Temporal Logics for Learning and Detection of Anomalous Behavior

Gabriele Voltan[1]

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

20/12/2023

Outline

Introduction

Introduction to the problem

Systems are increasingly complex and handle more and more data

Introduction to the problem

Systems are increasingly complex and handle more and more data

Goal: detect anomalous behavior caused by internal errors or external attacks

Introduction to the problem

Systems are increasingly complex and handle more and more data

Goal: detect anomalous behavior caused by internal errors or external attacks

- *Stuxnet*: malware that hit a nuclear power plant in Iran [6]

Introduction to the problem

Systems are increasingly complex and handle more and more data

Goal: detect anomalous behavior caused by internal errors or external attacks

- *Stuxnet*: malware that hit a nuclear power plant in Iran [6]
- *WannaCry*: ransomware that in just four days knocked out more than 200 thousand computers in 150 different countries [8]

Introduction to the problem

Systems are increasingly complex and handle more and more data

Goal: detect anomalous behavior caused by internal errors or external attacks

- *Stuxnet*: malware that hit a nuclear power plant in Iran [6]
- *WannaCry*: ransomware that in just four days knocked out more than 200 thousand computers in 150 different countries [8]
- *Mirai*: a botnet that targeted the Dyn DNS service [1]

What task do we want to solve?

Two different types of tasks:

- **Anomaly Learning**: find a temporal logic formula that can be used to distinguish normal system behaviors from anomalous ones (*Supervised learning*)

What task do we want to solve?

Two different types of tasks:

- **Anomaly Learning**: find a temporal logic formula that can be used to distinguish normal system behaviors from anomalous ones (*Supervised learning*)
- **Anomaly Detection**: find a temporal logic formula that detects anomalous behavior (*Unsupervised learning*)

What task do we want to solve?

Two different types of tasks:

- **Anomaly Learning**: find a temporal logic formula that can be used to distinguish normal system behaviors from anomalous ones (*Supervised learning*)
- **Anomaly Detection**: find a temporal logic formula that detects anomalous behavior (*Unsupervised learning*)

Problem: in continuous systems, behavior can evolve over time

What task do we want to solve?

Two different types of tasks:

- **Anomaly Learning**: find a temporal logic formula that can be used to distinguish normal system behaviors from anomalous ones (*Supervised learning*)
- **Anomaly Detection**: find a temporal logic formula that detects anomalous behavior (*Unsupervised learning*)

Problem: in continuous systems, behavior can evolve over time

- **Online Anomaly Learning**: find a *new* temporal logic formula that can be used to distinguish normal system behaviors from anomalous ones, whenever new data is available (*Supervised learning*)

Advantages of this work

Most of the works present in the literature for this type of problem are:

- Estimation of the parameters associated with an STL structure given as input [2][3][10]

Advantages of this work

Most of the works present in the literature for this type of problem are:

- Estimation of the parameters associated with an STL structure given as input [2][3][10]
  - Problems: the structure must be provided as input by a domain expert, the structure could lead to cases of underfitting or overfitting

Advantages of this work

Most of the works present in the literature for this type of problem are:

- Estimation of the parameters associated with an STL structure given as input [2][3][10]
  - Problems: the structure must be provided as input by a domain expert, the structure could lead to cases of underfitting or overfitting
- Machine or deep learning models [9]

Advantages of this work

Most of the works present in the literature for this type of problem are:

- Estimation of the parameters associated with an STL structure given as input [2][3][10]
  - Problems: the structure must be provided as input by a domain expert, the structure could lead to cases of underfitting or overfitting
- Machine or deep learning models [9]
  - Problems: they produce a solution that is difficult to interpret even by industry experts

Advantages of this work

Most of the works present in the literature for this type of problem are:

- Estimation of the parameters associated with an STL structure given as input [2][3][10]
    - Problems: the structure must be provided as input by a domain expert, the structure could lead to cases of underfitting or overfitting
- Machine or deep learning models [9]
    - Problems: they produce a solution that is difficult to interpret even by industry experts

This work:

- the structure of the formula is calculated together with the optimal parameters, exploiting a search based on the robustness degree [4][5]

Advantages of this work

Most of the works present in the literature for this type of problem are:

- Estimation of the parameters associated with an STL structure given as input [2][3][10]
  - Problems: the structure must be provided as input by a domain expert, the structure could lead to cases of underfitting or overfitting
- Machine or deep learning models [9]
  - Problems: they produce a solution that is difficult to interpret even by industry experts

This work:

- the structure of the formula is calculated together with the optimal parameters, exploiting a search based on the robustness degree [4][5]
- STL formulas are easy to understand and interpret

Preliminaries

Signal

Definition 1
Given a time domain $\mathbb{T} := [0, \infty)$, or a finite prefix, a *continuous-valued signal* is a function $s : \mathbb{T} \to \mathbb{R}^n$.

Notation:

- 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. $s[t] = \{s(\tau) | \tau \geqslant t\}$.
- x_s, y_s or v_s indicates the one-dimensional signal corresponding to the variable x, y or v of the signal s

Signal Temporal Logic

Definition 2

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

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

The *language of* ϕ, denoted $L(\phi)$, is the set of all signals s such that $s \models \phi$

Signal Temporal Logic

Definition 2
Signal Temporal Logic (STL) is a temporal logic defined on signals.
The syntax of STL is defined as

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

The *language of* ϕ, denoted $L(\phi)$, is the set of all signals s such that $s \models \phi$

Definition 3
ϕ_1 and ϕ_2 are *semantically equivalent*, denoted $\phi_1 \equiv \phi_2$, if $L(\phi_1) = L(\phi_2)$.

System

Definition 4
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 $L(\mathcal{S})$

System

Definition 4

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 $L(\mathcal{S})$

Definition 5

Normal behaviors are the set of signals $L_N(\mathcal{S})$ and the *anomalous behaviors* are the set of signals $L_A(\mathcal{S})$, such that $L_N(\mathcal{S}) \cap L_A(\mathcal{S}) = \emptyset$ and $L_N(\mathcal{S}) \cup L_A(\mathcal{S}) = L(\mathcal{S})$.

Inference Parametric Signal Temporal Logic

Syntax of iPSTL

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

Syntax of iPSTL

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

Definition 6

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 $(y_s > \pi)$.

Sematics of iPSTL

Definition 7
The semantics of iPSTL is recursively defined as

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

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

Signed distance and Robustness Degree I

Definition 8

A *signed distance* from a signal $s : \mathbb{T} \rightarrow \mathbb{R}^n$ 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.

Notation:

- inf/sup of a subset S is the *greatest/least* element that is *less/greater* than or equal to each element of S

Signed distance and Robustness Degree II

Definition 9

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. It 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.

Meaning of the Robustness Degree

How to interpret the values of $r(s, \varphi, t)$?

Meaning of the Robustness Degree

How to interpret the values of $r(s, \varphi, t)$?

- If the sign of $r(s, \phi)$ is *negative* it means that s does not satisfy ϕ
- If the sign of $r(s, \phi)$ is *positive* it means that s satisfy ϕ

Meaning of the Robustness Degree

How to interpret the values of $r(s, \varphi, t)$?

- If the sign of $r(s, \phi)$ is *negative* it means that s does not satisfy ϕ
- If the sign of $r(s, \phi)$ is *positive* it means that s satisfy ϕ

And what about the magnitude of $r(s, \varphi, t)$?

Meaning of the Robustness Degree

How to interpret the values of $r(s, \varphi, t)$?

- If the sign of $r(s, \phi)$ is *negative* it means that s does not satisfy ϕ
- If the sign of $r(s, \phi)$ is *positive* it means that s satisfy ϕ

And what about the magnitude of $r(s, \varphi, t)$?

The magnitude of $r(s, \phi)$ gives a measure of how different s would have to be in order to satisfy the formula ϕ

Properties of iPSTL: partial orders I

Definition 10
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 11
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.

Properties of iPSTL: partial orders II

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 $\phi_1 \equiv \phi_2$.

($\preceq_P$) The idea is that, thanks to the independence of the assignments of each parameter, the evaluation of the parameters of the formula can be decomposed into the evaluation of its subsets of parameters. The proof develops identically to that of ($\preceq_S$), but considering all θ. See Appendix A of [7] for detailed demonstration. $\square$

Properties of iPSTL: partial orders III

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.

Proof.

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". $\square$

Properties of iPSTL: DAG I

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.

A partially ordered set $\langle X, \preceq \rangle$ froms a lattice if any two elements $x_1, x_2 \in X$ have a *join* and *meet*. The join $\sqcup : X \times X \to X$ and meet $\sqcap : X \times X \to X$ can be computed by means of two binary operators using the supremum (for $\sqcup$) and infimum (for $\sqcap$).

Any partially ordered set with a lattice structure can be rapresented by a DAG, adding directions to a Hasse diagram. Now, for proving the theorem, it is enough to show that the set of all iPSTL formulae with partial order $\preceq_P$ form a lattice. See Appendix B of [7] for detailed demonstration. $\square$

Properties of iPSTL: DAG II

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.

($\Rightarrow$) Since $L(\phi_1) \subset L(\phi_2)$, for any signal s, there are three possibilities: $s \in L(\phi_1)$; $s \in L(\neg\phi_1) \cap L(\phi_2)$; $s \in L(\neg\phi_1) \cap L(\neg\phi_2)$. Now, explore the three cases by evaluating $D(s, \phi_i)$. See Appendix C of [7].

($\Leftarrow$) Assume that there exists a signal s s.t. $D(s, \phi_1) \leq D(s, \phi_2)$ and $s \in L(\phi_1)$ but $s \notin L(\phi_2)$. Thus, we have $D(s, \phi_1) \geq 0$ and $D(s, \phi_2) \leq 0$, which is a contradiction since $D(s, \phi_1)$ and $D(s, \phi_2)$ cannot be zero simultaneously. $\square$

Properties of iPSTL: DAG III

Corollary 1

The following statements are equivalent:

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

Remark: 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 $D(s, \phi)$. If we replace $D(s, \phi)$ with $r(s, \phi)$ in Theorem 2, it is still true that 2) implies 1), but the implication from 1) to 2) doesn't always hold.

Counterexample: $\phi_1 = G_{[0,1]}(x_s \geq 1) \wedge G_{[0,1]}(x_s < 1)$ and $\phi_2 = G_{[0,1]}(x_s \geq 2) \wedge G_{[0,1]}(x_s < 2)$. It is clear that $\phi_1 \preceq_S \phi_2$. However, for a constant signal $x_s(t) = 0 \; \forall t$, we have that $r(s, \phi_1) = -1 > r(s, \phi_2) = -2$.

Algorithms

Optimization with Robustness Degree

The most used metric to measure the performance of anomaly detection algorithms is the *Missclassification Rate* (MR)

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

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.

Optimization with Robustness Degree

The most used metric to measure the performance of anomaly detection algorithms is the *Missclassification Rate* (MR)

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

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.

Problem: this rate ignores the degree of the error made

Optimization with Robustness Degree

The most used metric to measure the performance of anomaly detection algorithms is the *Missclassification Rate* (MR)

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

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.

Problem: this rate ignores the degree of the error made

Solution: use the *robusteness degree* $r(s, \phi)$

★ 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

Offline Anomaly Learning

Definition 12

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, $p_i = -1$ anomalous).

Goal: 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

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

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

$$l(p_i, r(s_i, \phi_\theta)) = \max\left(0, \epsilon_r - p_i r(s_i, \phi_\theta)\right) \tag{4}$$

where $\epsilon \ll 1$.

Offline Anomaly Learning algorithm I

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 δ
A formula length bound W
Output:
An iPSTL formula φ and valuation θ

for $i = 1$ to W **do**
  if $i = 1$ **then**
    $\mathcal{G}_1 \leftarrow DAGInizialitazion(V)$
    $List \leftarrow ListInizialization(\mathcal{G}_1)$
  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 (φ, θ)
return $MinimumCostNode(\mathcal{G}_W)$

Offline Anomaly Learning algorithm II

```
for i = 1 to W do
    if i = 1 then
        G_1 ← DAGInizialitazion(V)
        List ← ListInizialization(G_1)
    else
        G_i ← PruningAndGrowing(G_{i-1})
        List ← Ranking(G_i \ G_{i-1})
```

DAGInitialization: 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 $x_s \in V$

Offline Anomaly Learning algorithm II

for $i = 1$ to W **do**
  if $i = 1$ **then**
    $\mathcal{G}_1 \leftarrow DAGInizialitazion(V)$
    $List \leftarrow ListInizialization(\mathcal{G}_1)$
  else
    $\mathcal{G}_i \leftarrow PruningAndGrowing(\mathcal{G}_{i-1})$
    $List \leftarrow Ranking(\mathcal{G}_i \backslash \mathcal{G}_{i-1})$

DAGInitialization: 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 $x_s \in V$

PruningAndGrowing: applies the principle of Feature Subset Selection (FSS) by eliminating a fixed number of nodes. 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 $\varphi^{'} = \varphi_j \vee \varphi_b$
- if the *False Alarm Rate* is higher, the function would add $\varphi^{''} = \varphi_j \wedge \varphi_b$ to the graph

Offline Anomaly Learning algorithm III

for $i = 1$ to W **do**
  if $i = 1$ **then**
    $\mathcal{G}_1 \leftarrow DAGInizialitazion(V)$
    $List \leftarrow ListInizialization(\mathcal{G}_1)$
  else
    $\mathcal{G}_i \leftarrow PruningAndGrowing(\mathcal{G}_{i-1})$
    $List \leftarrow Ranking(\mathcal{G}_i \backslash \mathcal{G}_{i-1})$

Ranking: 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 $pa(k_i)$.

Offline Anomaly Learning algorithm IIII

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 (φ, θ)
return $MinimumCostNode(\mathcal{G}_W)$

ParameterEstimation: uses *simulated annealing* to find the optimal evaluation for φ by minimizing the cost J_α

Offline Anomaly Learning algorithm IIII

```
    while List ≠ ∅ do
        φ ← List.pop()
        (θ, MR) ← ParameterEstimation({(s_i, p_i)}_{i=1}^M, φ)
        if MR ≤ δ then
            return (φ, θ)
return MinimumCostNode(𝒢_W)
```

ParameterEstimation: uses *simulated annealing* to find the optimal evaluation for φ by minimizing the cost J_α

Termination: when a formula with low enough misclassification rate is found or W iterations are completed. In the second case, MinimumCostNode($\mathcal{G}_i$) returns the node with the minimum cost within $\mathcal{G}_i$

Offline Anomaly Learning algorithm IIII

```
    while List ≠ ∅ do
        φ ← List.pop()
        (θ, MR) ← ParameterEstimation({(s_i, p_i)}_{i=1}^M, φ)
        if MR ≤ δ then
            return (φ, θ)
return MinimumCostNode(𝒢_W)
```

ParameterEstimation: uses *simulated annealing* to find the optimal evaluation for φ by minimizing the cost J_α

Termination: when a formula with low enough misclassification rate is found or W iterations are completed. In the second case, MinimumCostNode($\mathcal{G}_i$) returns the node with the minimum cost within $\mathcal{G}_i$

Complexity:

- *Structural Inference:* from $\mathcal{O}(W \cdot 2^{|V|})$ to $\mathcal{O}(W \cdot |V|^2)$ thanks to Pruning
- *Parameter Estimation:* $\mathcal{O}(W(n^2m) \cdot log(M))$, where n is the number of step for each iteration and m is the number of valuation for each "temperature"

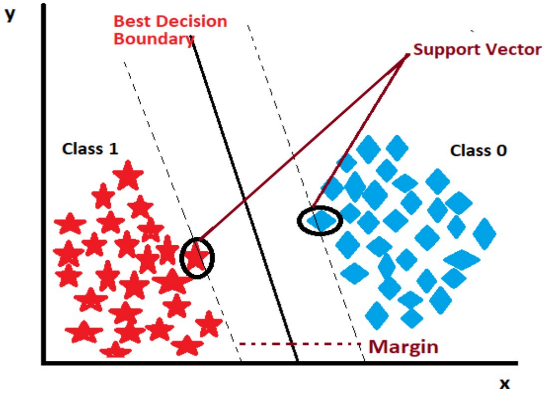
Anomaly Detection algorithm I

Problem: the *anomaly detection* problem requires dividing anomalous data from normal data, without knowing their labels

Anomaly Detection algorithm I

Problem: the *anomaly detection* problem requires dividing anomalous data from normal data, without knowing their labels

Solution: using a *one-class Support Vector Machine* (SVM)



Anomaly Detection algorithm II

Definition 13

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” between two predicted classes, 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 $L(\phi_\theta)$.

Anomaly Detection algorithm III

The algorithm proposed for the offline anomaly learning problem can be adapted to solve this problem, with two simple modifications:

- The input signals are not labeled, i.e. the inputs are $\{s_i\}_{i=1}^{M}$
- The ParameterEstimation function resolves

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

instead of 3.

Online Anomaly Learning algorithm I

Definition 14

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}.

Online Anomaly Learning algorithm I

Definition 14

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}.

What about parameters' optimization?

We can use the well known *Stochastic Gradient Descent*

Online Anomaly Learning algorithm II

For a fixed iPSTL formula structure φ, *Stochastic Gradient Gescent* can find its optimal parameterization θ^* if there exists a ϕ_{θ^*} with structure φ that can classify the data.

Definition 15
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.

Online Anomaly Learning algorithm III

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 η_{max}, η_{min}
Geometric rate α
Number of iterations before updating formulae database $checkInt$
Maximum number of iterations $numIters$
Output:
An iPSTL formula φ and valuation θ

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 $(i$ mod $checkInt) == 0$ **then**
    $formulae \leftarrow UpdateFormulae(formulae)$
  $\eta_{min} \leftarrow \eta$
return $bestFormula(formulae, traces)$

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

Online Anomaly Learning algorithm IIII

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 $(i$ mod $checkInt) == 0$ **then**
    $formulae \leftarrow UpdateFormulae(formulae)$
  $\eta_{min} \leftarrow \eta$
return $bestFormula(formulae, traces)$

ParameterEstimation: the initial one is done using the simulated annealing

Online Anomaly Learning algorithm IIII

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 $bestFormula(formulae, traces)$

ParameterEstimation: the initial one is done using the simulated annealing

ParameterUpdate: uses Stochastic Gradient Descent for updating the parameters

Online Anomaly Learning algorithm IIII

```
for i = 1, ..., numIters do
    traces ← UpdateTraces(traces, s_i, p_i)
    for (φ_k, θ_k) ∈ formulae do
        θ_k ← ParameterUpdate((s_i, p_i), φ_k, θ_k)
    η ← max(αη, η_min)
    if (i mod checkInt) == 0 then
        formulae ← UpdateFormulae(formulae)
    η_min ← η
return bestFormula(formulae, traces)
```

ParameterEstimation: the initial one is done using the simulated annealing

ParameterUpdate: uses Stochastic Gradient Descent for updating the parameters

★ 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}

Online Anomaly Learning algorithm IIII

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 $(i$ mod $checkInt) == 0$ **then**
    $formulae \leftarrow UpdateFormulae(formulae)$
  $\eta_{min} \leftarrow \eta$
return $bestFormula(formulae, traces)$

ParameterEstimation: the initial one is done using the simulated annealing

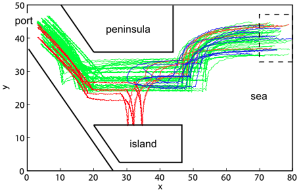
ParameterUpdate: uses Stochastic Gradient Descent for updating the parameters

★ 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}

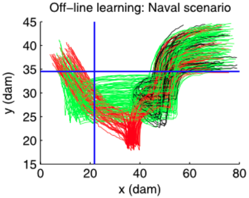
FormulaUpdate: If the missed detection rates are greater then the false alarm rates, then the conjunction of the two formulae is added to the formula database. Otherwise, the disjunction of the two is added.

Case studies

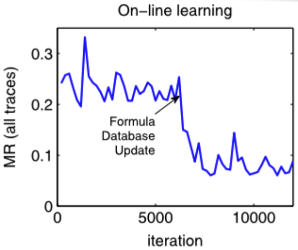
Naval Surveillance: scenario



(a) A naval surveillance example



(b) Results of offline inference



(c) The online misclassification rate over time

Scenario: we consider a scenario where we want to detect terrorist attacks or human trafficking near a naval port, using data provided by the Automatic Identification System (AIS)

Naval Surveillance: results

Anomaly Learning: 50 normal signals, 25 human trafficking signals, and 25 terrorist signals; with $n = 15$ and $m = 15$ this formula was obtained

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

Result: total misclassification rate 0.095

Naval Surveillance: results

Anomaly Learning: 50 normal signals, 25 human trafficking signals, and 25 terrorist signals; with $n = 15$ and $m = 15$ this formula was obtained

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

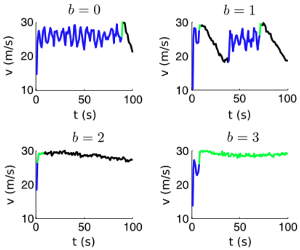
Result: total misclassification rate 0.095

Online Anomaly Learning: 1000 normal signals, 500 human trafficking signals, and 500 terrorist signals; After 2000 signals, with $\alpha = 0.995$, $\eta_{max} = 0.2$ and $\eta_{min} = 0.01$ this formula was obtained

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

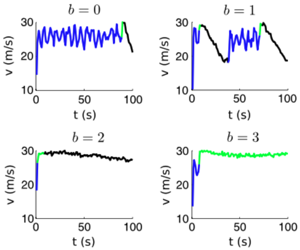
Result: final misclassification rate 0.0885

Train Network Monitoring



Scenario: the braking system is automated to regulate the velocity v below unsafe speeds and above low speeds, as shown in the top-left sub-figure. One possible attack is to disable the brake system, making the speed unregular.

Train Network Monitoring



Scenario: the braking system is automated to regulate the velocity v below unsafe speeds and above low speeds, as shown in the top-left sub-figure. One possible attack is to disable the brake system, making the speed unregular.

Anomaly Detection: 43 normal trajectories and 7 attack trajectories; with $n = 15$ and $m = 15$, this formula, which *perfectly separates the data*, was obtained

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

Conclusions

Conclusions

- Inference Parametric Signal Temporal Logic (iPSTL) was introduced

Conclusions

- Inference Parametric Signal Temporal Logic (iPSTL) was introduced
- Three *domain-independent* algorithms for anomaly detection and anomaly learning, both offline and online, were introduced

Conclusions

- Inference Parametric Signal Temporal Logic (iPSTL) was introduced
- Three *domain-independent* algorithms for anomaly detection and anomaly learning, both offline and online, were introduced
- The algorithms do not require as input a structure calculated by a domain expert

Conclusions

- Inference Parametric Signal Temporal Logic (iPSTL) was introduced
- Three *domain-independent* algorithms for anomaly detection and anomaly learning, both offline and online, were introduced
- The algorithms do not require as input a structure calculated by a domain expert
- The algorithms, thanks to the use of STL, produce an understandable formula as output

Conclusions

- Inference Parametric Signal Temporal Logic (iPSTL) was introduced
- Three *domain-independent* algorithms for anomaly detection and anomaly learning, both offline and online, were introduced
- The algorithms do not require as input a structure calculated by a domain expert
- The algorithms, thanks to the use of STL, produce an understandable formula as output
- Two case studies to demonstrate the abilities of the proposed algorithms

Thanks for the attention

References

References I

[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.

References II

[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.

References III

[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.

References IV

[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.