Rich Time Series Classification Using Temporal Logic

Chanyeol Yoo
Centre for Autonomous Systems
University of Technology Sydney
NSW 2007, Australia
Email: chanyeol.yoo@uts.edu.au

Calin Belta
Department of Mechanical Engineering
Boston University
Boston, MA 02215, USA
Email: cbelta@bu.edu

***Abstract*—Time series classification is an important task in robotics that is often solved using supervised machine learning. However, classifier models are typically not 'readable' in the sense that humans cannot intuitively learn useful information about the relationship between inputs and outputs. In this paper, we address the problem of rich time series classification where we propose a novel framework for finding a temporal logic classifier specified in a human-readable form. The classifier is represented as a *signal temporal logic* (STL) formula that is expressive in capturing spatial, temporal and logical relations from a continuous-valued dataset over time. In the framework, we first find a set of representative logical formulas from the raw dataset, and then construct an STL classifier using a tree-based clustering algorithm. We show that the framework runs in polynomial time and validate it using simulated examples where our framework is significantly more efficient than the closest existing framework (up to 920 times faster).**

I. Introduction

Classification of time series data is an important task in robotics and machine learning where a given data sequence is assigned to one of a number of categories. In robotics, sensors provide time series data, leading to many applications such as behaviour classification [24], anomaly detection [7], surveillance [10] and self-diagnosis [15], in which patterns of adversarial trajectories and abnormal behaviours over time should be learned. Beyond the assignment of data to categories, however, it is interesting to consider how to enrich the classifier model to be as intuitive as possible, so that humans can easily understand the underlying rules, learn from the model and use the classifier to inform decision making. This idea is aligned with current interest in methods that describe hidden system or environment models in a rich, readable manner so as to enable a deep level of interaction between autonomous systems and their designers and/or users [9, 2, 12]. Such algorithms should be computationally efficient in order to handle large datasets and execute on a variety of platforms; we would like to address the task of finding a rich and expressive classifier in a computationally efficient manner.

Time series classification is well studied in supervised machine learning [3, 7, 14, 25, 23, 16, 20]. However, the classifier models in traditional machine learning are often not in human-readable form, such as a hyperplanes in high-dimensional abstract parameter space [27]. Human readability is important because it can provide intuition into the internal reasoning of the classifier, which in turn can help non-expert users to gain a qualitative and quantitative understanding of the classification output. Recent work in *temporal logic inference* exploits the expressivity of temporal logic in modelling such a time series classifier [6, 13, 28, 21]. However, these frameworks are too computationally expensive to handle large datasets and may not run on mobile robots with limited resources, or expressivity is limited due to their logical structure.

In this paper, we propose a novel framework for the time series classification problem using natural language. The objective of the framework is to learn a classifier in the form of *signal temporal logic* (STL) formula from a set of labelled sequences. The framework consists of two parts. First, we extract *regions of interest* automatically from the data using a kernel function. We use the regions to find a set of logical formulas that represents the original dataset. Secondly, we construct an STL formula using a tree-based clustering algorithm given the set of logical formulas. On each node of the tree, an STL formula that minimises the misclassification rate is found. The framework is computationally efficient, and we show that it runs in polynomial time.

The classifier is expressed in such a way that spatial, temporal and logical relations are presented in human-readable form. A surprising side effect is that the classifier itself can be used online, prescriptively or proscriptively, to guide system behaviour (such as in self-diagnosis and recovery) to a desirable category.

We demonstrate the framework using two simulated examples, naval surveillance and self-diagnosis, and compare our expressivity and complexity results against similar work [6]. For both examples, we present the resulting temporal logic classifier and the corresponding misclassification rate. Our results show that the proposed framework is significantly more efficient than the closest work in the field.

This paper is organised as follows. In Sec. II, we briefly summarise existing work. In Sec. III, we present preliminaries, and in Sec. IV, we provide the formal problem statement. In Sec. V-A, we present a set of recursive rules for con-

C. Yoo is formerly with the Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA. This work was partially supported by the Office of Naval Research (N00014-14-1-0554).

verting an STL formula into its logical form, and how the misclassification rate of a formula can be computed w.r.t. a dataset. In Sec. VI and VII, we introduce our framework with a running example. In Sec. VIII, we show complexity analysis of the framework and how we achieve significant efficiency. Our simulated results are presented in Sec. IX and we discuss conclusions and future work in Sec. X.

II. Related Work

Various models in supervised machine learning can classify continuous-valued time series. *Support vector machines* (SVMs) extract features in parameter space and find hyperplanes that separate data into one of the categories [27]. Neural networks also learn to perform the same task [14]. Other important work includes dimensionality reduction and feature selection (e.g., principal component analysis or linear discriminant analysis) [1, 18]. However, this work does not provide features in intuitive form.

Some frameworks provide the output of classification in more readable form. *Hidden Markov models* [22, 5] address the classification problem of learning a classifier in the form of a transition matrix, where the matrix describes the state evolution. However, the matrix is insufficient in describing sophisticated features.

Reasoning about temporal relations has been well studied in artificial intelligence [26]. The main objectives are to find how events are correlated in time and to describe the relations using natural language. In [19], the authors use relations such as *after*, *before*, *begins* and *includes* for two given events. In [8, 28, 21], the use of temporal logic, such as *modal temporal logic*, *reified temporal logic*, *metric temporal logic* and *linear temporal logic*, is shown to reason about temporal relations. However, the focus of these approaches is to find a temporal ordering of logical events, and they do not consider important quantitative parameters such as time intervals and spatial metrics.

Temporal logic has been studied extensively for robotic task planning [29, 11, 17, 30]. More recent work attempts to use STL as a means to express a classifier intuitively. Unlike modal temporal logic, STL is capable of expressing time intervals to which a system is constrained. Early work in learning an *inference parametric STL* (iSTL) formula focused on finding parameters of given types of formulas [13]. The learning goal is to find the unknown spatial and temporal parameters of the formula. This approach is, however, not flexible in structure, and not scalable in practice.

The closest framework to ours is proposed in [6]. This work uses a decision tree to learn an STL classifier. In the framework, a pool of STL formulas is enumerated to find an optimal formula for a given decision tree node. The goal of the framework is to complete the decision tree and construct an overall STL formula. This work is similar to our approach in the sense that we also partially use a tree-based clustering algorithm, and the form of STL formula is quite general. However, in our approach, we have a clear stopping condition in the clustering algorithm, whereas their approach relies on an arbitrarily chosen stopping level. More importantly, our work is significantly faster in practice. The main reason is because we reduce the size of the dataset by finding a set of logical formulas that represents the raw dataset. We also have reduced the computation time by proving that the misclassification rate of a formula is related to the sum of individual logical formulas.

III. Preliminaries

A. Signals

An n-dimensional continuous-valued signal $\mathbf{x}$ is defined over a finite discrete time domain $\mathbf{T} = \{0, 1, \cdots, T-1\}$, where T is the length of the signal. We denote $\mathbf{x}(t) \in \mathbb{R}^n$ as the t-th state of the signal (i.e., $\mathbf{x} = \mathbf{x}(0)\mathbf{x}(1)\cdots\mathbf{x}(T-1)$), and $\mathbf{x}[t]$ as the suffix of the signal from time t (i.e., $\mathbf{x}[t] = \mathbf{x}(t)\mathbf{x}(t+1)\cdots$). A label $d = \top$ (i.e., true) is given if a signal exhibits a desired behaviour ($\bot$, otherwise).

A finite set of labelled signals is given as $\mathbf{X} = \{\{\mathbf{x}_1, d_1\}, \{\mathbf{x}_2, d_2\}, \cdots, \{\mathbf{x}_N, d_N\}\}$. We denote $\mathbf{X}_d \subseteq \mathbf{X}$ as the set of desired signals and $\mathbf{X}_{\bar{d}} \subseteq \mathbf{X}$ as that of undesired signals, where $\mathbf{X}_d \cap \mathbf{X}_{\bar{d}} = \emptyset$, and N_d, $N_{\bar{d}}$ and N are the numbers of signals in $\mathbf{X}_d$, $\mathbf{X}_{\bar{d}}$ and $\mathbf{X}$, respectively.

B. Signal Temporal Logic (STL) and Notations

The syntax of STL is defined as:

$$\phi ::= \top \mid \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathcal{G}_{[t_1,t_2]}\phi, \tag{1}$$

where $\top$ is *true* from Boolean algebra, $\mu \in \mathbf{P}$ is a *symbolic predicate*, $\neg$ and $\wedge$ are regular Boolean operators for negation and conjunction, respectively, and $\mathcal{G}$ is the temporal operators for '$\mathcal{G}$lobally' ('always') between the time interval $[t_1, t_2]$. Disjunction operator $\vee$ can be derived from conjunction and negation operators. Likewise, temporal operators $\mathcal{F}$ and $\mathcal{U}$ for 'in $\mathcal{F}$uture' ('eventually') and '$\mathcal{U}$ntil' can be derived [4].

The semantics of STL is defined over a discrete-time continuous-space signal $\mathbf{x}$ as:

$$\begin{aligned}
\mathbf{x}[t] &\models \mu \iff \mathbf{x}[t] \in \Pi(\mu)\\
\mathbf{x}[t] &\models \phi \iff \mathbf{x}[t] \models \phi\\
\mathbf{x}[t] &\models \neg\phi \iff \mathbf{x}[t] \not\models \phi\\
\mathbf{x}[t] &\models \phi_1 \wedge \phi_2 \iff \mathbf{x}[t] \models \phi_1 \text{ and } \mathbf{x}[t] \models \phi_2\\
\mathbf{x}[t] &\models \mathcal{F}_{[t_1,t_2]}\phi \iff \exists t' \in [t+t_1, t+t_2], \mathbf{x}[t'] \models \phi\\
\mathbf{x}[t] &\models \mathcal{G}_{[t_1,t_2]}\phi \iff \forall t' \in [t+t_1, t+t_2], \mathbf{x}[t'] \models \phi,
\end{aligned} \tag{2}$$

where $\Pi : \mathbf{P} \to \{[\mathbb{R}, \mathbb{R}]^n\}$ defines a set of *spatial predicate* as n-dimensional rectangles (i.e., *hyperrectangles*). For an n-dimensional signal state $\mathbf{x}(t) = [x_1, \cdots, x_n]^{\mathrm{T}}$ and $\Pi(\mu) = \{[p_1^1, p_2^1], \cdots, [p_1^n, p_2^n]\}$, $\mathbf{x}[t] \in \Pi(\mu)$ holds true if and only if $\bigwedge_{i\in[1,n]}(p_1^i \le x_i \le p_2^i)$ holds true.

Example 1. Given a 2-dimensional signal $\mathbf{x} = [1,1]^{\mathrm{T}}[3,3]^{\mathrm{T}}$, the term $\mathbf{x}(0)$ holds true for spatial predicate $\{[0,2],[0,2]\}$, whereas the term $\mathbf{x}(1)$ holds false. □

Throughout this paper, we use extra notations $\bar{\phi}$, $\phi + \phi$, $\phi \cdot \phi$ (or $\phi\phi$ without $\cdot$) for Boolean negation, disjunction and conjunction, respectively.

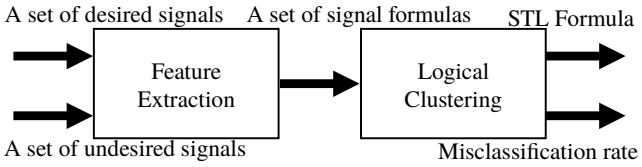


Fig. 1. Learning an STL formula from a labelled dataset

IV. Problem Statement

In our classification problem, the objective is to find a binary classifier for time series data in the form of an STL formula from a training set. Using the classifier, a testing signal is assigned to one of two classes: desired and undesired. The problem of finding the classifying STL formula, as shown in Fig. 1, is divided into two sub-problems:

Problem 1 (Feature extraction)**.** *Given a set of labelled signals* $\mathbf{X}$*, find a finite set of spatial predicates* Π *of interest and a set of logical expressions* $\mathbf{F}$ *(i.e., symbolic strings) corresponding to the signals.* □

Problem 2 (STL formula construction)**.** *Given a set of logical expression* $\mathbf{F}$ *over the set of predicates* Π*, find an STL formula* ϕ^**, such that the misclassification rate of* ϕ^* *is minimised with respect to* $\mathbf{F}$*.* □

V. Signal Formulas and Misclassification Rate

In this section, we formally define a geometrical representation of STL formulas and their mutually-exclusive form. We then define misclassification rate and present properties.

A. Geometrical Representation of STL Formulas

Any discrete-time STL formula ϕ can be written in a logical expression $f = \mathbf{f}(\phi, t)$ over a discrete-time signal $\mathbf{x}$ at time t. We call such an expression a *signal formula*.

Definition 1 (Signal formula)**.** A signal formula f is a logical expression evaluating the satisfaction of a given signal $\mathbf{x}$ over time. The formula is of the form

$$f ::= \mu_t \mid f + f \mid f \cdot f, \tag{3}$$

where $\mu_t \in \mathbf{V}_f$ is a *signal variable*, $\mathbf{V}_f = \{\mu_t \mid \forall \mu \in \{\emptyset, \mathbf{P}\}$ and $\forall t \in \mathbf{T}\}$, $\mathbf{V}_f(t) = \{\mu_\tau \in \mathbf{V}_f \mid \tau \equiv t\}$ and $\emptyset_t = \prod_{p \in \mathbf{P}} \bar{p}_t$. A signal variable μ_t holds true for signal $\mathbf{x}$ if and only if $\mathbf{x}(t) \in \Pi(\mu)$ holds true. □

Example 2. Suppose we have a signal formula $f = A_1 + B_0 \cdot B_1$, where $\mathbf{P} = \{A, B\}$ and $\mathbf{T} = [0, 1]$. The set of signal variables is $\mathbf{V}_f = \{A_0, B_0, A_1, B_1\}$. A signal $\mathbf{x}$ has to hold true for the following condition in order to satisfy the signal formula: $(\mathbf{x}(0) \in \Pi(A)) \vee (\mathbf{x}(0) \in \Pi(B)) \wedge (\mathbf{x}(1) \in \Pi(B))$ (i.e., 'the signal has to be in region A at time 0, or region B at time 0 and 1'). □

A signal formula f can be derived for an STL formula ϕ by the following recursive rules:

$$\begin{aligned}
\mathbf{f}(\top, t) &= \top \\
\mathbf{f}(\mu, t) &= \mu_t \\
\mathbf{f}(\neg\phi, t) &= \neg\mathbf{f}(\phi, t) \\
\mathbf{f}(\phi_1 \wedge \phi_2, t) &= \mathbf{f}(\phi_1, t) \wedge \mathbf{f}(\phi_2, t) \\
\mathbf{f}(\phi_1 \vee \phi_2, t) &= \mathbf{f}(\phi_1, t) \vee \mathbf{f}(\phi_2, t) \\
\mathbf{f}(\mathcal{G}_{[t_1,t_2]}\phi, t) &= \bigwedge_{t' \in [t_1,t_2]} \mathbf{f}(\phi, t + t') \\
\mathbf{f}(\mathcal{F}_{[t_1,t_2]}\phi, t) &= \bigvee_{t' \in [t_1,t_2]} \mathbf{f}(\phi, t + t').
\end{aligned} \tag{4}$$

Example 3. Given an STL formula $\phi = \mathcal{G}_{[0,1]}(\mathcal{F}_{[0,2]}A \wedge \mathcal{F}_{[0,2]}B)$ at time $t = 0$, the signal formula is:

$$\begin{aligned}
\mathbf{f}(\phi, 0) =& \mathbf{f}(\mathcal{F}_{[0,2]}A \wedge \mathcal{F}_{[0,2]}B, 0) \cdot \mathbf{f}(\mathcal{F}_{[0,2]}A \wedge \mathcal{F}_{[0,2]}B, 1) \\
=& \mathbf{f}(\mathcal{F}_{[0,2]}A, 0) \cdot \mathbf{f}(\mathcal{F}_{[0,2]}B, 0) \\
& \cdot \mathbf{f}(\mathcal{F}_{[0,2]}A, 1) \cdot \mathbf{f}(\mathcal{F}_{[0,2]}B, 1) \\
=& (A_0 + A_1 + A_2) \cdot (B_0 + B_1 + B_2) \cdot \\
& (A_1 + A_2 + A_3) \cdot (B_1 + B_2 + B_3) \\
=& A_1B_1 + A_1B_2 + \cdots + A_2A_3B_2B_3,
\end{aligned} \tag{5}$$

where the set of signal variables is $\{A_0, B_0, A_1, B_1, A_2, B_2\}$. □

In this paper, we assume that no predicates overlap in space, such that:

$$A_i \cdot B_i = \perp \text{ and } \Pi(A) \cap \Pi(B) = \emptyset, \forall A, B \in \mathbf{P}, \forall i \in \mathbf{T}. \tag{6}$$

Under the assumption, the following interesting reduction properties can be derived:

$$\begin{aligned}
A_i + \overline{B_i} &\iff \overline{B_i} \\
A_i \cdot \overline{B_i} &\iff A_i \\
\overline{A_i} + \overline{B_i} &\iff \top,
\end{aligned} \tag{7}$$

for any $i \in \mathbf{T}$ and $A, B \in \mathbf{P}$, where $A \neq B$.

Example 4. With the assumption, the resulting formula in Example 3 can further be simplified as

$$\begin{aligned}
\mathbf{f}(\phi, 0) =& A_1B_2 + B_1A_2 \\
& + A_0B_1A_3 + A_0B_2A_3 + B_0A_1B_3 + B_0A_2B_3.
\end{aligned} \tag{8}$$

□

B. Mutually Exclusive-Signal Formulas

Any signal formula f can be expressed in disjunctive form consisting *mutually-exclusive signal formulas* ρ, called *mutually-exclusive disjunctive normal form (exDNF)*

$$\begin{aligned}
\hat{f} &::= \rho \mid \hat{f} + \hat{f} \\
\rho &::= \prod_{t \in \mathbf{T}} v_t \in \mathbf{V}_f(t).
\end{aligned} \tag{9}$$

The set of disjuncts for $\hat{f}$, denoted as $\mathbf{D}_f$, is called an *exclusive set* of f. Intuitively, each disjunct contains one signal variable for all $t \in \mathbf{T}$.

Example 5. Continued from Example 2, the signal formula $f = A_1 + B_0B_1$ expressed in exDNF and the exclusive set are

$$\begin{aligned} f &= \emptyset_0 A_1 + B_0 A_1 + A_0 A_1 + B_0 B_1 \\ \mathbf{D}_f &= \{\emptyset_0 A_1, B_0 A_1, A_0 A_1, B_0 B_1\}. \end{aligned} \tag{10}$$

□

C. Misclassification Rate

Given a set of spatial predicates Π and a set of labelled signals $\mathbf{X}$, the misclassification rate can be calculated with respect to a signal formula or an STL formula $\psi \in \{f, \phi\}$ as follows:

$$\mathbf{r}(\psi, \Pi, \mathbf{X}) = \frac{N_d - g + b}{N}, \tag{11}$$

where g is the number of desired signals satisfying the formula ψ and b is the number of undesired signals satisfying the formula, such that:

$$\begin{aligned} g &= \mathbf{g}(\psi) = \|\{\mathbf{x} \in \mathbf{X} \mid (\mathbf{x} \models \psi) \wedge (d \models \top)\}\| \\ b &= \mathbf{b}(\psi) = \|\{\mathbf{x} \in \mathbf{X} \mid (\mathbf{x} \models \psi) \wedge (d \models \bot)\}\|. \end{aligned} \tag{12}$$

Throughout this paper, we call $\lambda = N_d/N$ a *diminishing constant*. In a short form, we often use $r(\psi)$, instead of $r(\psi, \Pi, \mathbf{X})$.

Lemma 1. The misclassification rate of a formula in the form $\psi_1 \vee \psi_2$ is $\mathbf{r}(\psi_1 \vee \psi_2) = \mathbf{r}(\psi_1) + \mathbf{r}(\psi_2) - \mathbf{r}(\psi_1 \wedge \psi_2)$. □

Proof: From (11), we have

$$\mathbf{r}(f_i) = \frac{N_d}{N} + \frac{b_i - g_i}{N}. \tag{13}$$

We substitute (13) into $\mathbf{r}(f_1 \vee f_2)$:

$$\begin{aligned} &\mathbf{r}(\psi_1 \vee \psi_2) \\ &= \frac{N_d - (g_1 + g_2 - g_{1\wedge 2}) + (b_1 + b_2 - b_{1\wedge 2})}{N} \\ &= \frac{N_d}{N} + \frac{b_1 - g_1}{N} + \frac{b_2 - g_2}{N} - \frac{b_{1\wedge 2} - g_{1\wedge 2}}{N} \\ &= r(\psi_1) + r(\psi_2) - r(\psi_1 \wedge \psi_2), \end{aligned} \tag{14}$$

where $g_{1\wedge 2}$ and $b_{1\wedge 2}$ are the number of desired and undesired signals in formula $\psi_1 \wedge \psi_2$. ■

Assuming mutual exclusion between two signal formulas of the form (9), we can simplify the equation further.

Lemma 2. The misclassification rate of a mutually-exclusive disjunction formula is $r(\hat{\psi}_1 \vee \hat{\psi}_2) = r(\hat{\psi}_1) + r(\hat{\psi}_2) - \lambda$. □

Proof: When two signals are mutually exclusive, $r(\hat{\psi}_1 \wedge \hat{\psi}_2) = \lambda$, since both $g_{1\wedge 2}$ and $b_{1\wedge 2}$ are zero. Substituting the equality into (14) gives:

$$r(\hat{\psi}_1 \vee \hat{\psi}_2) = r(\hat{\psi}_1) + r(\hat{\psi}_2) - \lambda. \tag{15}$$

■

Remark 1. By Lemma 2, $r(\hat{\psi}_1 \vee \hat{\psi}_2) < r(\hat{\psi}_1)$ holds true if and only if $r(\hat{\psi}_2) < \lambda$ for mutually exclusive formulas $\hat{\psi}_1$ and $\hat{\psi}_2$ of the form (9). For any added disjuncts satisfying the condition, the misclassification rate always reduces. □

Derived from (11), we have the following equalities for negations:

$$\begin{aligned} r(\overline{f}) &= 1 - r(f) \\ r(\overline{f_1} + \overline{f_2}) &= 1 - r(f_1 \cdot f_2). \end{aligned} \tag{16}$$

VI. Feature Extraction

In this section, we present our approach to convert a set of continuous-valued signals into the corresponding set of signal formulas. In this approach, we first extract *regions of interest* (ROI) from a set of continuous signals (*spatial extraction*), and then construct a set of representative signal formulas (*logical extraction*).

A. Extracting spatial predicates (spatial extraction)

With the set of continuous signals, we first extract a set of n-dimensional hyperrectangles that represents a set of ROI in the form of spatial predicates Π. We use the following extraction function over $\mathbb{R}^n$ to find a set of hyperplanes:

$$\begin{aligned} \mathbf{k}(\mathrm{x}, \mathbf{X}) = &\sum_{t \in \mathbf{T}} \sum_{\mathbf{x} \in \mathbf{X}_d} \exp(-\lambda \|\mathrm{x}(t) - \mathbf{x}(t)\|^2) \\ &- \sum_{t \in \mathbf{T}} \sum_{\mathbf{x} \in \mathbf{X}_{\bar{d}}} \exp(-\lambda \|\mathrm{x}(t) - \mathbf{x}(t)\|^2). \end{aligned} \tag{17}$$

The function is smoothened using weighted average and the positive and negative peaks over $\mathbb{R}^n$ with respect to $\mathbf{X}$ are extracted. Intuitively, positive and negative peaks represent the regions of high 'traffic' intensity for desired/undesired signals.

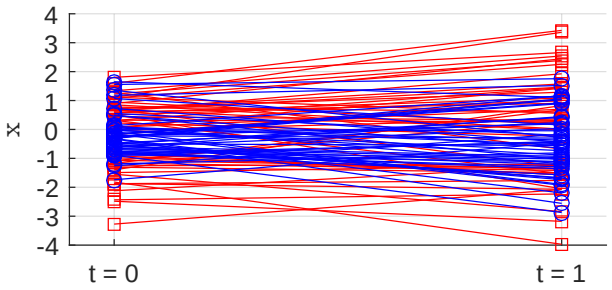
It is important to note that our spatial predicates are more representative than the ones from [6], since the extraction finds multi-dimensional regions while the other work only considers a single variable for a predicate. More details are in Sec. IX-B.

Example 6 (Running example)**.** In Fig. 2a, we have a set of 1-dimensional signals $\mathbf{x} \in \mathbf{X}$ over two time steps ($T = 2$), where time is on the horizontal axis and the value of signal is on the vertical axis. Signals are coloured in blue and red for exhibiting desired and undesired behaviours, respectively. In Fig. 2c, the signals are represented as a point in $\mathbb{R}^2$ space where the state at $t = 0$ is on horizontal axis and the state at $t = 1$ is on the vertical axis. Fig. 2b shows two peaks from spatial extraction resulting in two spatial predicates $\Pi = \{\{[-1.07, -0.01]\}, \{[1.02, 1.91]\}\}$ shown in Fig. 2c. □

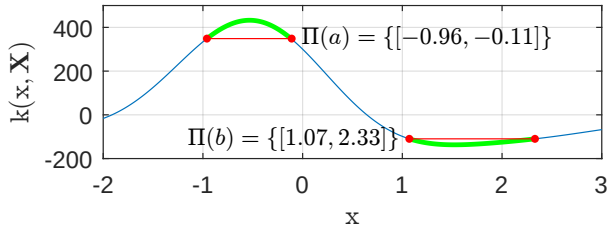
B. Constructing a set of signal formulas (logical extraction)

Given the set of spatial predicates Π for n-dimensional continuous signal $\mathbf{X}$, we construct the corresponding set of signal formulas $\mathbf{F}$. This process is called logical extraction. Suppose we have a signal $\mathbf{x}$, we find a signal formula in exDNF form

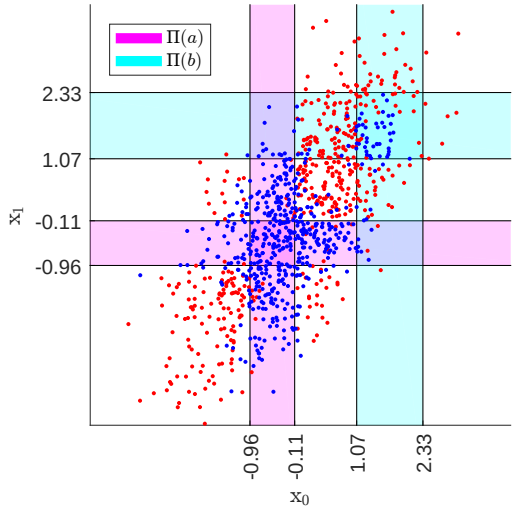
$$f = v_0 v_1 \cdots v_{T-1}, \tag{18}$$



(a) 1-dimensional signals over two time steps. Desired signals are plotted in blue; undesired signals are in red.



(b) Spatial extraction function over $\mathbb{R}^1$ after smoothing. Two ROI found.



(c) i-th axis represents the value of a signal at time i. Two spatial predicates $\Pi(a)$ and $\Pi(b)$ shown.

Fig. 2. An example set of desired and undesired signals ($N_d = 534$, $N_{\bar{d}} = 466$ and $N = 1000$) over two time steps

where $v_t \in \mathbf{V}_f(t)$ and $\mathbf{x}(t) \in \Pi(v)$. By the definition from (9), each signal formula is mutually exclusive. We find the set of signal formulas for all $\mathbf{X}$ and prune any duplicates to obtain a set of unique signal formulas.

Example 7. Continued from Example 6, we have a set of 9 representative signal formulas in Table I with diminishing constant $\lambda = 0.534$. Note that we only need to consider the symbolic formulas as opposed to 1000 continuous signals. □

VII. Logical Clustering

In this section, we present a decision tree-based algorithm for converting the set of signal formulas $\mathbf{F}$ into an STL formula. Starting from base node s_0 with $\mathbf{F}_0 = \mathbf{F}$, the

TABLE I

A set of mutually exclusive signal formulas that represents all signals $\mathbf{X}$ for Example 6. The formulas are in descending order of misclassification rates. The diminishing constant $\lambda = N_d/N$ is 0.534.

$\mathbf{F}$	$t=0$	$t=1$	f_i	$\mathbf{g}(f_i)$	$\mathbf{b}(f_i)$	$\mathbf{r}$
f_1	a	-	$a_0 \cdot \emptyset_1$	176	13	0.371
f_2	-	a	$\emptyset_0 \cdot a_1$	120	8	0.422
f_3	a	a	$a_0 \cdot a_1$	82	0	0.452
f_4	b	b	$b_0 \cdot b_1$	39	22	0.517
f_5	a	b	$a_0 \cdot b_1$	11	0	0.523
f_6	b	a	$b_0 \cdot a_1$	6	2	0.530
f_7	b	-	$b_0 \cdot \emptyset_1$	6	59	0.587
f_8	-	b	$\emptyset_0 \cdot b_1$	13	90	0.611
f_9	-	-	$\emptyset_0 \cdot \emptyset_1$	81	272	0.725

algorithm is to find an STL formula ϕ_k by enumerating a *pool* of STL formulas Φ at each node s_k, such that the misclassification rate of ϕ_k with respect to $\mathbf{F}_k$ is minimal. The set $\mathbf{F}_k$ is separated into two sets $\mathbf{F}_{k,c}$ and $\mathbf{F}_{k,d}$ for subsequent nodes, namely conjunction and disjunction nodes. The recursive algorithm is shown in Alg. 1.

Particularly in this paper, we are interested in the following forms of STL formulas

$$\begin{aligned} \Phi &= \{\Phi_1, \Phi_2, \Phi_3, \Phi_4\} \\ \text{where } \Phi_1 &= \{\mathcal{G}_{[t_1,t_2]}\mu\}, \Phi_2 = \{\mathcal{F}_{[t_1,t_2]}\mu\}, \\ \Phi_3 &= \{\mathcal{F}_{[t_1,t_2]}\neg\mu\}, \Phi_4 = \{\mathcal{G}_{[t_1,t_2]}\neg\mu\}, \end{aligned} \tag{19}$$

for all $\mu \in \mathbf{P}$, $t_1 \in [0, T-1]$ and $t_2 \in [t_1, T-1]$. Note that the framework can use any forms. For example, more complex forms such as $\mathcal{G}_{[t_1,t_2]}\mathcal{F}_{[t_3,t_4]}\mu$ and $\mu_1\mathcal{U}_{[t_1,t_2]}\mu_2$ can be used. For the forms in (19), we only need to compute misclassification rates for Φ_1 and Φ_4 using (11) and those for Φ_2 and Φ_3 can be computed using (16) (i.e., $r(\Phi_3) = 1-r(\Phi_1)$ and $r(\Phi_4) = 1-r(\Phi_2)$). Line 3 of Alg. 1 enumerates the pool to find the optimal formula and the misclassification rate. For a given STL formula $\hat{\phi}_k$ and set of signal formulas $\mathbf{F}_k$ for node k, the misclassification rate is computed, as described in (15), by

$$\begin{aligned} r(\phi) &= r\left(\bigvee_{f \in \mathbf{D}_{\mathbf{f}(\phi)}} f\right) \\ &= \left(\sum_{f \in \mathbf{D}_{\mathbf{f}(\phi)}} r(f)\right) - (|\mathbf{D}_{\mathbf{f}(\phi)}| - 1) \cdot \lambda_k. \end{aligned} \tag{20}$$

It is important to note that computing the misclassification for a given STL formula is simply the sum of all misclassification rates of the formulas in the exclusive set. If the rate were calculated without the table, we had to enumerate all signals to count the numbers in each category.

Example 8 (Misclassification rate of an STL formula)**.** For $\phi = \mathcal{F}_{[0,1]}a$, the signal formula and its exclusive set are

$$\begin{aligned} \mathbf{f}(\phi) &= a_0 + a_1 \\ \mathbf{D}_{\mathbf{f}(\phi)} &= \{a_0\emptyset_1, a_0b_1, a_0a_1, \emptyset_0a_1, b_0a_1\} \\ &= \{f_1, f_5, f_2, f_3, f_6\}, \end{aligned} \tag{21}$$

where f_i is from Table I. The misclassification rate of ϕ can be computed, as defined in (11), using Table I, such that

$$\begin{aligned} r(\phi) &= r(f_1) + r(f_2) + r(f_3) + r(f_5) + r(f_6) - 4\lambda \\ &= 0.162. \end{aligned} \tag{22}$$

□

By Remark 1, we are guaranteed to reduce the overall misclassification rate with ϕ_k if and only if $r(\phi_k) < \lambda_k$. Therefore, the algorithm stops when the rate is greater than the diminishing constant, returning $\phi_k = \bot$. If the misclassification rate of the formula is less than the diminishing constant, we use the formula to separate $\mathbf{F}_k$ into $\mathbf{F}_{k,c}$ and $\mathbf{F}_{k,d}$, such that

$$\begin{aligned} \mathbf{F}_{k,c} &= \{f \in \mathbf{F}_k \mid f \in \mathbf{D}_{\hat{\phi}_k}\} \\ \mathbf{F}_{k,d} &= \{f \in \mathbf{F}_k \mid f \notin \mathbf{D}_{\hat{\phi}_k}\}. \end{aligned} \tag{23}$$

If $\mathbf{F}_{k,d}$ is empty, it means there is no atomic formula that can separate $\mathbf{F}_k$ further. In this case, we finish branching from the node. Otherwise, we call another instances for $\mathbf{F}_{k,c}$ and $\mathbf{F}_{k,d}$ to find the corresponding $\phi^*_{k,c}$ and $\phi^*_{k,d}$.

The STL formula ϕ^*_k for node s_k is recursively found by

$$\phi^*_k = (\phi_k \wedge \phi^*_{k,c}) \vee (\neg\phi_k \wedge \phi^*_{k,d}), \tag{24}$$

where $\phi^*_k = \phi_k$ if the node is terminated. The overall formula for the whole dataset is $\phi^* = \phi^*_0$.

Example 9 (Decision tree)**.** Continued from Example 6, we demonstrate the decision tree in Fig. 3. In the base node s_0, we find the optimal atomic formula ϕ_0 that separates $\mathbf{F}_0$ into $\mathbf{F}_1$ and $\mathbf{F}_2$. At node s_1, no separation occurred for $\mathbf{F}_1$ since $\phi_1 \equiv \phi_0$ (i.e., $\mathbf{F}_4$ is empty). Hence, the STL formula for node s_1 is $\mathcal{F}_{[0,1]}a$. Similarly at node s_2, we get $\mathcal{G}_{[0,1]}b$. At the base node, we take the formulas ϕ_1 and ϕ_2 to construct the overall STL formula ϕ^*. With simplifications, we have

$$\begin{aligned} \phi^* &= (\phi_0 \wedge \phi_1) \vee (\neg\phi_0 \wedge \phi_2) \\ &= (\mathcal{F}_{[0,1]}a \wedge \mathcal{F}_{[0,1]}a) \vee (\neg\mathcal{F}_{[0,1]}a \wedge \mathcal{G}_{[0,1]}b) \\ &= \mathcal{F}_{[0,1]}a \vee \mathcal{G}_{[0,1]}b. \end{aligned} \tag{25}$$

The misclassification rate of the formula $r(\phi^*)$ is 0.145, where $\mathbf{D}_{\phi^*} = \{f_1, f_2, f_3, f_4, f_5, f_6\}$. □

VIII. Analysis

The time complexity for solving feature extraction is $\mathcal{O}(n \cdot N \cdot T \cdot |\mathbf{P}| + N^2)$. This is to perform the tasks of spatial/logical extraction and finding unique set of signal formulas. The worst-case number of signal formulas is $\max \mathbf{F} = \min(N, (|\mathbf{P}|+1)^T)$. However, the number is usually much smaller in practice since spatial predicates are found by locating the regions where signals commonly pass.

For each node s_k in the logical clustering algorithm in Alg. 1, the most time consuming task is to enumerate a pool to find the minimal signal formula with respect to the misclassification rate over $\mathbf{F}_k$. For the given forms of the formula pool in (19), the time complexity of enumeration is $\mathcal{O}(|\mathbf{P}| \cdot T^2)$. Note that the complexity of enumeration varies with a different pool. The worst-case number of nodes in the logical clustering is $(4 \cdot |\mathbf{F}| - 1)$. Hence, the worst-case time complexity for logical clustering is $\mathcal{O}(|\mathbf{F}| \cdot |\mathbf{P}| \cdot T^2)$. Overall, the running time is polynomial in $|\mathbf{F}|$, $|\mathbf{P}|$, T and N.

The data size also plays an important role in reducing computational load in practice. The raw signal size is $(N \cdot T)$ bytes assuming 8 bytes for each variable as used in Matlab. After feature extraction, the size reduces to $(T \cdot |\mathbf{F}| \cdot \text{ceil}(\log_2(|\mathbf{P}|+1))/8)$ bytes since each signal variable can be represented using $\text{ceil}(\log_2(|\mathbf{P}|+1))$ bits.

The form of our classifier is significantly more readable than that of other existing supervised learning frameworks. For instance, *support vector machine* (SVM) can be used for similar set of problems. However, the form of SVM classifier is a set of hyperplanes in *feature* space [27], from which human operator cannot understand and learn behaviours intuitively.

Algorithm 1 Decision tree-based logical clustering

function DecisionTree($\mathbf{F}, \mathbf{g}, \mathbf{b}$)
  initialise $\phi^*_c \leftarrow \top$, $\phi^*_d \leftarrow \bot$
  $\phi_k, r_a \leftarrow$ MinFormula($\mathbf{F}, \mathbf{g}, \mathbf{b}$)
  $\lambda \leftarrow |\mathbf{g}|/(|\mathbf{g}| + |\mathbf{b}|)$
  if $r_a > \lambda$ **then**
    return $\phi \leftarrow \bot$
  $\mathbf{F}_c, \mathbf{F}_d, \mathbf{g}_c, \mathbf{g}_d, \mathbf{b}_c, \mathbf{b}_d \leftarrow$ Separate $\mathbf{F}, \mathbf{g}, \mathbf{b}$ w.r.t. ϕ_k
  if IsEmpty($\mathbf{F}_c$) **then**
    $\phi^*_d \leftarrow$ DecisionTree($\mathbf{F}_d, \mathbf{g}_d, \mathbf{b}_d$)
  else if $\neg$IsEmpty($\mathbf{F}_c$) and $\neg$IsEmpty($\mathbf{F}_d$) **then**
    $\phi^*_c \leftarrow$ DecisionTree($\mathbf{F}_c, \mathbf{g}_c, \mathbf{b}_c$)
    $\phi^*_d \leftarrow$ DecisionTree($\mathbf{F}_d, \mathbf{g}_d, \mathbf{b}_d$)
  $\phi^* \leftarrow (\phi_k \wedge \phi^*_c) \vee (\neg\phi_k \wedge \phi^*_d)$
  return ϕ^*

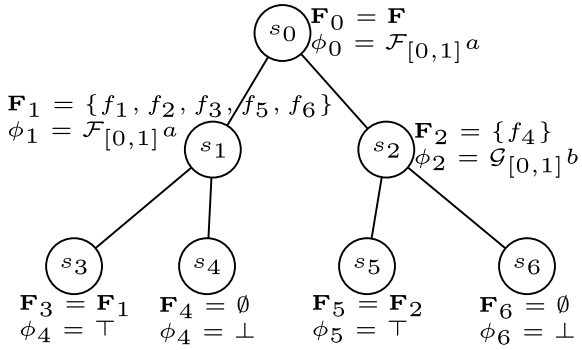
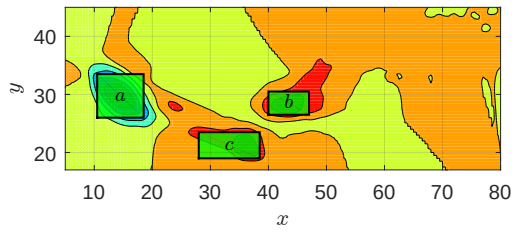


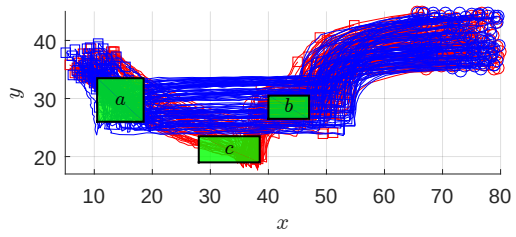
Fig. 3. The decision tree for Example 6 with respect to Table I. The overall formula is $\phi^* = \mathcal{F}_{[0,1]}a \vee \mathcal{G}_{[0,1]}b$..

IX. Results

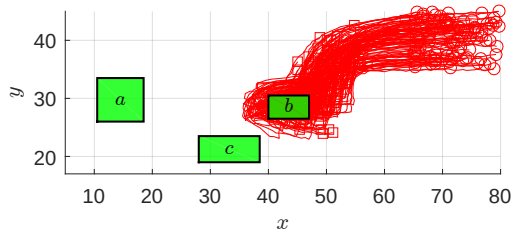
The framework is validated using simulated examples to demonstrate the expressivity of the learned classifier and the computational efficiency of the framework. For comparison, we use naval surveillance and self-diagnosis examples similar to [6]. In the simulations, we used MATLAB R2016b on Intel



(a) Result of spatial extraction showing ROI, where blue and red represent high intensity of successful and failed signals, respectively. Three regions a, b and c are extracted in descending order of absolute intensity.



(b) Vessel trarjectories and regions of interest



(c) Vessel is likely to fail (i.e., return) when it does not visit region b between t =16 and 21

Fig. 4. Naval surveillance dataset. Trajectories are over 61 sampling points (blue for desired, red for undesired). Starting and ending points are marked in circle and square.

i7 processor with 8Gb RAM without parallel computation. We have empirically validated the efficiency of the framework, presented in Table II, where the computation time and the reduction in data size are shown with 8 bytes for each real-valued variable. The numbers are compared against [6].

A. Naval Surveillance

In this example, a robot system surveils a waterway to look for anomalous behaviour of vessels. The simulated dataset of observations consists of 2000 2-dimensional vessel trajectories over 61 time steps entering Boston harbour as shown in Fig. 4, where the vessels are approaching from right to left. Starting and ending points are marked using circles and squares, respectively. Half of the trajectories, coloured in blue, are desired trajectories (i.e., successfully entered the harbour), and the rest, coloured in red, are undesired. Figure 4 shows a portion of the dataset.

The result of spatial extraction over $\mathbb{R}^2$ is presented in Fig. 4a. The dark red/blue colours represent the ROI, where blue regions have high intensity of desired trespass and vice versa for the red. In this example, we have chosen the top three peaks. Using the result, we extract three regions of interest a, b and c, in descending order of absolute intensity (i.e., $|\mathbf{k}|$).

TABLE II
RUNNING TIME, DATA SIZE FOR THE EXAMPLES ARE SHOWN. THE RUNNING TIME FOR [6] IS COMPARED.

Sim.	N, T	Feature Extraction	Logical Clustering	Total	Total [6]
Naval	2000, 61 (953kb)	$29.7s$ (23.7kb)	$64.2s$	$93.9s$	$\approx$ 24h
Fuel	1200, 200 (1.83Mb)	$6.69s$ (2.73kb)	$7.48s$	$14.2s$	$\approx$ 3h

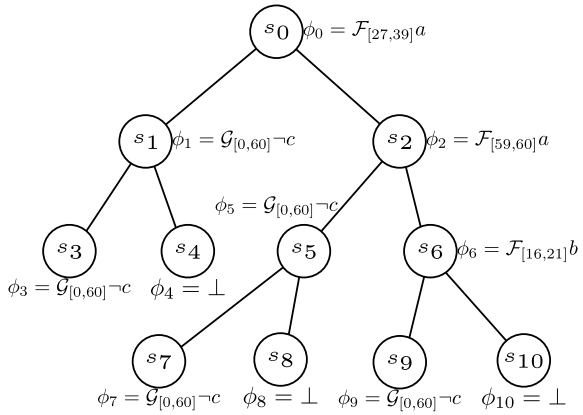


Fig. 5. Decision tree for naval surveillance example

The extracted spatial predicates are

$$\begin{aligned}\Pi(a) &= \{[11.5, 17], [27.5, 31.5]\}\\ \Pi(b) &= \{[41.5, 45], [27, 29]\}\\ \Pi(c) &= \{[34.5, 38], [20, 22.5]\}.\end{aligned} \tag{26}$$

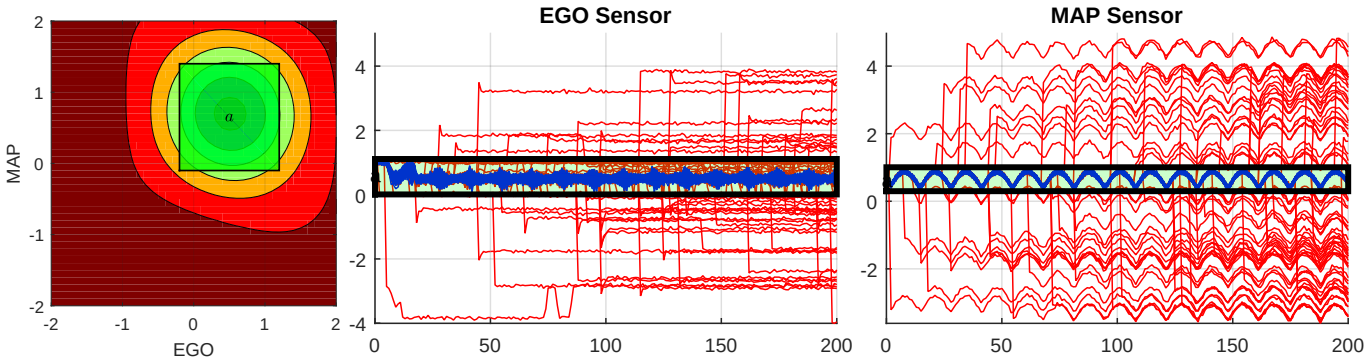
After logical extraction, we use the decision tree algorithm, as shown in Fig. 5, to find the following STL formula:

$$\phi^* = \mathcal{G}_{[0,60]}\neg c \wedge (\mathcal{F}_{[27,39]}a \vee \mathcal{F}_{[59,60]}a \vee \mathcal{F}_{[16,21]}b), \tag{27}$$

where the overall misclassification rate of the formula is 0.029. Glancing at the dataset, the most intuitive STL formula $\mathcal{F}_{[0,60]}a \wedge \mathcal{G}_{[0,60]}\neg c$ ('eventually reach region a and always avoid region c') has a misclassification rate of 0.125.

Intuitively, it is easy to see that region a is to be visited while region c is avoided, but the role of region b is not as obvious as the others. In Fig. 4c, we show a portion of undesired trajectories passing region b in the early part of the trajectories (i.e., between time 0 and 13). From this figure, we could conclude that the vessels approaching region b in the early stage are likely to head back to the entrance (i.e., failed to enter the harbour). Our proposed framework was able to capture and reveal such a hidden and non-obvious behaviour, and reason about the classification result. Such important information is not captured in [6].

The running times for feature extraction and logical clustering are 29.7 and 64.2 seconds, respectively (93.9 seconds overall). Compared to [6] where the algorithm running time was as large as nearly 24 hours (6 splits, 4 hours/split), our framework is dramatically more efficient (920 times faster). The substantial gain in computational complexity is due to the



(a) Spatial extraction over 2-d space. One positive ROI is found.

(b) EGO sensor readings over 200 time steps

(c) MAP sensor readings over 200 time steps

Fig. 6. Fuel control dataset. Desired and undesired signals are in green and red; undesired signals are shown in red. A portion of the dataset is shown. Regions of interest are shown as boxes with black border.

feature extraction part where the continuous-valued signals are abstracted into symbolic strings with much smaller data size. In this example, the size of the raw signals was approximately 953kb which reduced down to 23.7kb (40 times smaller).

B. Self-Diagnosis of an Engine Fuel Control System

In this example, a robot performs self-diagnosis of its engine fuel control system. We use a dataset of simulated observations of fuel system inputs and outputs that consists of 1200 signals over 200 time units, (600 desired). At each time point, the state of signal is represented by two values, EGO and MAP, as shown in Fig. 6, where blue represents desired behaviour and red represents undesired. More details can be found in [6].

Figure 6a shows the result of the spatial extraction over MAP and EGO space, where one spatial predicate is found:

$$\Pi(a) = \{[-0.2, 1.2], [-0.1, 1.4]\}. \tag{28}$$

Using the spatial predicate, the number of signals reduced from 1200 to 112 (signal formulas). The inferred STL formula after running the decision tree algorithm is:

$$\phi^* = \mathcal{G}_{[0,199]} a \tag{29}$$

The result is visually demonstrated in Fig. 6b and 6c.

In [6], each predicate is in the form $(\mathrm{x} \sim c)$ where $\sim \in \{>, \leq\}$ and $\mathrm{x}, c \in \mathbb{R}^1$. Because of its nature, it is not easy to capture high-dimensional regions of interest intuitively. For instance, the inferred STL formula in [6] is

$$\begin{aligned} \phi^*_{[6]} &= \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4 \\ \phi_1 &= \mathcal{G}_{[0,199]}(\mathrm{x}_2 > -0.536), \phi_2 = \mathcal{G}_{[0,199]}(\mathrm{x}_2 \leq 1.91) \\ \phi_3 &= \mathcal{G}_{[0,199]}(\mathrm{x}_1 > -0.819), \phi_4 = \mathcal{G}_{[79,199]}(\mathrm{x}_1 \leq 1.78), \end{aligned} \tag{30}$$

where x_1 and x_2 are EGO and MAP values, respectively. This result aligns with ours in a sense that the values are bounded most of the time, however, the formula is subtle and not easily readable as the spatial relations are not considered like in ours.

Our algorithm took 6.69 and 7.48 seconds for feature extraction and logical clustering, respectively (14.2 seconds in total), whereas the algorithm in [6] took about one hour (3 splits, 18 min/split). Again, our framework is significantly faster (253 times faster). The data size was reduced from 1.83Mb to 2.73kb (686 times smaller).

C. Complex Scenarios and Readability

The overall inferred STL formulas for complex scenarios beyond those demonstrated should remain human-interpretable. For a complex case, the framework is likely to consider a large number of spatial predicates, atomic formulas and logical relations compared to simple scenarios, which could result in an unreadably large formula. However, because spatial predicates and atomic formulas are induced in the order of importance, it is possible to keep the length of overall formula within a reasonable bound. If feature extraction gives too many regions, we could increase the threshold to accept only a top portion. Likewise, we could limit the number of iteration in logical clustering to restrict the number of induced atomic formulas. As a result, the limits could serve as tuning parameters for misclassification rate and readability.

X. Conclusion and Future Work

This paper has successfully addressed the problem of expressivity and computational efficiency in classifying time series datasets. The learned classifier is represented in a highly intuitive form that expresses spatial, temporal and logical relations using STL. The framework produces a set of symbolic formulas that represents the raw dataset. From this reduction, the framework is significantly more efficient compared to existing work in the field, and the time complexity is polynomial in important variables of interest.

The proposed framework has a number of important potential areas of future work, including *online* detection of faults (i.e., anomaly detection) and incremental learning. It is interesting to consider an extension to the unsupervised case, and this is relatively straightforward since our framework is a special case of the unsupervised setting.

REFERENCES

[1] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[2] Rajeev Alur, Salar Moarref, and Ufuk Topcu. Counter-strategy guided refinement of GR(1) temporal logic specifications. In *Proc of FMCAD*, pages 26–33, 2013.

[3] Andrés Arévalo, Jaime Niño, German Hernández, and Javier Sandoval. *High-Frequency Trading Strategy Based on Deep Neural Networks*, pages 424–436. 2016. ISBN 978-3-319-42297-8. doi: 10.1007/978-3-319-42297-8_40.

[4] Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Softw. Eng.*, 29(6):524–541, 2003.

[5] Phil Blunsom. Hidden Markov models. *Lecture notes, Dept. Comput. Sci., Software Eng. Melbourne School of Engineering, Australia*, 2004.

[6] Giuseppe Bombara, Cristian-Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. A decision tree approach to data classification using signal temporal logic. In *Proc. of ACM HSCC*, 2016.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM CSUR*, 41(3):15, 2009.

[8] Luca Chittaro and Angelo Montanari. Temporal representation and reasoning in artificial intelligence: Issues and approaches. *Ann. Math. Artif. Intell.*, 28(1-4):47–106, 2000.

[9] Jonathan DeCastro, Rüdiger Ehlers, Matthias Rungger, Ayça Balkan, and Hadas Kress-Gazit. Automated generation of dynamics-based runtime certificates for high-level control. *Discrete Event Dyn. Syst.*, 2016. 10.1007/s10626-016-0232-7".

[10] Xu Chu Ding, Calin Belta, and Christos G. Cassandras. Receding horizon surveillance with temporal logic specifications. In *Proc. of IEEE CDC*, pages 256–261, 2010.

[11] Xuchu Ding, Stephen L Smith, Calin Belta, and Daniela Rus. Optimal control of Markov decision processes with linear temporal logic constraints. *IEEE Trans. Automat. Contr.*, 59 (5):1244–1257, 2014.

[12] Rüdiger Ehlers and Ufuk Topcu. Resilience to intermittent assumption violations in reactive synthesis. In *Proc. of ACM HSCC*, pages 203–212, 2014.

[13] Austin Jones, Zhaodan Kong, and Calin Belta. Anomaly detection in cyber-physical systems: A formal methods approach. In *Proc. of IEEE CDC*, pages 848–853, 2014.

[14] Ken-ichi Kamijo and Tetsuji Tanigawa. Stock price pattern recognition-a recurrent neural network approach. In *Proc. of IEEE IJCNN*, pages 215–221, 1990.

[15] Kuniaki Kawabata, Tomoki Akamatsu, and Hajime Asama. A study of self-diagnosis system of an autonomous mobile robot: expansion of state sensory systems. In *Proc. of IEEE/RSJ IROS*, pages 1802–1807, 2002.

[16] Rami N Khushaba, Lei Shi, and Sarath Kodagoda. Time-dependent spectral features for limb position invariant myoelectric pattern recognition. In *Proc. of IEEE ISCIT*, pages 1015–1020, 2012.

[17] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. Robot.*, 25(6):1370–1381, 2009. doi: 10.1109/TRO.2009.2030225.

[18] Daniel T Larose. *Data mining methods & models*. John Wiley & Sons, 2006.

[19] Chong Min Lee. Temporal relation identification with endpoints. In *Proc. of NAACL HLT Student Research Workshop*, pages 40–45. Association for Computational Linguistics, 2010.

[20] Scott Lenser and Manuela Veloso. Non-parametric time series classification. In *Proc. of IEEE ICRA*, pages 3918–3923, 2005.

[21] Wenchao Li, Alessandro Forin, and Sanjit A Seshia. Scalable specification mining for verification and diagnosis. In *Proc. of ACM DAC*, pages 755–760, 2010.

[22] Lawrence Rabiner and Bing-Hwang Juang. An introduction to hidden Markov models. *IEEE Trans. Acoust., Speech, Signal Process*, 3(1):4–16, 1986.

[23] Mamun Bin Ibne Reaz, MS Hussain, and Faisal Mohd-Yasin. Techniques of EMG signal analysis: detection, processing, classification and applications. *Biol. Proced. Online*, 8(1):11–35, 2006.

[24] Patrick Riley and Manuela Veloso. Towards behavior classification: A case study in robotic soccer. In *Proc. of NCAI*, 2000.

[25] Saeid Sanei and Jonathon A Chambers. *EEG signal processing*. John Wiley & Sons, 2013.

[26] Lluís Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7(1):4–28, 1994.

[27] Hui Xue, Qiang Yang, and Songcan Chen. SVM: Support vector machines. CRS Press, 2010.

[28] Hengyi Yang, Bardh Hoxha, and Georgios Fainekos. Querying parametric temporal logic properties on embedded systems. In *Proc. of IFIP ICTSS*, pages 136–151, 2012.

[29] Chanyeol Yoo, Robert Fitch, and Salah Sukkarieh. Probabilistic temporal logic for motion planning with resource threshold constraints. In *Proc. of RSS*, 2012.

[30] Chanyeol Yoo, Robert Fitch, and Salah Sukkarieh. Provably-correct stochastic motion planning with safety constraints. In *Proc. of IEEE ICRA*, pages 981–986, 2013.