Signal Temporal Logics for Learning and Detection of Anomalous Behavior

Voltan Gabriele[1]

[1]Department of Mathematics, Computer Science and Physics, University of Udine

Abstract

Anomaly detection is a task that plays an increasingly fundamental role in many fields, from industrial to cybersecurity. This problem is often solved using techniques that are difficult to interpret or require the presence of a domain expert. In this work, three algorithms are introduced for the problem of anomaly learning and anomaly detection, based on a fragment of the signal temporal logic, called iPSTL, which can be used to express continuous system properties, including temporal constraints and physical system parameters. These algorithms, in addition to returning a formula that can be easily interpreted by a human, are independent of the context. To demonstrate the capabilities of the latter, two case studies were reported in two different application domains.

1 Introduction

With the technological development of the last twenty years, information systems have become a fundamental part of many infrastructures, including critical infrastructures, such as industrial or train ones. At the same time as this technological adoption, the need for greater IT security for these systems has also grown, in order to protect them from external attacks. In recent history, several relevant attacks have occurred, such as Stuxnet [6], a malware that hit a nuclear power plant in Iran, WannaCry [8], a ransomware that in just four days knocked out more than 200 thousand computers in 150 different countries, or Mirai [1], a botnet that targeted the Dyn DNS service, consequently attacking many famous services that relied on it. To avoid serious events like these, it is necessary to have systems capable of analyzing the behavior of a system and recognizing any anomalous behavior, caused by faults or external attacks.

In this work, model-free algorithms are presented, capable of learning and detecting anomalous behaviors. These algorithms use formulas from *Signal Temporal Logic* (STL), a specification language that can be used to express continuous system properties, including temporal constraints and physical system parameters. An example property could be "If the train stays on route X maintaining a speed below 50km/h, is it guaranteed to reach station Y within 30 minutes?". As you can see, STL formulas resemble natural language, allowing them to be easily understood by human operators.

Based on the types of data available, we can distinguish two problems: the first is *anomaly learning*, while the second is *anomaly detection*. The first problem belongs to the domain of supervised learning, a domain in which all data (in this case signals) have their own label. This task consists of finding a temporal logic formula that can be used to distinguish normal system behaviors from anomalous ones. The second belongs to the domain of unsupervised learning, a domain in which the data is provided without the corresponding label. In this case, the goal is to find a temporal logic formula that detects anomalous behavior.

However, in continuous systems, the normal behavior may change over time (for example a scenario in which a router is added to a network and an intrusion detection system will have to learn that its traffic is not anomalous) and for this it may be necessary to evolve also knowledge of the analysis tool. For this reason, this work also presents an efficient *online anomaly learning* algorithm, capable of modifying the formula as new data are collected.

The rest of the paper is organized as follows: in section 1.1 a brief overview of the works present in the literature is presented; in section 2 some basic concepts fundamental for understanding the rest of the paper are introduced; in section 3 the Inference Parametric Signal Temporal Logic is introduced, together with its properties; in section 4 the algorithms to solve the problems introduced are proposed; finally, before the conclusions, in section 5, two case studies are presented to test the effectiveness of the proposed algorithms.

1.1 Related works

Most of the recent works on logical inference have focused on estimating the parameters associated with a structure of the temporal logic given in input, that is, given a structure such as "The capacity reaches x liters within τ seconds", find the optimal values for x and τ [2][3][10]. However, this task requires as input a formula structure, which must be specified by a domain expert and which could lead to cases of overfitting or underfitting. To avoid all this, in the proposed procedure the structure of the formula is calculated together with the optimal parameters, exploiting a search based on the robustness degree [4][5], a metric that indicates how much a signal violates or satisfies a formula.

Another task much studied in the literature is that of anomaly detection, i.e. the detection of patterns different from those expected. For cybersecurity, fault detection or video surveillance purposes, many statistical or machine learning tools have been used to solve this problem [9]. The problem with this solutions is that they produce a solution that is difficult to interpret even by industry experts, unlike STL formulas which are easy to understand and interpret.

2 Preliminaries

In this section, some preliminary concepts are introduced, which are fundamental for understanding the rest of the paper.

2.1 Signal

Given a time domain $\mathbb{T} := [0, \infty)$ or a finite prefix, a *continuous-valued signal* is a function $s : \mathbb{T} \to \mathbb{R}^n$. With $s(t)$ we indicate the value of the signal s at time t, while with $s[t]$ we indicate the suffix of the signal s from time t, i.e. $s[t] = \{s(\tau)|\tau \geqslant t\}$. The notation x_s, y_s or v_s indicates the one-dimensional signal corresponding to the variable x, y or v of the signal s.

2.2 Signal Temporal Logic

Signal Temporal Logic (STL) is a temporal logic defined on signals. The syntax of STL is defined as

$$\phi := p|\neg\phi|\phi_1 \wedge \phi_2|\phi_1 \vee \phi_2|\phi_1 U_{[a,b)} \phi_2|F_{[a,b)}\phi|G_{[a,b)}\phi \tag{1}$$

where $p : \mathbb{R}^n \to \{\top, \bot\}$ is a predicate, $U_{[a,b)}$ is the "Until" operator[1], $F_{[a,b)}$ and $G_{[a,b)}$ are the temporal "Finally" ("eventually") and "Globally" ("always") operators, re-

[1]The Until operator is not used in iPSTL, so it will be omitted from now on.

spectively. The semantics of STL can be easily understood and deduced by analyzing the semantics of the iPSTL which will be introduced later.

We call the set of all signals s such that $s \models \phi$ the *language of* ϕ, denoted $L(\phi)$. We say that ϕ_1 and ϕ_2 are *semantically equivalent*, denoted $\phi_1 \equiv \phi_2$, if $L(\phi_1) = L(\phi_2)$.

2.3 System

A system is an object $\mathcal{S}$ that produces observable output signals that are related to the evolution of the internal state. The set of all trajectories $x : \mathbb{T} \to \mathbb{R}^n$ that an object $\mathcal{S}$ can produce is called the *language of* $\mathcal{S}$, denoted as $L(\mathcal{S})$.

Since we are interested in cases in which the system behaves anomalously, we define the *normal behaviors* as the set of signals $L_N(\mathcal{S})$ and the *anomalous behaviors* as the set of signals $L_A(\mathcal{S})$, such that $L_N(\mathcal{S}) \cap L_A(\mathcal{S}) = \emptyset$ and $L_N(\mathcal{S}) \cup L_A(\mathcal{S}) = L(\mathcal{S})$.

3 Inference Parametric Signal Temporal Logic

In this section, Inference Parametric Signal Temporal Logic is introduced, together with its syntax, its semantics, its expressiveness and its properties, which will be useful for the algorithms that will be presented. Furthermore, the concept of distance is introduced, using the Robustness Degree.

Parametric STL is an extension of STL in which the constants involved in predicates and time intervals are replaced with free parameters. Assigning real values to the parameters of a PSTL formula returns an STL formula. In this work we use a fragment of PSTL called Inference PSTL (iPSTL).

3.1 Syntax of iPSTL

The syntax of iPSTL is defined as

$$\begin{aligned} \varphi &::= F_{[\tau_1,\tau_2)}\varphi_i \\ \varphi_i &::= F_{[\tau_1,\tau_2)}p | G_{[\tau_1,\tau_2)}p | \varphi_i \vee \varphi_i | \varphi_i \wedge \varphi_i \end{aligned} \tag{2}$$

where p is a linear predicate of the form $(y_s \leq \pi)$ or $(y_s > \pi)$, where y_s is a coordinate of the signal s, π is a scale parameter, τ_1 and τ_2 are time parameters. $F_{[\tau_1,\tau_2)}$ and $G_{[\tau_1,\tau_2)}$ are the temporal "Finally" ("eventually") and "Globally" ("always") operators, respectively. The first formula of eq. (2) can be read as "At some time t in the time interval $[\tau_1, \tau_2)$, an event described by φ_i occurs".

3.2 Semantics of iPSTL

The semantics of iPSTL is recursively defined as

$$\begin{gathered} s[t] \models (y_s \sim \pi) \Leftrightarrow y_s(t) \sim \pi \\ s[t] \models \phi_1 \vee \phi_2 \Leftrightarrow s[t] \models \phi_1 \vee s[t] \models \phi_2 \\ s[t] \models \phi_1 \wedge \phi_2 \Leftrightarrow s[t] \models \phi_1 \wedge s[t] \models \phi_2 \\ s[t] \models G_{[\tau_1,\tau_2)}(y_s \sim \pi) \Leftrightarrow y_s(t^{‘}) \sim \pi \ \forall t^{‘} \in [t+\tau_1, t+\tau_2) \\ s[t] \models F_{[\tau_1,\tau_2)}(y_s \sim \pi) \Leftrightarrow \exists t^{‘} \in [t+\tau_1, t+\tau_2) \text{ s.t. } y_s(t^{‘}) \sim \pi \end{gathered}$$

where $\sim \in \{\leq, >\}$.

3.3 Expressivity

iPSTL, like STL in general, can be used to express a very wide range of properties of continuous systems. Two types of properties are:

- **Bounded-time invariance**, for example "There exists a time $t \in [0, \tau_1)$ such that y_s will always be greater than π in the time interval $[t + \tau_2, t + \tau_3)$"
- **Reachability to multiple regions**, for example "There exists a time $t \in [0, \tau_1)$ in which y_s is either greater than π_1 or less than π_2 in the time interval $[t + \tau_2, t + \tau_3)$"

3.4 Distance and Robustness Degree

In this subsection the concept of distance and robustness degree is introduced.
A *signed distance* from a signal $s : \mathbb{T} \to \mathbb{R}^n$ to a set $S \subseteq \mathcal{F}(\mathbb{T}, \mathbb{R}^n)$ is defined as

$$D_\rho(s, S) := \begin{cases} -\inf\{\rho(s, s^{`})|s^{`} \in cl(S)\} & \text{if } s \notin S \\ \inf\{\rho(s, s^{`})|s^{`} \in \mathcal{F}(\mathbb{T}, \mathbb{R}^n)\} & \text{if } s \in S \end{cases}$$

where $cl(S)$ denotes the closure of S, ρ is a metric defined as

$$\rho(s, s^{`}) = \sup_{t \in T}\{d(s(t), s(t^{`}))\}$$

where d corresponds to the metric defined on the domain $\mathbb{R}^n$ of signal s.

The *robustness degree* $r(s, \phi, t)$ of a signal s with respect to an STL formula ϕ at time t is a metric that indicates whether and by how much the signal satisfies the formula. In fact, if the sign of $r(s, \phi)$ is negative it means that s does not satisfy ϕ, if positive it does. Furthermore, the magnitude of $r(s, \phi)$ gives a measure of how different s would have to be to satisfy the formula ϕ. The robustness degree can be calculated as:

$$\begin{aligned}
r(s, (y_s \geq c_1), t) &= y_s(t) - c_1 \\
r(s, (y_s < c_1), t) &= c_1 - y_s(t) \\
r(s, \phi_1 \vee \phi_2, t) &= \max(r(s, \phi_1, t), r(s, \phi_2, t)) \\
r(s, \phi_1 \wedge \phi_2, t) &= \min(r(s, \phi_1, t), r(s, \phi_2, t)) \\
r(s, G_{[c_1,c_2)}\phi, t) &= \min_{t^{`} \in [t+c_1, t+c_2)} r(s, \phi, t^{`}) \\
r(s, F_{[c_1,c_2)}\phi, t) &= \max_{t^{`} \in [t+c_1, t+c_2)} r(s, \phi, t^{`})
\end{aligned}$$

where c_i are real values. Furthermore, it is important to underline that, as demonstrated in [5], the robustness degree $r(s, \phi)$ is an under-approximation of its corresponding signed distance $D(s, \phi)$.

3.5 Properties of iPSTL: partial order and DAG

In this subsection some properties related to the iPSTL are described, which will then be useful for the construction of the algorithms.

3.5.1 Partial orders over iSTL and iPSTL

Definition 1. *For two iSTL formulae ϕ_1 and ϕ_2, $\phi_1 \preceq_S \phi_2$ iff $\forall s \in \mathcal{F}(\mathbb{T}, \mathbb{R}^n)$, $s \models \phi_1 \Rightarrow s \models \phi_2$, i.e. $L(\phi_1) \subseteq L(\phi_2)$.*

Definition 2. *For two iPSTL formulae φ_1 and φ_2, $\varphi_1 \preceq_P \varphi_2$ iff $\forall \theta$, $\phi_{1,\theta} \preceq_S \phi_{2,\theta}$, where the domain of θ is the union of parameters appearing in φ_1 and φ_2.*

Based on these definitions and the semantics of iSTL and iPSTL, we have two propositions.

Proposition 1. *Both $\preceq_S$ and $\preceq_P$ are partial orders.*

Proof. A partial order $\preceq$ is a binary relation that is reflexive, transitive and antisymmetric.

($\preceq_S$) *Reflexive* $\phi_1 \preceq_S \phi_1$ is equivalent to $L(\phi_1) \subseteq L(\phi_1)$, which is trivially true. *Transitivity* $\phi_1 \preceq_S \phi_2$ and $\phi_2 \preceq_S \phi_3$ is equivalent to $L(\phi_1) \subseteq L(\phi_2)$ and $L(\phi_2) \subseteq L(\phi_3)$. It implies $L(\phi_1) \subseteq L(\phi_3)$, which means $\phi_1 \preceq_S \phi_3$. *Antisymmetry* $\phi_1 \preceq_S \phi_2$ and $\phi_2 \preceq_S \phi_1$ is equivalent to $L(\phi_1) \subseteq L(\phi_2)$ and $L(\phi_2) \subseteq L(\phi_1)$. It implies $L(\phi_1) = L(\phi_2)$, which means $\phi_1 \equiv \phi_2$.

($\preceq_P$) See Appendix A of [7]. □

Proposition 2. *The partial order $\preceq_P$ satisfies the following properties:*

1. $\varphi_1 \wedge \varphi_2 \preceq_P \varphi_j \preceq_P \varphi_1 \vee \varphi_2$ *for* $j = 1, 2$
2. $G_{[\tau_1,\tau_2)} p \preceq_P F_{[\tau_1,\tau_2)} p$ *where p is a linear predicate.*

The first property is an extension of the propositional logic rules $A \wedge B \Rightarrow A \Rightarrow A \vee B$. The second property states "If a property is always true over a time interval, then it is trivially true at some point in that interval".

3.5.2 DAG and Signed Distance

The structure of iPSTL and the definition of the partial order $\preceq_P$ enable the definition of the following theorem.

Theorem 1. *The formulae in iPSTL have an equivalent representation as nodes in an infinite DAG. A path exists from formula φ_1 to φ_2 iff $\varphi_1 \preceq_P \varphi_2$. The DAG has a unique top element ($\top$) and a unique bottom element ($\bot$).*

Proof. See Appendix B of [7]. □

Next, we establish a relationship between the signed distance of a signal s with respect to iSTL formula ϕ and the partial order $\preceq_S$.

Theorem 2. *The following statements are equivalent:*

1. $\phi_1 \preceq_S \phi_2$
2. $\forall s \in \mathcal{F}(\mathbb{T}, \mathbb{R}^n)$, $D(s, \phi_1) \leq D(s, \phi_2)$.

Proof. See Appendix C of [7]. □

Corollary 1. *The following statements are equivalent:*

1. $\varphi_1 \preceq_P \varphi_2$
2. $\forall s \in \mathcal{F}(\mathbb{T}, \mathbb{R}^n)$, $\forall \theta$ $D(s, \phi_{1,\theta}) \leq D(s, \phi_{2,\theta})$.

4 Proposed algorithms

In this section, the algorithms for solving the three problems, namely *Anomaly Learning*, *Anomaly Detection* and *Online Anomaly Learning*, introduced in section 1 of the paper, are proposed.

4.1 Learning as Optimization with Robustness Degree

The three problems described are all three problems that can be traced back to a binary classification (*normal/anomalous*). To measure performance in this type of task, a metric called *Missclassification Rate* is usually used and is calculated as

$$MR(\{(s_i, p_i)\}_{i=1}^{M}, \phi_N) = \frac{FA + MD}{M}$$

where ϕ_N is the formula that should recognize normal behaviors, FA $= |\{s_i | s_i \not\models \phi, p_i = 1\}|$ is the number of false alarms and $MD = |\{s_i | s_i \models \phi, p_i = -1\}|$ is the number of missed detections.

However, the problem with this rate is that it ignores the degree of the error made. One solution could be to use the signed distance $D(s, \phi)$, but this cannot be calculated or represented analytically for many real systems [5]. The solution therefore is to use the *robusteness degree* $r(s, \phi)$, which is an approximation of the signed distance.

Then, according to Theorem 1 and Corollary 1, the optimization problem can be solved by combining a discrete search over a DAG to find an iPSTL formula φ with a continuous search to find its appropriate parameterization θ.

For the search, in offline problems, the *Simulated Annealing* heuristic is used, while for the online one, the *Stochastic Gradient Descent* is used.

4.2 Offline Anomaly Learning

Problem Let $\{x_i\}_{i=1}^{M}$ be a set of trajectories generated by a system $\mathcal{S}$, s_i the corresponding observed output signal and p_i the respective label ($p_i = 1$ normal behavior, $p_i = -1$ anomalous behavior). Find an iSTL formula ϕ_{N,θ_N}, which describes normal behavior and such that the iPSTL formula φ_N and the evaluation θ_N minimize

$$J_\alpha(\varphi, \theta) = \frac{1}{M} \sum_{i=1}^{M} l(p_i, r(s_i, \phi_\theta)) + \lambda \|\phi_\theta\| \qquad (3)$$

where r is the robustness degree, ϕ_θ is derived from φ with valuation θ, λ is a weighting parameter, $\|\phi_\theta\|$ is the length of ϕ_θ and l is a *loss function*, calculated as

$$l(p_i, r(s_i, \phi_\theta)) = \max(0, \epsilon_r - p_i r(s_i, \phi_\theta)) \qquad (4)$$

where $\epsilon \ll 1$.

As we can see from eq. (3) the length of the formula is penalized because if ϕ_{N,θ_N} grows too much it becomes as complex to represent as the data itself, making the procedure useless.

Algorithm The procedure begins by initializing the DAG with *basic nodes*, i.e. linear predicates of the form $O_{[\tau_1,\tau_2)}(x_s \sim \pi_1)$, where $O \in \{G, F\}$, $\sim \in \{\geq, <\}$ and $x_s \in V$, where V is the set of variables represented in the system output signal. Edges are constructed from φ_j to φ_k in the initial graph iff $\varphi_j \preceq_P \varphi_k$. ListInitialization($\mathcal{G}_1$) generates a ranked list of formulae from the basis node.

After building the graph, for each node, ParameterEstimation($\{(s_i, p_i)\}_{i=1}^{M}, \varphi$) uses simulated annealing to find the optimal evaluation for φ by minimizing the cost J_α.

PruningAndGrowing($\mathcal{G}_{i-1}$) first applies the principle of Feature Subset Selection (FSS) by eliminating a fixed number of nodes that do not fit the observed data. Then, if the missed detection rate of $\phi_{j,\theta}$ is higher than the false alarm rate, the function adds to the graph a formula $\varphi^{'} = \varphi_j \vee \varphi_b$, where $\varphi_b \in$ Basis is a basis formula with good performance. Instead, if the false alarm rate is higher, the function would add $\varphi^{''} = \varphi_j \wedge \varphi_b$ to the graph.

Algorithm 1 Anomaly Learning

```
  Input:
  A set of labeled signal $\{(s_i, p_i\}_{i=1}^M$
  A variable set $V$
  A missclassification rate threshold $\delta$
  A formula length bound $W$
  Output:
  An iPSTL formula $\varphi$ and valuation $\theta$
for $i = 1$ to W do
  if $i = 1$ then
    $\mathcal{G}_1 \leftarrow DAGInitialitazion(V)$
    $List \leftarrow ListInitialization(\mathcal{G}_1)$
  else
    $\mathcal{G}_i \leftarrow PruningAndGrowing(\mathcal{G}_{i-1})$
    $List \leftarrow Ranking(\mathcal{G}_i \backslash \mathcal{G}_{i-1})$
  while $List \neq \emptyset$ do
    $\varphi \leftarrow List.pop()$
    $(\theta, MR) \leftarrow ParameterEstimation(\{(s_i, p_i)\}_{i=1}^M, \varphi)$
    if $MR \leq \delta$ then
      return $(\varphi, \theta)$
return $MinimumCostNode(\mathcal{G}_W)$
```

Ranking($\mathcal{G}_i \backslash \mathcal{G}_{i-1}$) ranks the newly grown nodes based on a heuristic function

$$\frac{1}{|pa(k_i)|} \sum_{k_{i-1} \in pa(k_i)} J_a(k_{i-1}) \tag{5}$$

where k_i is a node in $\mathcal{G}_i$, $pa(k_i)$ is the set of k_i's parents and $|pa(k_i)|$ is the size of $pa(k_i)$.

The iterative graph growing and parameter estimation procedure is performed until a formula with low enough misclassification rate is found or W iterations are completed. At this point, MinimumCostNode($\mathcal{G}_i$) returns the node with the minimum cost within $\mathcal{G}_i$.

4.3 Anomaly Detection

The *anomaly detection* problem requires dividing anomalous data from normal data, without knowing their labels. This problem can be solved with a *one-class Support Vector Machine* (SVM).

Problem Find an iPSTL formula ϕ_{N,θ_N}, such that the formula φ_N and the evaluation θ_N minimize

$$\min_{\phi_\theta, \epsilon} d(\phi_\theta) + \frac{1}{\nu N} \sum_{i=1}^{N} \mu_i - \epsilon \tag{6}$$

such that

$$\mu_i := \begin{cases} 0 & \text{if } r(s_i, \phi_\theta) > \frac{\epsilon}{2} \\ \frac{\epsilon}{2} - r(s_i, \phi_\theta) & \text{else} \end{cases} \tag{7}$$

where ϵ is the "gap" in signal space between outputs identified as normal and outputs identified as anomalous, ν is the upper bound of the a priori probability that a signal $x_i \in L_A(\mathcal{S})$, and μ_i is a slack variable which is positive if s_i does not satisfy ϕ_θ with minimum robustness $\epsilon/2$. The function d is a "tightness" function that penalizes the size of $L(\phi_\theta)$.

By maximazing the gap ϵ, optimization attemps to maximize the separation between normal and anomalous outputs. Instead, by minimizing the function $d(\phi_\theta)$, optimization prevents the learned formula finding a formula such that $L(\phi) = L(\mathcal{S})$, which would render the optimization redundant.

Algorithm The algorithm proposed for the offline anomaly learning problem can be adapted to solve the problem 6, with two simple modifications: the input signals are not labeled, i.e. the inputs are $\{s_i\}_{i=1}^M$; the ParameterEstimation function resolves 6 instead of 3.

4.4 Online Anomaly Learning

Problem *Online anomaly learning* has the goal of, given a series of outputs s_i with the respective label p_i, keeping a formula ϕ_N^t updated such that the misclassification rate, defined as for the problem of offline anomaly learning, is minimized. Each time a new pair (s_{t+1}, p_{t+1}) becomes available, the algorithm will have to use the ϕ_N^t and the new pair to produce an updated formula ϕ_N^{t+1}.

For a fixed iPSTL formula structure φ, *stochastic gradient descent* can find its optimal parameterization θ^* if there exists a ϕ_{θ^*} with structure φ that can classify the data. Let θ_i be the parameterization of φ after i observed pairs of signals and labels. The stochastic gradient descent that minimizes the loss function l is given by

$$\theta_{i+1} := \begin{cases} \theta_i & \text{if } p_i r(s_i, \phi_{\theta_i}) \geq \epsilon_r \\ \theta_i + \eta \frac{\partial r}{\partial \theta} p_i & \text{otherwise} \end{cases} \tag{8}$$

where $\eta > 0$ is the learning rate and the partial derivative is calculated according to the centered first difference, a technique for approximating the derivative. If the structure φ is the correct iPSTL formula for the classification, it should be expected that there exist a step $\bar{i}$ such that $p_i r(s_i, \phi_{\theta_i}) \geq 0$ for all $i \geq \bar{i}$, and therefore the misclassification rate converges towards 0. Instead, if φ is not the correct formula, then the improvement on classification performance saturates at a certain step $\bar{i}$ and a new formula should be calculated.

This procedure operates on a collection of N_f, where N_f is at least as large as the size of the basis, candidate formulae $\{\varphi_j, \theta_j\}_{j=1}^{N_f}$, instead of considering a single formula at a time because there is initially very little information about what kinds of behaviours the system may satisfy.

Algorithm The procedure starts initializing the set of formulae to N_f formulae from the basis and corresponding initial valuation guesses that are found via simulated annealing. Then, when a new signal and label pair (s_i, p_i) becomes available, the function ParameterUpdate is called to update each value θ_j in the formula database using the stochastic gradient descent. Every $checkInt$ signals, the misclassification rates of each φ_{j,θ_j}, with respect to the available traces, are evaluated and the formula database is the populated with new formulae. The algorithm returns the best formula that is the candidate formula of the database that minimizes the missclassification rate, between the two best candidate formulas.

If the missed detection rates are greater then the false alarm rates, i.e. the size of the languages of the formulae are too large, then the conjunction of the two formulae is added to the formula database. Otherwise, the disjunction of the two is added. The function $getValuation$ maps the two valuation θ_b and θ_{sb} to the corresponding θ_{new} of the simplified combined formula.

Note that, in the algorithm, the learning rate η starts from η_{max} and decrease in a geometric fashion with rate $0 \ll \alpha < 1$ until it reaches a level η_{min}. This trick, also called *learning rate decay*, allows the $ParameterUpdate$ function to make bigger steps whenever we initially know very little about the optimal parameterizations for each structure and make smaller steps after more information has been collected.

Algorithm 2 Online Anomaly Learning

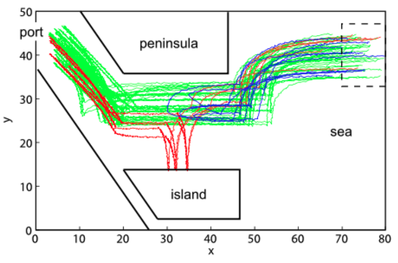
```
  Input:
  A sequence of labeled signal $\{(s_i, p_i\}_{i=1}^{M}$
  Database of candidate STL classifiers formulae
  Maximum and minimum learning rates $\eta_{max}$, $\eta_{min}$
  Geometric rate $\alpha$
  Number of iterations before updating formulae database $checkInt$
  Maximum number of iterations $numIters$
  Output:
  An iPSTL formula $\varphi$ and valuation $\theta$
for $\varphi_k \in formulae$ do
    $\theta_k \leftarrow ParameterEstimation((s_i, p_i), \varphi_k)$
for $i = 1, ..., numIters$ do
    $traces \leftarrow UpdateTraces(traces, s_i, p_i)$
    for $(\varphi_k, \theta_k) \in formulae$ do
        $\theta_k \leftarrow ParameterUpdate((s_i, p_i), \varphi_k, \theta_k)$
    $\eta \leftarrow \max(\alpha\eta, \eta_{min})$
    if $(i \bmod checkInt) == 0$ then
        $formulae \leftarrow UpdateFormulae(formulae)$
    $\eta_{min} \leftarrow \eta$
return $bestFormula(formulae, traces)$
```

5 Case studies and Experiments

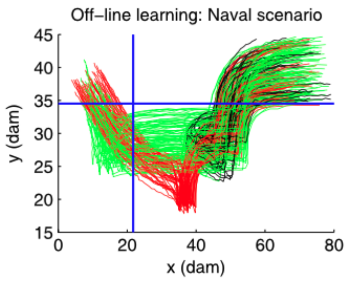
This section reports two experiments conducted with the proposed algorithms, in two different application scenarios.

5.1 Naval Surveillance

In maritime surveillance, the Automatic Identification System (AIS) is used, which is a system capable of collecting information, such as speed, position, direction, of many boats. In this case study we consider a scenario where we want to detect terrorist attacks or human trafficking near a naval port.



(a) A naval surveillance example



(b) Results of offline inference

The *normal behavior* $L_N(\mathcal{S})$ (green lines) requires that the boat arriving from the sea approaches the closest crossing point between the peninsula and the island and then enters the port.

There are two *anomalous behaviors* $L_A(\mathcal{S})$ considered in this case study. The first, i.e. the case of a terrorist attack (blue lines), involves the boat approaching a ferry and then returning to the open sea. The second, i.e. the case of human trafficking (red lines), involves the boat deviating towards the island before entering the port. For a clearer understanding look at fig. 1a.

Algorithm 3 UpdateFormulae

```
    Input:
    Trace databse $tr$
    Formula databse $f$
    Output:
    Updated database $f$
for $k = 1$ to $N_f$ do
    $(\varphi_b, \theta_b) \leftarrow bestFormula(f, tr)$
    $uf1 \leftarrow uf1 \cup \{(\varphi_b, \theta_b)\}$
    $(mdb, fab) \leftarrow calculateRates(\varphi_b, \theta_b, tr)$
    for $m = 1$ to $N_f - k$ do
        $(\varphi_{sb}, \theta_{sb}) \leftarrow bestFormula(f, tr \backslash (uf1 \cup uf2), tr)$
        $(mdsb, fasb) \leftarrow calculateRates(\varphi_{sb}, \theta_{sb}, tr)$
        if $(mdsb + mdb > fab + fasb)$ then
            $\varphi_{new} \leftarrow simplify(\varphi_b \wedge \varphi_{sb})$
        else
            $\varphi_{new} \leftarrow simplify(\varphi_b \vee \varphi_{sb})$
        $\theta_{new} \leftarrow getValuation(\varphi_{new}, \theta_b, \theta_{sb})$
        $f \leftarrow f \cup \{(\varphi_{new}, \theta_{new})\}$
        $uf2 \leftarrow uf2 \cup (\varphi_{sb}, \theta_{sb})$
return $f$
```

Each boat is modeled as a Dubins' vehicle:

$$\begin{cases} x = v\cos\alpha \\ y = v\sin\alpha \\ \alpha = \omega \end{cases} \tag{9}$$

where x and y are the boat's coordinates, v is the constant speed, α is its heading and ω is its angular velocity.

Anomaly Learning For this test, 50 normal tracks, 25 tracks representing human trafficking and 25 tracks representing terrorist attacks were used. Using the algorithm 1 with $n = 15$ and $m = 15$ the following formula was obtained

$$\phi_N = F_{[0,320)}(G_{[28,227)}y_s > 21.73) \wedge (G_{[308,313)}x_s < 34.51) \tag{10}$$

with a total misclassification rate 0.095. Figure 1b shows some examples of traces and the thresholds of the formula 10 are represented in blue.

Online Anomaly Learning For this test, a larger set of signal was used. At each iteration of the algorithm 2, we drew a signal uniformly at random from a set of 2000 trajectories, which consisted of 1000 normal trajectories, 500 human trafficking trajectories, and 500 terrorist trajectories. The *learning rate* parameters were $\alpha = 0.995$, $\eta_{max} = 0.2$ and $\eta_{min} = 0.01$.

Figure 2 shows the misclassification rate of the inferred formula at time i with respect to all 2000 traces. The formula that was inferred after all 2000 trajectories were used is

$$\phi_N = F_{[0,320)}(G_{[174,228)}y_s > 19.88) \wedge (G_{[92,297)}x_s < 34.08) \tag{11}$$

with a total misclassification rate 0.0885.

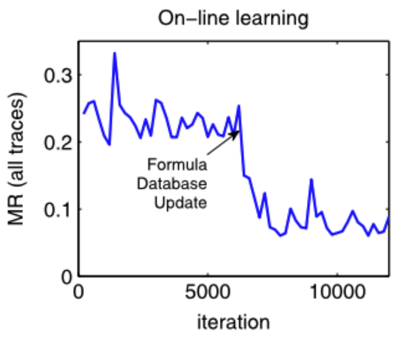


Figure 2: The misclassification rate over time for online learning

5.2 Train Network Monitoring

Consider a train using an electronically-controlled pneumatic (ECP) braking system. The train has 3 cars, each of which has its own braking system. The braking system is automated to regulate the velocity v below unsafe speeds and above low speeds to ensure that the train reaches its destination, as shown in the top-left sub-figure of fig. 3. However, one possible attack is to disable the brake system, making the speed unregular.

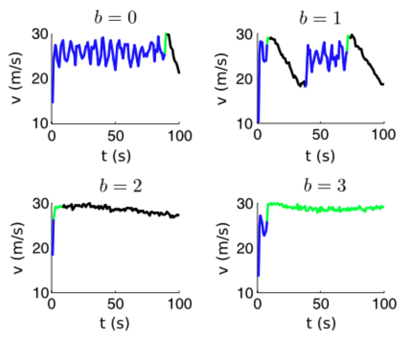


Figure 3: Outputs of the train velocity system under normal (upper-left) and attack scenarios

Anomaly Detection In this test 43 normal trajectories and 7 attack trajectories were used. The algorithm proposed in section 4.3 inferred the formula

$$\phi = F_{[0,100)}(F_{[10,69)}(v_s < 24.9) \wedge F_{[13.9,44.2)}(v_s > 17.66)) \tag{12}$$

This formula is consistent with the observed un-attacked signals (shown in top left of fig. 3), as the properly functioning brake system forces the velocity to be below 24.9 m/s regularly while ensuring that the speed never deviates too far below some desired minimum speed. In contrast, when under attack (fig. 3, bottom right), this regulation never occurs. and the velocity is allowed to remain about 24.9 m/s indefinitely. The formula 12 perfectly separates the data, i.e., the misclassification rate is 0. The formula was inferred using 15 simulated annealing cycles with 15 sample points per cycle.

6 Conclusion

This paper explored the world of anomaly detection and learning, a particularly complex world often solved with machine and deep learning techniques. However, this work, after introducing Inference Parametric Signal Temporal Logic (iPSTL), shows three domain-independent algorithms for anomaly detection and anomaly learning, both offline and online. The latter represents an important challenge, especially in contexts such as that of controlling the traffic of a computer network, where behavior can evolve over time.

Finally, to demonstrate the abilities of these algorithms, they were tested in two different application contexts, namely that of maritime surveillance and that of monitoring train networks, obtaining in both scenarios a Missclassification Rate of less than 0.1.

References

[1] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. *26th USENIX security symposium (USENIX Security 17)*, pages 1093–1110, 2017.

[2] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. *Runtime Verification: Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers 2*, pages 147–160, 2012.

[3] Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. On the robustness of temporal properties for stochastic models. *arXiv preprint arXiv:1309.0866*, 2013.

[4] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 92–106, 2010.

[5] Georgios E Fainekos and George J Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[6] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, symantec corp., security response*, 5(6):29, 2011.

[7] Zhaodan Kong, Austin Jones, and Calin Belta. Temporal logics for learning and detection of anomalous behavior. *IEEE Transactions on Automatic Control*, 62(3):1210–1222, 2016.

[8] Savita Mohurle and Manisha Patil. A brief study of wannacry threat: Ransomware attack 2017. *International journal of advanced research in computer science*, 8(5):1938–1940, 2017.

[9] Gabriele Voltan, Gian Luca Foresti, and Marino Miculan. Pairing an autoencoder and a sf-soinn for implementing an intrusion detection system. *Proceeding of Ital-IA 2023*, page 25, 2023.

[10] Hengyi Yang, Bardh Hoxha, and Georgios Fainekos. Querying parametric temporal logic properties on embedded systems. *Testing Software and Systems: 24th IFIP WG 6.1 International Conference, ICTSS 2012, Aalborg, Denmark, November 19-21, 2012. Proceedings 24*, pages 136–151, 2012.