q,a,q'} \qquad \forall q, q' \in Q, \forall a \in \Sigma$$

These constraints simply model the boolean function $e_{q,q'} \leftrightarrow \bigvee_{a\in\Sigma} \delta_{q,a,q'}$. Again, $\lambda_p \in \mathbb{R}$ is a weight term to be chosen by the user.

Note that unlike to approaches in machine learning such as feature visualization [Montavon *et al.*, 2019] for a particular input or local model representations [Ribeiro *et al.*, 2016] we aim to give a comprehensive explanation of the entire model. This allows to better understand how a model generalizes and to inject domain expertise by adapting the model.

6 Experimental Evaluation

We implemented a prototype of both learning setups using Python. As a MILP solver, we use the industry-strength Guroibi Optimizer[1] via the Pyomo optimization modeling framework[2] [Bynum *et al.*, 2021; Hart *et al.*, 2011].

We evaluated our setups on a data set generated by Shvo *et al.* [2021] from the ALFRED benchmark set [Shridhar *et al.*, 2020]. The data set contains sequences of step-by-step instructions (action plans), encoded as bit vectors, that achieve a specific goal in the ALFRED setting. For each of the seven goals, we created a training and test set as follows: We combine all plans for that goal with plans for the other goals (i.e., anomalies) in a 9:1 ratio. This set is split into train and test set in a 80:20 ratio. The precise sizes of all train and test sets are displayed in Table 1 together with the share of anomalies, rounded to the next percent points.

	Set Sizes		Bounds	
Goal	Training	Test	Lower	Upper
0	250	62	0.09	0.10
1	352	84	0.10	0.11
2	336	81	0.10	0.11
3	306	75	0.09	0.10
4	310	76	0.09	0.10
5	319	76	0.11	0.12
6	368	88	0.09	0.10

Table 1: Overview of the data sets used for evaluation.

Goal	F1 Score	Time [ms]	# states
0	1.00	70	2
1	0.67	52	2
2	0.46	152	2
3	1.00	110	2
4	1.00	64	2
5	0.77	68	2
6	0.77	123	2

Table 2: Time required, F1 score obtained and automaton size generated by two-bound learning.

On each training set, we learn DFAs using both setups. For single-bound learning, we examined all target automaton sizes between two and ten. As lower and upper acceptance bounds, we used the exact ratio of correct plans to anomalies, rounded to the lower and upper percent values. We then evaluated the performance of the learned DFAs on the test set.

We limiting solving to roughly 100 seconds of CPU time over all threads using Gurobi's `WorkLimit` parameter and ran the experiments were run on an Intel Core i9-7960X CPU.

Two-Bound DFA Learning. For two-bound DFA learning, we examined the time required to build the model and solve it, the F1 score obtained on the training set as well as the number of states in the generated DFA. These are shown in Table 2. We started with a minimum of two states and observed that in every experiment, the first automaton with two states satisfied the constraints and was returned. This shows that for our training sets the anomaly patterns can satisfyingly be detected by automata with just two states. For three sets, a detector with a score of 1 was obtained and only for set 2, the score is less than 0.5. The performance is good, with the algorithm terminating in less than 200 milliseconds for all goals.

Single-Bound DFA Learning. For single-bound DFA learning, we performed a similar evaluation. Since this setup requires a fixed target size, we learned automata for all sizes between two and ten. The results are shown in Figure 3. For most goals, the time required increases with the number of states up to a point, then drops off again. For goal 2, the 9-state automaton learning timed out. In general, the time required by this setup exceeds that of the two-bound one. The number of states' impact on the F1 score is dependent on the goal. For goals 0, 2, and 5, the score remains mostly independent of size, for the others, some sizes cause a marked reduction in

[1] https://www.gurobi.com/solutions/gurobi-optimizer/

[2] https://www.pyomo.org/

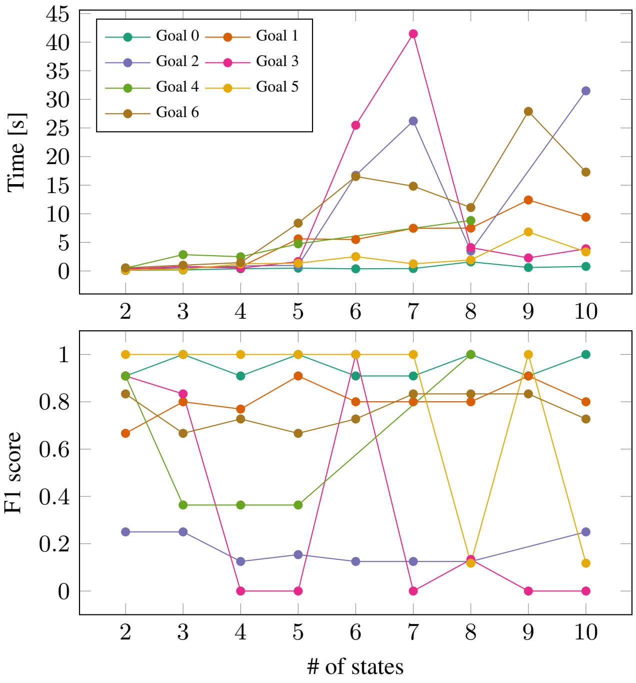

Figure 3: Time required and F1 score achieved by single-bound learning for increasing numbers of states.

score. This indicates that overfitting on the training set may occur for these sizes. A size of 2— the same size returned by two-bound learning—seems to be a good choice for generating anomaly detectors. It requires run times of less than 600 milliseconds and yields good scores for all goals except 2, for which no automaton (including that generated by the two-bound learning) achieves a high score. Going forward, we will focus on size 2 automata.

Loosened Bounds. Finally, we analyzed the impact of loosening the bounds for learning on the F1 score and the runtime. Since in reality, no per-trace ground truth may be available and the share of anomalies can only be estimated, we simulate this effect by increasing the bounds by a value between 0 and 0.05. For single-bound learning, the bound is reduced, for two-bound learning, the lower bound is reduced and the higher increased. No substantial impact on the runtime could be observed for all setups. However, the score is not robust to bound loosening, as can be seen from the selected results in Figure 4. On goal 1, scores are mostly robust, for goal 2, the score improves for loosenings of 0.3 and greater, and for goal 5, the score fluctuates. This phenomena are consistent between learning setups. A possible explanation is that for training sets that contain an easily-recognizable class of correct data that “fits” within the loosened bounds, this class is learned as anomalous instead of truly anomalous data. Therefore, our setups require an accurate estimate of the share of anomalous data to operate correctly.

Summary. Both the two-bound and single-bound learning with size 2 are fast to apply and yield small automata that are nonetheless effective as anomaly detectors in most cases. However, the two-bound setup is slightly faster and yields better detectors for most goals, including goal 2. We conclude that while both learning setups are feasible for learning anomaly detectors, the first is slightly preferrable. However, both setups require precise bound information.

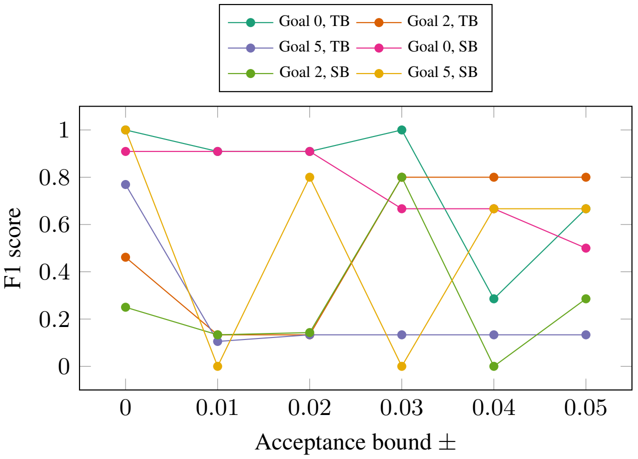

Figure 4: F1 score achieved by two-bound (TB) and single-bound learning ($|Q| = 2$, SB) for increasingly loose bounds on goal sets 0, 2, and 5.

Threats to Validity. The validity of our approach requires anomalies to be detectable by a DFA. This property is guaranteed by the data set, since each strategy can only solve a single ALFRED goal. The internal validity of our results stems from Gurobi’s deterministic behavior, which guarantees that the scoring results can be reproduced as-is. The observed running time showed standard deviations of less than 250 milliseconds and is therefore also reproducible. The external validity is limited by the ALFRED data set we used. While this set has seen widespread use, the results may not fully extend to different types of traces generated by different scenarios.

7 Conclusion

This paper has studied the task of learning inherently interpretable anomaly detectors in the form of DFAs. We have defined two unsupervised learning setups, studied their properties (e.g., their computational complexity), and developed two learning algorithms that utilize off-the-shelf constraint optimization tools. In addition, we have shown how regularization can improve the interpretability of the learned DFAs. Our empirical evaluation has demonstrated that our approach can efficiently generate anomaly detectors for various tasks of the ALFRED data set.

We see various promising directions for future research. First, we intend to relax the requirement to provide bounds on the number of anomalies in the sample. Our idea here is to use a similarity measure, similar to clustering, and learn automata that minimize the similarity between normal data while maximizing the similarity to anomalies. Second, we plan to develop heuristics that sacrifice the optimality of a solution in favor of computational efficiency. Third, we want to extend our approach to more expressive automata classes, such as register automata, to handle data over continuous domains.

References

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

[Biermann and Feldman, 1972] Alan W. Biermann and Jerome A. Feldman. On the synthesis of finite-state machines from samples of their behavior. *IEEE Trans. Computers*, 21(6):592–597, 1972.

[Bynum *et al.*, 2021] Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo — Optimization Modeling in Python*, volume 67 of *Springer Optimization and Its Applications*. Springer International Publishing, Cham, 2021.

[Carmel and Markovitch, 1995] David Carmel and Shaul Markovitch. Opponent modeling in multi-agent systems. In Gerhard Weiß and Sandip Sen, editors, *Adaption and Learning in Multi-Agent Systems, IJCAI'95 Workshop, Montréal, Canada, August 21, 1995, Proceedings*, volume 1042 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 1995.

[Chandola *et al.*, 2009] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[De la Higuera, 2010] Colin De la Higuera. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.

[Garey and Johnson, 1979] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Geiger *et al.*, 2020] Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. TadGAN: Time series anomaly detection using generative adversarial networks. In Xintao Wu, Chris Jermaine, Li Xiong, Xiaohua Hu, Olivera Kotevska, Siyuan Lu, Weija Xu, Srinivas Aluru, Chengxiang Zhai, Eyhab Al-Masri, Zhiyuan Chen, and Jeff Saltz, editors, *2020 IEEE International Conference on Big Data (IEEE BigData 2020), Atlanta, GA, USA, December 10-13, 2020*, pages 33–43. IEEE, 2020.

[Gold, 1978] E. Mark Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978.

[Gurobi Optimization, LLC, 2022] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[Hammerschmidt *et al.*, 2016] Christian Albert Hammerschmidt, Sicco Verwer, Qin Lin, and Radu State. Interpreting finite automata for sequential data. *CoRR*, abs/1611.07100, 2016.

[Hart *et al.*, 2011] William E. Hart, Jean-Paul Watson, and David L. Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3):219, August 2011.

[Heule and Verwer, 2010] Marijn Heule and Sicco Verwer. Exact DFA identification using SAT solvers. In José M. Sempere and Pedro García, editors, *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings*, volume 6339 of *Lecture Notes in Computer Science*, pages 66–79. Springer, 2010.

[Leucker and Neider, 2012] Martin Leucker and Daniel Neider. Learning minimal deterministic automata from inexperienced teachers. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change - 5th International Symposium, ISoLA 2012, Heraklion, Crete, Greece, October 15-18, 2012, Proceedings, Part I*, volume 7609 of *Lecture Notes in Computer Science*, pages 524–538. Springer, 2012.

[Li *et al.*, 2019] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks. In Igor V. Tetko, Vera Kurková, Pavel Karpov, and Fabian J. Theis, editors, *Artificial Neural Networks and Machine Learning - ICANN 2019: Text and Time Series - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part IV*, volume 11730 of *Lecture Notes in Computer Science*, pages 703–716. Springer, 2019.

[Li *et al.*, 2021] Longyuan Li, Junchi Yan, Haiyang Wang, and Yaohui Jin. Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder. *IEEE Trans. Neural Networks Learn. Syst.*, 32(3):1177–1191, 2021.

[Lin *et al.*, 2007] Jessica Lin, Eamonn J. Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, 2007.

[Molnar, 2022] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[Montavon *et al.*, 2019] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: An overview. In Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 193–209. Springer, 2019.

[Neider *et al.*, 2021] Daniel Neider, Jean-Raphaël Gaglione, Ivan Gavran, Ufuk Topcu, Bo Wu, and Zhe Xu. Advice-guided reinforcement learning in a non-markovian environment. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9073–9080. AAAI Press, 2021.

[Neider, 2012] Daniel Neider. Computing minimal separating DFAs and regular invariants using SAT and SMT solvers.

In Supratik Chakraborty and Madhavan Mukund, editors, *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, volume 7561 of *Lecture Notes in Computer Science*, pages 354–369. Springer, 2012.

[Ribeiro *et al.*, 2016] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM, 2016.

[Roy *et al.*, 2020] Rajarshi Roy, Dana Fisman, and Daniel Neider. Learning interpretable models in the property specification language. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2213–2219. ijcai.org, 2020.

[Ruff *et al.*, 2018] Lukas Ruff, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert A. Vandermeulen, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4390–4399. PMLR, 2018.

[Shridhar *et al.*, 2020] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10737–10746, New York, NY, USA, June 2020. IEEE. CVPR 2020.

[Shvo *et al.*, 2021] Maayan Shvo, Andrew C. Li, Rodrigo Toro Icarte, and Sheila A. McIlraith. Interpretable sequence classification via discrete optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):9647–9656, May 2021. AAAI 2021.

[Trakhtenbrot and Barzdin, 1973] Boris A. Trakhtenbrot and Ian. M. Barzdin. Finite automata: behavior and synthesis. 1973.

[Xu *et al.*, 2018] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao. Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 187–196. ACM, 2018.