Time Series Learning using Monotonic Logical Properties

Marcell Vazquez-Chanlatte
University of California, Berkeley
marcell.vc@eecs.berkeley.edu

Shromona Ghosh
University of California, Berkeley
shromona.ghosh@eecs.berkeley.edu

Jyotirmoy V. Deshmukh
University of Southern California
jdeshmuk@usc.edu

Alberto Sangiovanni-Vincentelli
University of California, Berkeley
alberto@eecs.berkeley.edu

Sanjit A. Seshia
University of California, Berkeley
sseshia@eecs.berkeley.edu

ABSTRACT

We propose a new paradigm for time-series learning where users implicitly specify families of signal shapes by choosing *monotonic parameterized signal predicates*. These families of predicates (also called specifications) can be seen as infinite Boolean feature vectors, that are able to leverage a user's domain expertise and have the property that as the parameter values increase, the specification becomes easier to satisfy. In the presence of multiple parameters, monotonic specifications admit trade-off curves in the parameter space, akin to Pareto fronts in multi-objective optimization, that separate the specifications that are satisfied from those that are not satisfied. Viewing monotonic specifications (and their trade-off curves) as "features" for time-series data, we develop a principled way to bestow a distance measure between signals through the lens of a monotonic specification. A unique feature of this approach is that, a simple Boolean predicate based on the monotonic specification can be used to explain why any two traces (or sets of traces) have a given distance. Given a simple enough specification, this enables relaying at a high level "why" two signals have a certain distance and what kind of signals lie between them. We conclude by demonstrating our technique with two case studies that illustrate how simple monotonic specifications can be used to craft desirable distance measures.

ACM Reference Format:
Marcell Vazquez-Chanlatte, Shromona Ghosh, Jyotirmoy V. Deshmukh, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. 2018. Time Series Learning using Monotonic Logical Properties. In *Proceedings of ACM SIGKDD (SIGKDD'18)*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recently, there has been a proliferation of sensors that monitor diverse kinds of real-time data. This has led to system engineers facing a deluge of sensor data, with an urgent demand to extract meaningful analytics from it. Of particular relevance is the data representing *time-series behaviors* or *signals* generated by the systems and devices being monitored through such sensors. The broad impact of machine learning (ML) techniques for signal analysis is tangible in domains ranging from healthcare analytics [12] to smart transportation [6], and from autonomous driving [19] to social media [16]. While the importance of ML-based techniques cannot be stressed enough, there have been certain impediments to their universal adaptation by novice users. In particular, existing approaches to time-series learning need improvement along two directions: (1) the models obtained through ML algorithms are difficult to interpret, and (2) engineers with domain-expertise may not be ML experts, and are thus unable to use their domain knowledge to guide ML tasks.

A common way to encode domain specific knowledge into an ML task is to first transform the data into an *a priori* known *feature space*. To ease the burden on the user, such approaches often rely on large sets of generic features (such as those based on statistics or signal processing) coupled with dimensionality reduction techniques to identify and recommend significant features. This automated feature selection comes at a cost of the ability to leverage features as a tool to further the end-user understanding of the ML model. For example, a support vector machine or a clustering procedure using generic time series features may perform well on a given data-set, but for a given signal-trace, it is rarely clear *why* it gets assigned a particular label, or *why* it is grouped in a particular cluster.

One notable exception is recent work based on the idea of *shapelets*, where algorithms are developed to automatically identify shape-like features from the time-series data itself, and then use them for classification and clustering [15, 20, 24]. In [20], the authors extend basic shapelets to logical combinations of shapelets; in our view this method can be extended even further to lift shape-based reasoning to a more abstract form of logic-based reasoning where logical formulae can specify families of shapes. In this paper, we propose a new paradigm for time-series learning where users implicitly specify families of signal shapes by choosing *parameterized signal predicates*. These families of predicates (also called specifications) can be seen as infinite Boolean feature vectors, that are able to leverage a user's domain expertise.

Contributions. A key insight in our work is to specialize from parameterized specifications to monotonic specifications over signals. Informally, monotonic specifications have the property that as the parameter values increase, the specification becomes easier to satisfy. In the presence of multiple parameters, monotonic logics admit trade-off curves in the parameter space, akin to Pareto fronts

SIGKDD'18, 2018, London, UK
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

in multi-objective optimization, that separate the specifications that are satisfied from those that are not satisfied. Leveraging this insight we provide the following contributions:

(1) The introduction of monotonic specifications (and their trade-off curves) as a "feature" for time-series data.
(2) A principled way to bestow a distance measure between signals through the lens of a monotonic specification. Distance measure in hand, standard ML algorithms such as nearest neighbors (supervised) or agglomerative clustering (unsupervised) can be used to glean insights into the data.
(3) A simple Boolean predicate based on the monotonic specification that can be used to explain why any two traces (or sets of traces) have a given distance. Given a simple enough specification, this enables relaying at a high level "why" two signals have a certain distance and what kind of signals lie between them.

Motivating Example. Before developing our technique, we motivate with an example that illustrates how naïve feature spaces may fail to capture critical aspects of the data that the user may find desirable.

Most freeways have bottlenecks that lead to traffic congestion, and if there is a stalled or a crashed vehicle (or vehicles) at this site, then upstream traffic congestion can severely worsen. The problem of distinguishing a stalled vehicle from a vehicle facing regular traffic congestion from time-series data[1] of a vehicle's motion is challenging, as both vehicle trajectories have common characteristics such as slow average speeds, small minimum and maximum velocities, and so on. More concretely, Fig 1 shows a series of potential time series to which we would like to assign pairwise distances indicating the similarity (small values) or differences (large values) between two time-series. To ease exposition, we have limited our focus to the car's speed. In signals 0 and 1, both cars transition from high speed freeway driving to stop and go traffic. Conversely, in signal 2, the car transitions from stop and go traffic to high speed freeway driving. Signal 3 corresponds to a car slowing to a stop and then accelerating, perhaps due to difficulty merging lanes. Finally, signal 4 signifies a car encountering no traffic and signal 5 corresponds to a car in heavy traffic, or a possibly stalled vehicle. Suppose a user wished to find a feature space equipped with a measure to distinguish cars being stuck in traffic. Some properties might be:

(1) Signals 0 and 1 should be *very* close together since both show a car entering stop and go traffic in nearly the same way.
(2) Signals 2, 3, and 4 should be close together since the car ultimately escapes stop and go traffic.
(3) Signal 5 should be far from all other examples since it does not represent entering or leaving stop and go traffic.

For a strawman comparison, we consider two ways the user might assign a distance measure to the above signal space. At first, the user might treat the signals as a series of independent measurements and attempt to characterize the signals via standard statistical measures on the speed and acceleration (mean, standard deviation, etc.). Intuitively, the signal corresponding to acceleration should indicate a change in state of the car due to encountered

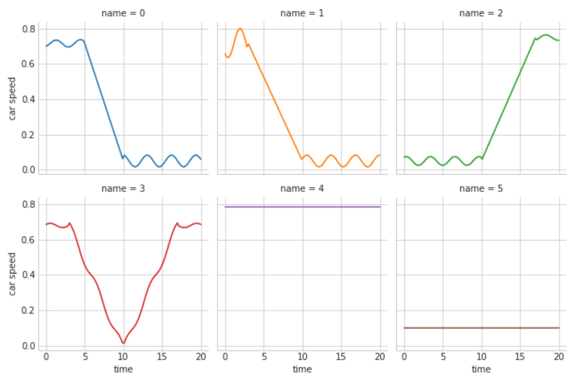


Figure 1: Example signals of car speeds on a freeway.

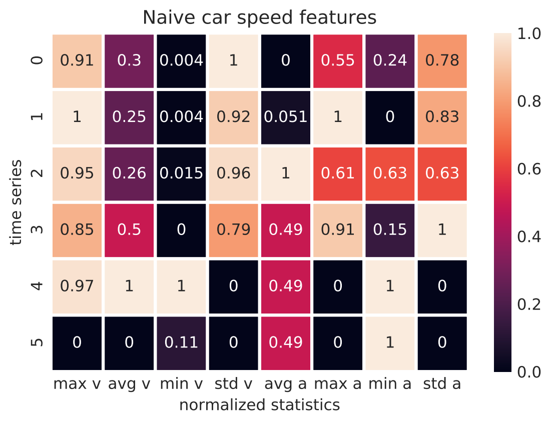


Figure 2: Statistical feature space

traffic. Similarly, the statistics on the velocity would measure if the car was mostly stationary or moving at a high speed. Fig 2 illustrates how the example signals look in this feature space with each component normalized between 0 and 1. The user might then use the euclidean distance of each feature to assign a distance between signals (the lower triangle of Fig 3). Unfortunately, in this measure, signal 4 is not close to signal 2 or 3, violating the second desired property. Further, signals 0 and 1 are not "very" close together violating the first property.

Next, seeing that related signals have similar shapes, the user then attempts to use dynamic time warping as a distance measure (the upper triangle of Fig 3). Now signals 0 and 1 are very close together, however, signal 3 is to close to signal 0 and signal 5 is too close to signals 0, 1, and 2.

[1]We note that such data can be obtained from fixed mounted cameras on a freeway, which is then converted into time-series data for individual vehicles, such as in [5].

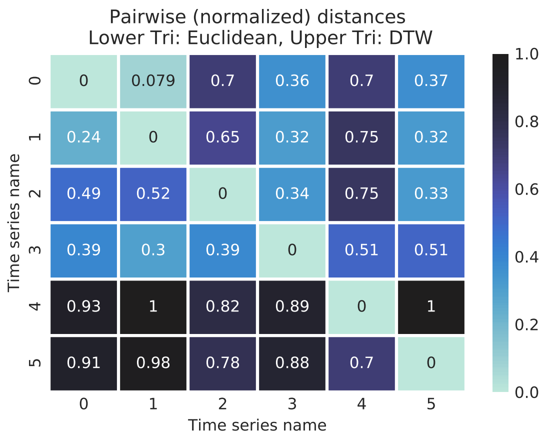


Figure 3: Straw man distance measures

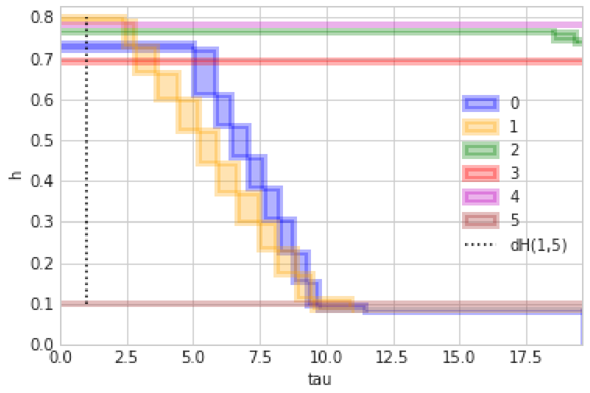


Figure 4: Trade-off boundaries in specification.

In the sequel, we shall show how using a simple monotonic specification to characterize the signal produces the desired results. Informally, the specification states

"Between time τ and 20, the car speed is always less than h."

Fig 4 illustrates the trade-off boundaries between h and τ induced by this specification and Fig 5 shows the pairwise Hausdorff distances between each boundary. As is easily confirmed, all 3 properties desired of the clustering algorithm hold. Furthermore, as a result of the Hausdorff distance between boundaries being equivalent to the distance between two parameter values, each distance between signals i and j is associated with a simple specification characterizing the signals that lie between them. For example, the dashed line in Fig 5 indicates the distance between signals 1 and 5. Our technique associates with this distance the specification,

"Between time 0 and 20, the car speed is always less than 8/10 AND between time 0 and 20, the car speed is eventually greater than 1/10."

which characterizes traces whose trade-off curves intersect the dashed line.

Before concluding the example, we note the human-interpretable property – car gets stuck in traffic – is captured by a related specification shown below; this specification gives rise to a distance measure that makes signal 5 to be closer to signals 0 and 1 and signal 3 to be between signals 1 and 2:

"Between time τ and 20, the car speed is always less than h AND between time τ' and 20, the car speed is always greater than h'".

Note that above human-interpretable specifications that enable clustering and classification are not obtained in an *ad hoc* fashion. In fact, these specifications are instantiations of signal predicates specified in a formal logic for time-series data known as Parametric Signal Temporal Logic (PSTL). PSTL is a logic originally developed for specifying signals in analog hardware circuits, and later extended to various domains including the automotive [8] and robotics [9]. It also is the underlying language for the specification library ST-Lib [7] used for embedded control specifications. While

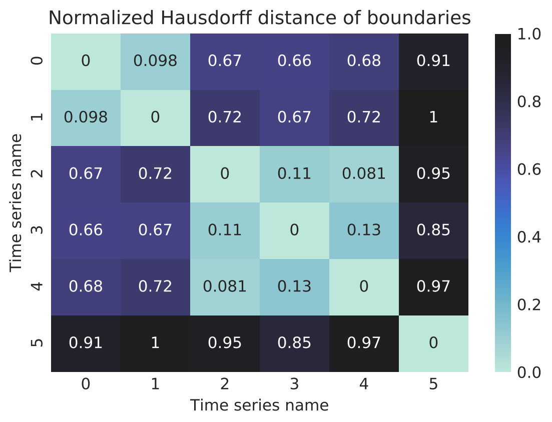


Figure 5: Boundary Hausdorff Distances

monotonic PSTL is a rich language that can specify many complex time-varying behaviors, we emphasize that end-users need not be experts in formal logics to use our techniques. In particular, one can create complex PSTL templates by composing simpler easier to understand templates.

2 PRELIMINARIES

The main object of analysis in this paper are time series.

Definition 2.1 (Time Series, Time Languages). Let T be a countable subset of $\mathbb{R}^{\geq 0}$ and $\mathcal{D}$ be some non empty set. A time series (or signal or trace), $\mathbf{x}$ is a map:

$$\mathbf{x} : T \to \mathcal{D} \tag{1}$$

Where T and $\mathcal{D}$ are called the time domain and (value) domain respectively. The set of all time series is denoted by $\mathcal{D}^T$.

Between any two time series one can define a metric[2] which measures their similarity.

[2]In this work we often conflate pseudo metrics and metrics for simplicity.

Definition 2.2 (Metric). Given a set X, a metric is a map,

$$d : X \times X \to \mathbb{R}^{\geq 0} \tag{2}$$

such that $d(x,y) = d(y,x)$, $d(x,y) = 0 \iff x = y$, $d(x,z) \leq d(x,y) + d(y,z)$.

Example 2.3 (Euclidean distance and Infinity Norm Metrics). The euclidean distance between two vectors is a metric $d_2(\vec{x},\vec{y}) \overset{\text{def}}{=} \sqrt{\vec{x}\cdot\vec{y}}$. Similarly, the infinity norm induced distance $d_\infty(\vec{x},\vec{y}) \overset{\text{def}}{=} \max_i(|x_i - y_i|)$ is a metric.

Example 2.4 (Hausdorff Distance). Given a set X with a distance metric d, the hausdorff distance is a distance metric between subsets of X. Namely, given subsets $A, B \subseteq X$:

$$d_H(A,B) \overset{\text{def}}{=} \max\left(\sup_{x\in A}\inf_{y\in B}(d(x,y)), \sup_{y\in B}\inf_{x\in A}(d(y,x))\right) \tag{3}$$

An ideal metric between traces should respect any domain specific properties that make two elements "similar".[3] A (logical) property φ assigns to each timed trace a truth value, and can be viewed as the subset of traces that have the property.

Definition 2.5 (Specification). A specification is a set of time series $\varphi \subseteq \mathcal{D}^T$. A time series, $\mathbf{x}$, is said to satisfy a specification iff $\mathbf{x} \in \varphi$. The set of all specifications is the power set of $\mathcal{D}^T$, $\mathcal{P}\left(\mathcal{D}^T\right)$.

Example 2.6. Consider the following specification related to the specification from the running example:

$$\phi_{ex} \overset{\text{def}}{=} \{\mathbf{x} \mid t > 0.2 \implies \mathbf{x}(t) < 1\} \tag{4}$$

Informally, this specification says that after $t = 0.2$, the value of the time series, $x(t)$, is always less than 1.

Given a finite number of properties, one can then "fingerprint" a time series as a Boolean feature vector. However, many times the relevant properties are not easily captured by a finite sequence of binary features. For example, one might image a single real valued feature $f(x(t))$ taking values between 0 and 1. This feature encodes an uncountably infinite family of Boolean features $f(x(t)) = k$ for $k \in [0,1]$. Such families are called parametric specifications. For simplicity, we assume that the parameters are a subset of the unit hyper-square.

Definition 2.7 (Parametric Specifications). A parametric specification is a map:

$$\varphi : [0,1]^n \to \mathcal{P}\left(\mathcal{D}^T\right) \tag{5}$$

where $n \in \mathbb{N}$ is the number of parameters.

As seen above, parametric specifications arise naturally from syntactically substituting constants with parameters in the description of a specification.

Example 2.8. The parametric specification given in the introduction can be formalized substituting τ for 0.2 and h for 1 in Ex 2.6.

$$\varphi_{ex}(\tau,h) \overset{\text{def}}{=} \{\mathbf{x} \mid t > \tau \implies \mathbf{x}(t) < h\} \tag{6}$$

[3]Colloquially, if it looks like a duck and quacks like a duck, it should have a small (or zero) distance to a duck.

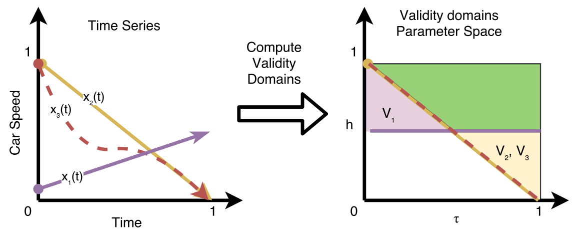


Figure 6: On the right are the validity domains for the traces, x_1, x_2, x_3 w.r.t. φ_{ex}. The green region is the intersection of all three validity domains. Observe that the validity domains of $x_2(t)$ and $x_3(t)$ are equivalent under this specification.

To generalize the "fingerprint" of a time series for parametric specifications, first observe that the value of a boolean feature vector is exactly determined by which entries are set to True. Analogously, the set of parameter values for which a parameterized specification, φ, would yield true on a given trace, called the validity domain, acts the "fingerprint".

Definition 2.9 (Validity domain, Validity domain boundary). Given a parametric specification of n parameters and a trace $\mathbf{x}$, the validity domain is the set,

$$\mathcal{V}_{\mathbf{x}}(\varphi) \overset{\text{def}}{=} \left\{\theta \in [0,1]^n \mid \mathbf{x} \in f(\theta)\right\} \tag{7}$$

In general, the validity domain can be arbitrarily complex which makes developing a distance metric between validity domains subtle. We circumvent such hurdles by specializing to monotonic specifications, for which the validity domains are remarkably simple.

Definition 2.10 (Monotonic Specifications). A parametric specification is said to be monotonic if

$$\theta \leq \theta' \implies f(\theta) \subseteq f(\theta') \tag{8}$$

where $\leq$ is the standard product ordering on $[0,1]^n$.

Before examining the validity domain of monotonic specifications, observe that the parametric specification in Ex 2.8 (and thus intro example) is monotonic.

Proposition 2.11. *Given a monotonic specification, φ and a time series $\mathbf{x}$, the boundary of the validity domain, $\partial\mathcal{V}_x(\varphi)$, of a monotonic specification is a hyper-surface that segments $[0,1]^n$ into two connected components.*

In the sequel, we develop a distance metric between validity domains which characterizes the similarity between two time series under the lens of a monotonic specification.

3 LOGIC-RESPECTING DISTANCE METRIC

Observe that the validity domains of monotonic specifications are uniquely defined by the hyper surface that separates them from the rest of the parameter space. Similar to Pareto fronts in a multi-objective optimization, these boundaries encode the trade-offs required in each parameter to make the specification satisfied for a given time series. This suggests a simple procedure to define a distance metric between time series that respects their logical properties: Given a monotonic specification, a set of time series, and a distance metric between validity domain boundaries:

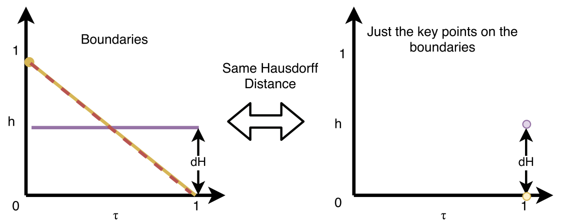


Figure 7: Illustration of Hausdorff distance between $x_1(t)$ and $x_2(t)$ from Fig 6. On the left the boundaries are shown and on the right the key points responsible for the Hausdorff distance are shown.

(1) Compute the validity domain boundaries for each time series.
(2) Compute a distance between the validity domain boundaries.

Of course, the benefits of using this metric would rely entirely on whether (i) The monotonic specification captures the relevant domain specific details (ii) The distance between validity domain boundaries is (in)sensitive to outliers. While the choice of specification is highly domain specific, we argue that for many monotonic specifications, the distance metric should be sensitive to outliers as this represents a large deviation from the specification. This sensitivity requirement seems particularly apt if the size of the specification grows linearly or super linearly as the parameters increase. To this end, we propose using Hausdorff distance between validity domains for three reasons:

(1) The Hausdorff distance is sensitive to outliers.
(2) The Hausdorff distance between two boundaries reduces to the distance between two parameters from each boundary. These elements explain why the boundaries differ.
(3) If two boundaries have Hausdorff distance d^* , if one boundary proposes a parameter (and thus specification), the other boundary must have a parameter (specification) within d^* units of distance. Thus, the Hausdorff distance measures how well the two validity domains can simulate each other.

We define our new distance metric between time series as:

Definition 3.1. Given a monotonic specification $\varphi : [0,1]^n \to \mathcal{P}\left(\mathcal{D}^T\right)$ and a distance metric on the parameter space $([0,1]^n, d)$, the logical distance between two time series, $\mathbf{x}(t), \mathbf{y}(t) \in \mathcal{D}^T$ is:

$$d_\varphi(\mathbf{x}(t), \mathbf{y}(t)) \overset{\text{def}}{=} d_H\left(\partial\mathcal{V}_\mathbf{x}(\varphi), \partial\mathcal{V}_\mathbf{y}(\varphi)\right) \tag{9}$$

As shown in the introduction, the property for the Hausdorff distance between two boundaries (signals) to reduce to the Hausdorff distance between two parameter values (and thus specifications) can be leveraged to describe the set of signals that lie "between" the signals. More precisely, if parameters θ and θ' are responsible for the Hausdorff distance of signals i and j, by signals "between" we mean all signals k whose validity domain boundary intersects the straight line from i to j. If using the infinity norm as the parameter space distance metric, then this line corresponds to a degenerate 1-d hyper-box whose specification [23] is given by:

$$\varphi(\theta') \cap \left([0,1]^n \setminus \varphi(\theta)\right) \tag{10}$$

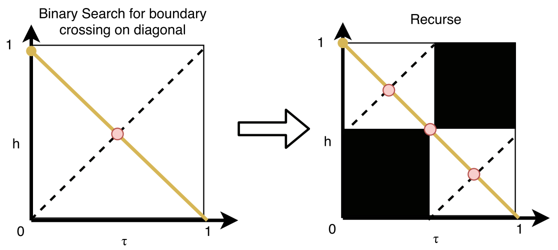


Figure 8: Illustration of procedure introduced in [17] to recursively approximate a validity domain boundary to arbitrary precision.

or in logical notation

$$\varphi(\theta') \wedge \neg\varphi(\theta) \tag{11}$$

where we have assumed w.o.l.o.g that $\theta \leq \theta'$.

Proposition 3.2. *As Eq 10 excludes signals whose boundaries are above and below θ' and θ resp., the boundaries corresponding to each signal in Eq 10 must intersect with the line between θ and θ'.*

3.1 Computing the Logical Distance

Before continuing onto the case studies, we briefly discuss how to compute the logical distance metric. First and foremost, a validity domain boundary of a monotonic specification can be recursively approximated to arbitrary precision via binary search on the diagonal of the parameter space [17]. This approximation yields a series of overlapping axis aligned rectangles that are guaranteed to contain the boundary (see Fig 8). Given enough precision, one can then sample points within each rectangle and compute the point wise Hausdorff distance [21]. Alg 1 describes the above procedure as pseudo-code.

Algorithm 1 Compute Logical Distance

```
1: function LOGICALDIST($\varphi, \mathbf{x}, \mathbf{y}, ([0,1]^n, d)$)
2:     $\text{recSet}_\mathbf{x} \leftarrow \text{approx\_boundary}(\mathbf{x}, \varphi)$
3:     $\text{recSet}_\mathbf{y} \leftarrow \text{approx\_boundary}(\mathbf{x}, \varphi)$
4:     $\text{points}_\mathbf{x} \leftarrow \bigcup_{r \in \text{rec\_set}_\mathbf{x}} \text{samplePoints}(r)$
5:     $\text{points}_\mathbf{y} \leftarrow \bigcup_{r \in \text{rec\_set}_\mathbf{y}} \text{samplePoints}(r)$
6:     return $\text{Hausdorff}(\text{points}_\mathbf{x}, \text{points}_\mathbf{y}, d)$
```

In our implementation [22], the boundary approximations occur in parallel and we use infinity norm for the distance in the parameter space. This enables a straight forward calculation to obtain upper and lower bounds by analyzing the rectangles directly.

Regarding scaling, we briefly remark that properly normalizing and pruning rectangles in the Hausdorff approximation is absolutely necessary to get moderate performance. In the worst case, the number of rectangles required to approximate the boundary scales exponentially and the Hausdorff distance is quadratic (although each part of the computation is embarrassingly parallel).

4 CASE STUDIES

In our case studies we utilized Parametric Signal Temporal Logic (PSTL) as a formalism/language to encode specifications.

4.1 PSTL as a Feature Design Language

Real-time temporal logics are a formalism for reasoning about finite or infinite timed series. These logics add to propositional logic modal operators to encode temporal concepts. Signal Temporal Logic [18] was proposed in the context of analog and mixed-signal circuits as a specification language for real-valued signals.

Signal Temporal Logic. Atoms in STL formulas take the form $f(\mathbf{x}) \sim c$, where f is a function from $\mathcal{D}$ to $\mathbb{R}$, $\sim \in \{\geq, \leq, =\}$, and $c \in \mathbb{R}$. Temporal formulas are formed using temporal operators, "globally" (denoted as **G**), "in the future" (denoted as **F**) and "until" (denoted as **U**) that can each be indexed by an interval I.

Definition 4.1 (Signal Temporal Logic). A formula in Signal Temporal Logic is syntactically defined via the grammar:

$$\begin{aligned} I &:= (a,b) \mid (a,b] \mid [a,b) \mid [a,b] \\ \varphi &:= \mathit{true} \mid f(\mathbf{x}) \sim c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \, \mathbf{U}_I \, \varphi_2 \end{aligned} \tag{12}$$

In the above grammar, $a, b \in T$, and $c \in \mathbb{R}$. The globally (**G**) and in the future (**F**) operators are defined syntactic sugar for special cases of the until operator: $\mathbf{F}_I\varphi \triangleq \mathit{true}\,\mathbf{U}_I\,\varphi$, and $\mathbf{G}_I\varphi \triangleq \neg\mathbf{F}_I\neg\varphi$. We use the notation $(\mathbf{x}, t) \models \varphi$ to mean that the suffix of the timed trace $\mathbf{x}$ beginning at time t satisfies the formula φ. The formal semantics of an STL formula are defined recursively:

$$\begin{array}{lcl} (\mathbf{x},t) \models f(\mathbf{x}) \sim c & \iff & f(\mathbf{x}(t)) \sim c \text{ is true} \\ (\mathbf{x},t) \models \neg\varphi & \iff & (\mathbf{x},t) \not\models \varphi \\ (\mathbf{x},t) \models \varphi_1 \wedge \varphi_2 & \iff & (\mathbf{x},t) \models \varphi_1 \;\wedge\; (\mathbf{x},t) \models \varphi_2 \\ (\mathbf{x},t) \models \varphi_1 \, \mathbf{U}_I \, \varphi_2 & \iff & \exists t_1 \in t \oplus I : (\mathbf{x},t_1) \models \varphi_2 \;\wedge \\ & & \quad \forall t_2 \in [t, t_1) : (\mathbf{x},t_2) \models \varphi_1 \end{array}$$

We write $\mathbf{x} \models \varphi$ as a shorthand of $(\mathbf{x}, 0) \models \varphi$.

Example 4.2. The running example specification, ϕ_{ex}, can encoded in STL as:

$$\phi_{ex} = \mathbf{G}_{[0.2,\infty)}(\mathbf{x} < 1) \tag{13}$$

which reads "Always between t=0.2 and infinity, $\mathbf{x}(t)$ is less than 1".

Parametric STL (PSTL) and Monotonic STL. PSTL [1] is a natural extension of STL that syntactically replaces constants within a STL formula with constants. The polar fragment of PSTL [1] is a syntactically identifiable subset of PSTL that are monotonic specifications in accordance to Def 2.10. The details and formal definition of this fragment are outside the scope of this work; however, all PSTL formula given are monotonic and in the polar fragment.

Example 4.3. The parametric specification, $\varphi_{ex}(\tau, h)$ can be encoded as PSTL as:

$$\varphi_{ex}(h, \tau) = \mathbf{G}_{[\tau,\infty)}(\mathbf{x} < h) \tag{14}$$

4.2 Case Study 1

In this case study, we take our running example, and attempt to apply the same (or similar) templates to real traffic data. To improve driver and traffic on highways, the Federal Highway Administration collected detailed traffic data on southbound US-101 freeway, in Los Angeles [5]. Traffic through the segment was monitored and recorded through eight synchronized cameras, next to the freeway. A total of 45 minutes of traffic data was recorded including vehicle trajectory data providing lane positions of each vehicle within the study area.

We picked three 80 seconds velocity trajectories, and split them into 20 second segments. We analyzed the resulting 12 segments using the following monotonic specifications,

$$\begin{aligned} \phi_{ec}(\tau, a) &= \mathbf{G}_{[\tau,\infty]}(v(t) \leq a) \\ \phi_{lc}(\tau, a) &= \mathbf{G}_{[\tau,\infty]}(v(t) \geq 60 - a) \end{aligned} \tag{15}$$

where ϕ_{ec} and ϕ_{lc} characterizes trajectories where a car potentially enters and leaves high congestion areas respectively. We combine the features by averaging their logical distances. Fig 9 shows the corresponding the resulting pairwise distances and clustering dendogram (signals from each clusters are shown in Fig 10). Cluster 0

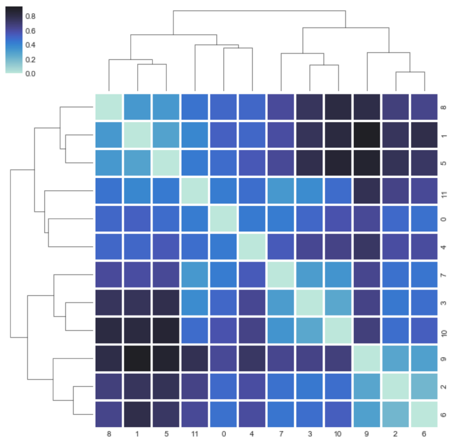


Figure 9: Cluster map corresponding to features ϕ_{ec} and ϕ_{lc} on Highway 101 data.

contains trajectories leaving high congestion and hence the velocity increases towards the end of the trajectory. Cluster 1 contains trajectories entering high congestion, shown by the decrease in velocity. Cluster 2 shows the trajectories that enter and leave high congestion areas in the same segment. Finally cluster 3 indicates entering moderate congestion. Observe that these clusters are remarkably similar to the toy example templates from our running example.

4.3 Case Study 2 - CPS Grader

Massively Open Online Courses (MOOCs) present instructors the opportunity to learn from a large amount of collected data to automatically identify common correct solutions and mistakes. Developing automatic grading systems for CyberPhysical System (CPS) MOOCs using simulation presents a particularly unique challenge.

Juniwal et al. [11] demonstrated a semi-supervised procedure for a CPS MOOC that first used dynamic time warping to cluster traces of student solutions, asked the instructor to label representatives from the clusters, and then extracted a STL formula from a PSTL template that characterized the cluster. Vazquez-Chanlatte et al. [23] provided an unsupervised technique which used PSTL templates from [11] to induce a distance metric that reproduced an arbitrarily chosen subset of the results from [11]. This distance metric required the user to provide an *a priori* total ordering on the parameter space which was used to select a single representative point on the

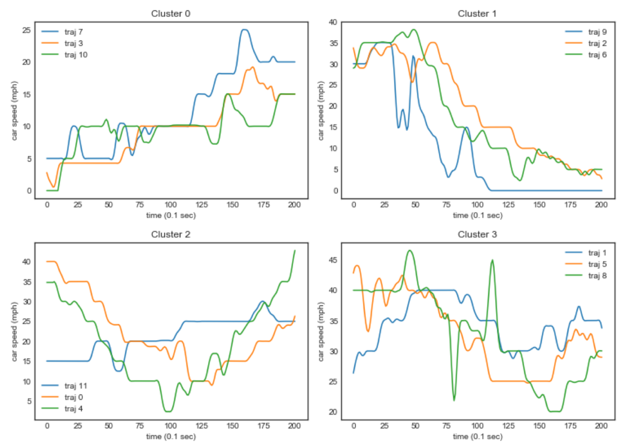


Figure 10: Clusters corresponding to the cluster map in Fig 9

boundary of the validity domain. Then for each signal, the distance between representative points served as the distance metric for the signals. Thus, while the PSTL induced distance lessened the labeling burden (since under the PSTL template, many superficially different clusters become equivalent), the total orderings required a fair amount of expertise to craft. In this experiment, we demonstrate that simple PSTL templates along with the logical distance (Eq (9)) can be used to reproduce the results of [11] without labeled data and without an *a priori* total ordering on the parameter space.

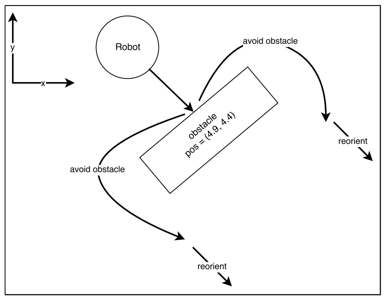


Figure 11: Cartoon illustration of the simulation task, where a robot bypasses an obstacle, and then reorients.

To illustrate, we focus on two tests centered around a simulated robot interacting with an obstacle. The robot is expected to (i) collide with the obstacle, (ii) bypass the obstacle, (iii) reorient to it's pre-collision orientation, and (iv) continue moving in the pre-collision orientation (see Fig 11). We use the following two PSTL templates (derived from [11]) to characterize the obstacle avoidance and re-orientation phase respectively. Crucially, when instantiated with particular parameters, these templates are essentially the actual specification of the robot.

$$\varphi_{avoid}(\tau_1, h_1) = \mathbf{F}_{[0,\tau_1]}(pos.y < h_1) \tag{16}$$

$$\varphi_{reorient}(\tau_2, h_2) = \mathbf{G}_{[0,60-\tau_2]}(pos.x \le h_2) \tag{17}$$

Our goal is to develop a distance metric, where similar solutions and failure modes are grouped together.

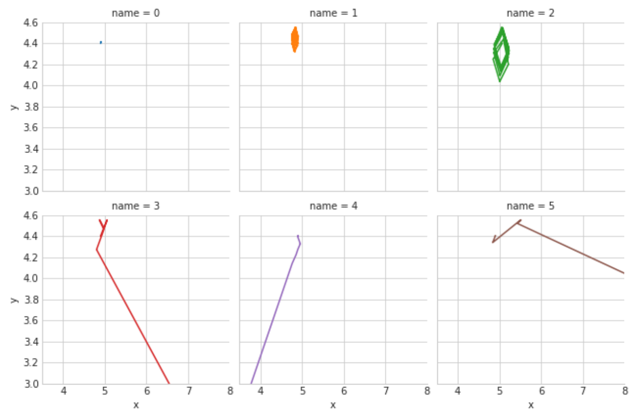


Figure 12: Representative sample of student submissions for obstacle avoidance task. A correct submission will tend towards the bottom right corner.

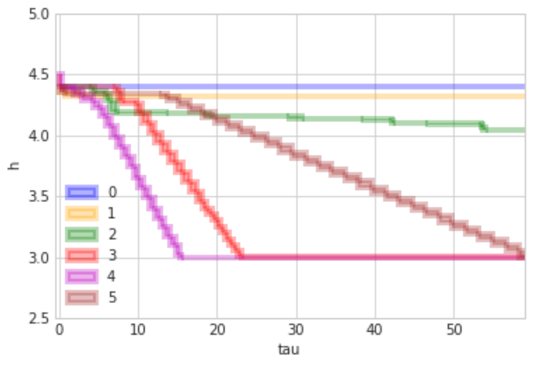


Figure 13: φ_{avoid} boundaries.

Consider the representative sample of student submissions show in Fig 12. Signal 0 shows a submission that never moves. Signal 1 shows a submission that continually collides with the obstacle. Signal 2 shows a submission that collides with the obstacle and then slides around the obstacle as it attempts to by bypass it. Signals 3 and 5 show two solutions going around the obstacle in two different ways. Signal 4 shows a submission that bypassed the obstacle, but failed to reorient.

The validity domain boundaries for φ_{avoid} and $\varphi_{reorient}$ are shown in Fig 13 and Fig 14 respectively. Observe that through the lens of φ_{avoid}, the signals that avoid the obstacle (3, 4, and 5) are correctly separated from signals the signals that do not (0, 1, 2). Similarly, through the lens of the reorient template, the signals that reorient (3,5) are separated from the signals that do not reorient (0,1,2,4). We averaged the logical distance (eq (9)) for both templates yielding the adjacency matrix and dendogram shown in Fig 15. Note that the first layer of the dendogram separates solutions that correctly avoided the obstacle. Within the avoided obstacle group, signals 3 and 5 are correctly grouped together since they both reoriented.

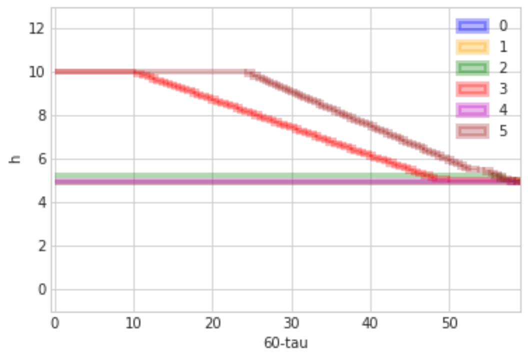
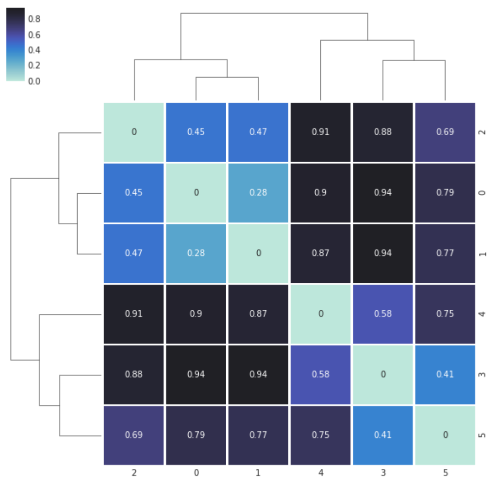


Figure 14: $\varphi_{reorient}$ boundaries.



	2	0	1	4	3	5
2	0	0.45	0.47	0.91	0.88	0.69
0	0.45	0	0.28	0.9	0.94	0.79
1	0.47	0.28	0	0.87	0.94	0.77
4	0.91	0.9	0.87	0	0.58	0.75
3	0.88	0.94	0.94	0.58	0	0.41
5	0.69	0.79	0.77	0.75	0.41	0

Figure 15: Resulting pairwise distances and dendogram of the averaged logical distances of φ_{avoid} and $\varphi_{reorient}$.

5 RELATED WORK

Time-series clustering and classification is a well-studied area in the domain of machine learning and data mining [14]. Time series clustering that work with raw time-series data combine clustering schemes such as agglomerative clustering, hierarchical clustering, k-means clustering among others, with similarity measures between time-series data such as the dynamic time-warping (DTW) distance, statistical measures and information-theoretic measures. Feature-extraction based methods typically use generic sets of features, but algorithmic selection of the right set of meaningful features is a challenge. Finally, there are model-based approaches that seek an underlying generative model for the time-series data, and typically require extra assumptions on the data such as linearity or the Markovian property. Please see [14] for detailed references to each approach. It should be noted that historically time-series learning focused on univariate time-series, and extensions to multivariate time-series data have been relatively recent developments.

More recent work has focused on automatically identifying features from the data itself, such as the work on *shapelets* [15, 20, 24], where instead of comparing entire time-series data using similarity measures, algorithms to automatically identify distinguishing motifs in the data have been developed. These motifs or shapelets serve not only as features for ML tasks, but also provide visual feedback to the user explaining why a classification or clustering task labels given data in a certain way. While we draw inspiration from this general idea, we seek to expand it to consider logical shapes in the data, which would allow leveraging user's domain expertise.

Automatic identification of motifs or basis functions from the data while useful in several documented case studies, comes with some limitations. For example, in [2], the authors define a subspace clustering algorithm, where given a set of time-series curves, the algorithm identifies a subspace among the curves such that every curve in the given set can be expressed as a linear combination of a deformations of the curves in the subspace. We note that the authors observe that it may be difficult to associate the natural clustering structure with specific predicates over the data (such as patient outcome in a hospital setting).

The use of logical formulas for learning properties of time-series has slowly been gaining momentum in communities outside of traditional machine learning and data mining [3, 4, 10, 13]. Here, fragments of Signal Temporal Logic have been used to perform tasks such as supervised and unsupervised learning. A key distinction from these approaches is our use of libraries of signal predicates that encode domain expertise that allow human-interpretable clusters and classifiers.

Finally, preliminary exploration of this idea appeared in prior work by some of the co-authors in [23]. The key difference is the previous work required users to provide a ranking of parameters appearing in a signal predicate, in order to project time-series data to unique points in the parameter space. We remove this additional burden on the user in this paper by proposing a generalization that projects time-series signals to trade-off curves in the parameter space, and then using these curves as features.

6 CONCLUSION AND FUTURE WORK

We proposed a new paradigm for time-series learning centered around using the validity domain boundaries of *monotonic parameterized specifications* to induce distance measures that respect the logical characteristic of the specification. A unique feature of this approach is that, a simple Boolean predicate based on the monotonic specification can be used to explain why any two traces (or sets of traces) have a given distance. We concluded by demonstrating our technique with two case studies that illustrate how simple monotonic specifications can be used to craft desirable distance measures. Future work includes applying these techniques to larger data sets, investigating how to the leverage massively parallel natural in the boundary and Hausdorff computation using graphical processing units, and characterizing alternative boundary distances such as the average distance.

REFERENCES

[1] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. 2011. Parametric identification of temporal properties. In *Proc. of RV*. 147–160.

[2] Mohammad T Bahadori, David Kale, Yingying Fan, and Yan Liu. 2015. Functional Subspace Clustering with Application to Time Series. In *Proc. of ICML*. 228–237.

[3] Ezio Bartocci, Luca Bortolussi, and Guido Sanguinetti. 2014. Data-driven statistical learning of temporal logic properties. In *International conference on formal modeling and analysis of timed systems*. Springer, 23–37.

[4] Giuseppe Bombara, Cristian-Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. 2016. A Decision Tree Approach to Data Classification using Signal Temporal Logic. In *Proc. of HSCC*. 1–10.

[5] J Colyar and J Halkias. 2007. US highway 101 dataset. *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030* (2007).

[6] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, Linhong Zhu, Rose Yu, and Yan Liu. 2016. Latent space model for road networks to predict time-varying traffic. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1525–1534.

[7] J. Kapinski et al. 2016. ST-Lib: A Library for Specifying and Classifying Model Behaviors. In *SAE Technical Paper*. SAE.

[8] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy V Deshmukh, and Sanjit A Seshia. 2015. Mining requirements from closed-loop control models. *IEEE TCAD of ICS* 34, 11 (2015), 1704–1717.

[9] Austin Jones, Derya Aksaray, Zhaodan Kong, Mac Schwager, and Calin Belta. 2015. Robust Satisfaction of Temporal Logic Specifications via Reinforcement Learning. *CoRR* abs/1510.06460 (2015). http://arxiv.org/abs/1510.06460

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

[11] Garvit Juniwal, Alexandre Donzé, Jeff C Jensen, and Sanjit A Seshia. 2014. CPS-Grader: Synthesizing temporal logic testers for auto-grading an embedded systems laboratory. In *Proc. of EMSOFT*. 24.

[12] David C Kale, Dian Gong, Zhengping Che, Yan Liu, Gerard Medioni, Randall Wetzel, and Patrick Ross. 2014. An Examination of Multivariate Time Series Hashing with Applications to Health Care. In *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 260–269.

[13] Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. 2014. Temporal logic inference for classification and prediction from data. In *Proc. of HSCC*. 273–282.

[14] T Warren Liao. 2005. Clustering of time series data survey. *Pattern recognition* 38, 11 (2005), 1857–1874.

[15] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. 2012. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 289–297.

[16] Yan Liu, Taha Bahadori, and Hongfei Li. 2012. Sparse-GEV: Sparse Latent Space Model for Multivariate Extreme Value Time Series Modeling. In *Proc. of ICML*.

[17] Oded Maler. 2017. Learning Monotone Partitions of Partially-Ordered Domains (Work in Progress). (July 2017). https://hal.archives-ouvertes.fr/hal-01556243 working paper or preprint.

[18] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Proc. of FORMATS/FTRTFT*. 152–166.

[19] Joel C McCall and Mohan M Trivedi. 2007. Driver behavior and situation aware brake assistance for intelligent vehicles. *Proc. IEEE* 95, 2 (2007), 374–387.

[20] Abdullah Mueen, Eamonn Keogh, and Neal Young. 2011. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1154–1162.

[21] Abdel Aziz Taha and Allan Hanbury. 2015. An efficient algorithm for calculating the exact Hausdorff distance. *IEEE transactions on pattern analysis and machine intelligence* 37, 11 (2015), 2153–2163.

[22] Marcell Vazquez-Chanlatte. 2018. Multidimensional Thresholds. https://github.com/mvcisback/multidim-threshold/. (2018).

[23] Marcell Vazquez-Chanlatte, Jyotirmoy V. Deshmukh, Xiaoqing Jin, and Sanjit A. Seshia. 2017. Logic-based Clustering and Learning for Time-Series Data. In *Proc. International Conference on Computer-Aided Verification (CAV)*.

[24] Lexiang Ye and Eamonn Keogh. 2009. Time series shapelets: a new primitive for data mining. In *In Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 947–956.