Mining explainable temporal specifications from data

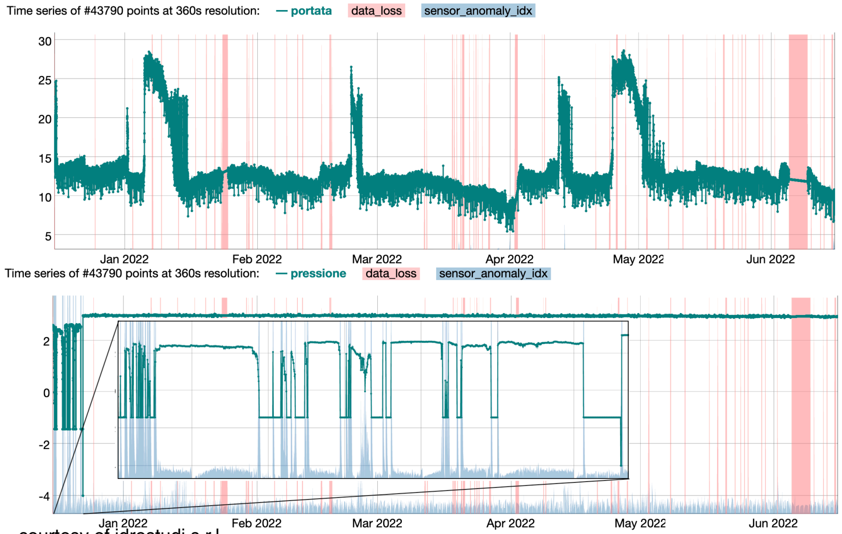
from syntax to semantics and back

Luca Bortolussi
University of Trieste, Italy

FBK, Digital Industry Center Seminar

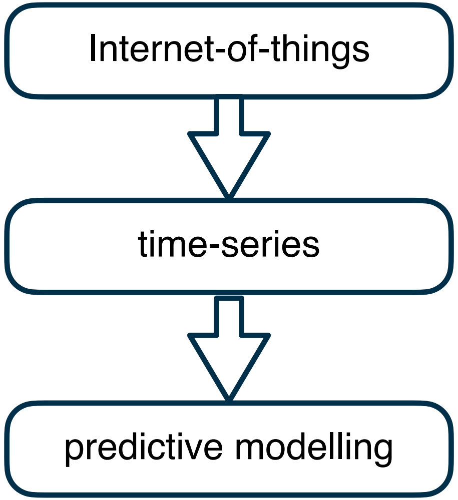
Trento, December 19, 2022

Context and Problem



courtesy of idrostudi s.r.l.

flow rate and pressure in a water network



(e.g. anomaly detection)

Need of human-interpretable models

Menu of the Day

Starter: STL requirement mining

Main: semantic-preserving STL embeddings

Dessert: ongoing work salad

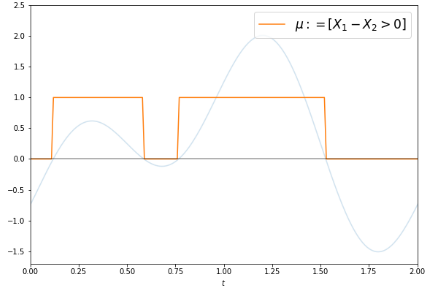
Alka seltzer?

Signal Temporal Logic (STL)

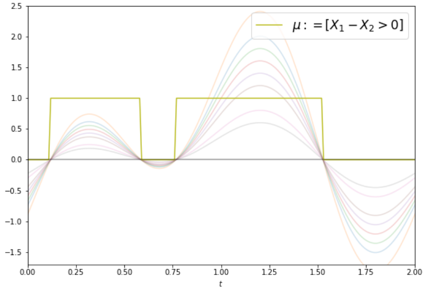
STL is a linear-time temporal logic suitable to specify property over continuous trajectories.

Syntax: $\varphi := tt | f(x) \geq 0 | \neg\varphi | \varphi_1 \wedge \varphi_2 | \varphi_1 \mathbf{U}_{[a,b]} \varphi_2$

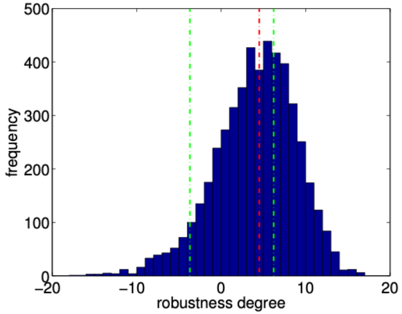
Extinction of an epidemics between 100 and 120 days from onset: $X_{inf} > 0 \; U_{[100,120]} \; X_{inf} = 0$



Boolean semantics: $\chi(r)$



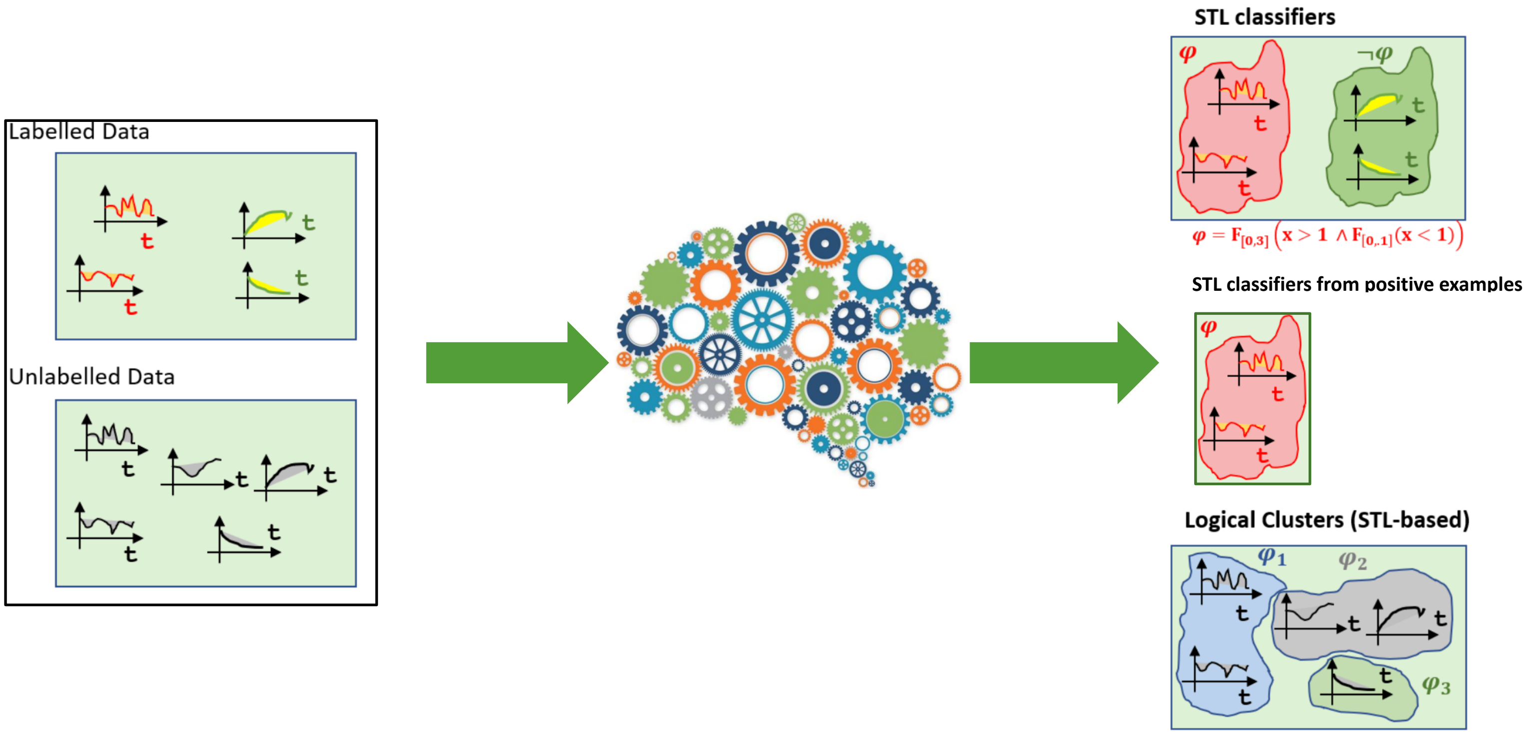
Quantitative semantics: $\rho(r)$



Satisfaction probability: $\mathbb{E}_{\mu(r)}[\chi(r)]$

Expected robustness: $\mathbb{E}_{\mu(r)}[\rho(r)]$

STL requirement mining



Source: https://jdeshmukh.github.io/research.html

STL classifier: ROGE

Learn an STL-classifier separating *good* from *bad* trajectories

Hybrid Genetic Algorithm:

- GA to explore STL syntactic tree space
- Bayesian Optimization for optimising formula parameters

$$G(\phi) = \frac{\mathbb{E}(R_\phi|\vec{X}_p) - \mathbb{E}(R_\phi|\vec{X}_n)}{\sigma(R_\phi|\vec{X}_p) + \sigma(R_\phi|\vec{X}_n)}.$$

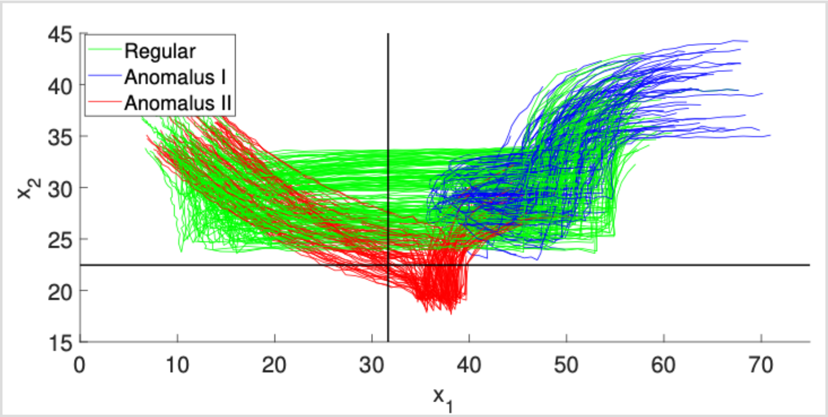
Require: $\mathcal{D}_p, \mathcal{D}_n, \mathbb{K}, Ne, Ng, \alpha, s$

1: $gen \leftarrow$ GENERATEINITIALFORMULAE(Ne, s)
2: $gen_\Theta \leftarrow$ LEARNINGPARAMETERS$(gen, G, \mathbb{K})$
3: **for** $i = 1 \dots Ng$ **do**
4: $\quad subg_\Theta \leftarrow$ SAMPLE(gen_Θ, F)
5: $\quad newg \leftarrow$ EVOLVE$(subg_\Theta, \alpha)$
6: $\quad newg_\Theta \leftarrow$ LEARNINGPARAMETERS$(newg, G, \mathbb{K})$
7: $\quad gen_\Theta \leftarrow$ SAMPLE$(newg_\Theta \cup gen_\Theta, F)$
8: **end for**
9: **return** gen_Θ

L. Nenzi, S. Silvetti, E. Bartocci, L. Bortolussi: A Robust Genetic Algorithm for Learning Temporal Specifications from Data. QEST 2018

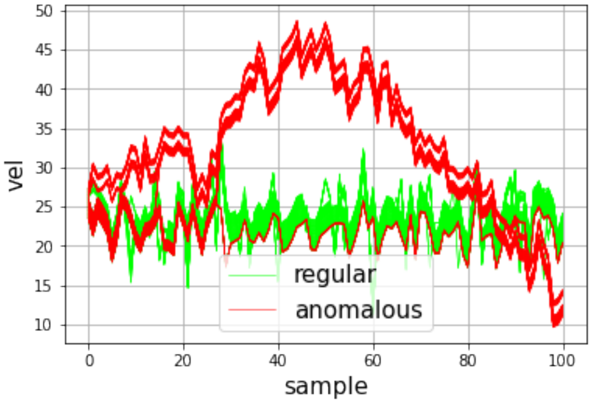
ROGE: results

Maritime surveillance



$$((x_2 > 22.46)\,\mathcal{U}_{[49,287]}\,(x_1 \leq 31.65))$$

Train cruise

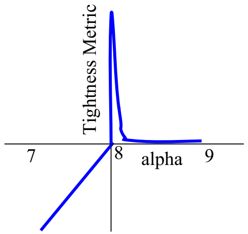


$$(\mathrm{F}_{[22,40]}(\mathtt{vel} > 24.48)) \wedge (\mathrm{F}_{[46,49]}(19.00 < \mathtt{vel} < 26.44))$$

Mining requirements from positive examples

Optimization Algorithm:

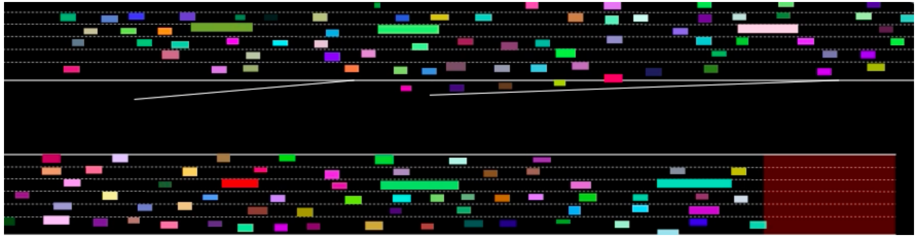
- GA to explore STL syntactic tree space
- Score function modelling the tightness of STL properties (positive robustness close to zero)



$$f(\varphi; X^+_{\mathcal{L}}) = \alpha \frac{1}{\left|X^+_{\mathcal{L}}\right|} \left|\{\boldsymbol{x} \in X^+_{\mathcal{L}} : \boldsymbol{x} \not\models \varphi\}\right| + \frac{1}{\sigma'_{\varphi, X^+_{\mathcal{L}}} \left|X^+_{\mathcal{L}}\right|} \sum_{\boldsymbol{x} \in X^+_{\mathcal{L}}} |\rho(\varphi, \boldsymbol{x})|$$

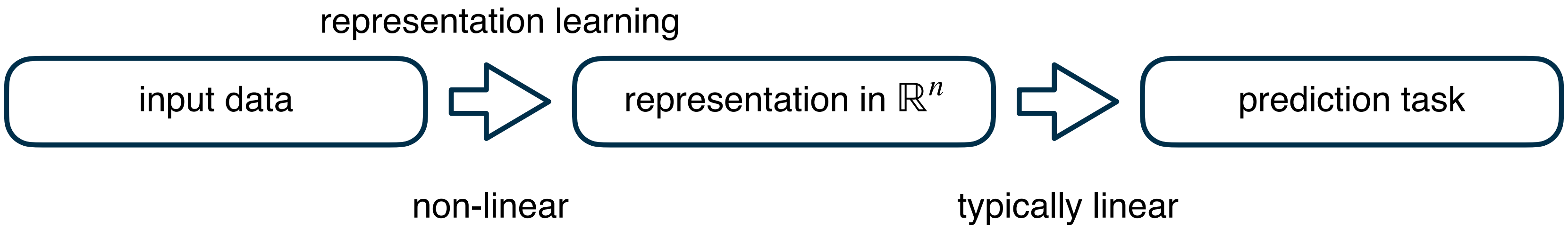
$$\sigma'_{\varphi, X} = \sqrt{\frac{1}{|X|} \sum_{\boldsymbol{x} \in X} \left(|\rho(\varphi, \boldsymbol{x})| - \frac{1}{|X|} \sum_{\boldsymbol{x} \in X} |\rho(\varphi, \boldsymbol{x})| \right)^2}$$

Learning STL traffic rules



S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, N. Shankar. TeLEx: learning signal temporal logic from positive examples using tightness metric, Formal Methods in System Design
F. Pigozzi, E. Medvet, L. Nenzi. Mining Road Traffic Rules with Signal Temporal Logic and Grammar-Based Genetic Programming, Applied Sciences, 2022

A modern machine learning approach



Goal: embed STL formulae in $\mathbb{R}^n$ meaningfully.

Ideally: distance between embedded formulae should reflect semantic distance.

Main: semantic-preserving embeddings

How to construct meaningful embeddings?

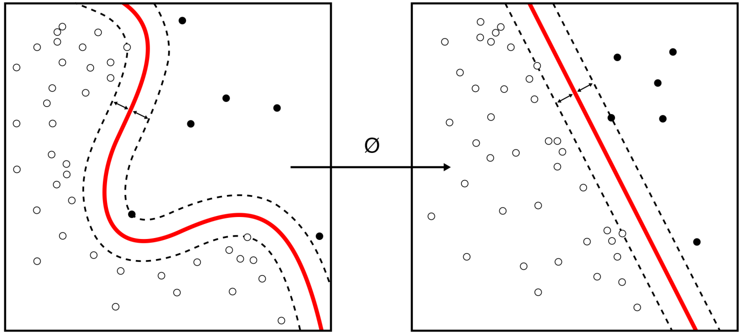
kernel-based methods

How to check that they are meaningful?

learning model checking

Bortolussi, L., Gallo, G. M., Křetínský, J., & Nenzi, L. *Learning model checking and the kernel trick for signal temporal logic on stochastic processes.* In: TACAS, 2022.

Kernels



Kernel’s application to linearize a problem

A *kernel* is a function *k* defining implicitly a scalar product in a feature space

$$k(x, y) =< \phi(x), \phi(z) > \forall x, z \in X$$

where ϕ is a map from X to the feature space

Kernel Trick

A linear regression problem in the feature space $\phi(X)$: $\sum_j w_j \phi_j(x)$

has a dual formulation depending on N dual variables α and on the kernel evaluated among training points $k(x_i, x_j)$.

Overview: kernel trick for STL

1. How to embed formulae in a Hilbert space?

   identify a formula with a functional via quantitative semantics: $\varphi : \mathscr{T} \to \mathbb{R}$

2. How to measure similarity on the feature representation?

   use scalar product in L_2 w.r.t. a base finite measure μ_0

3. How to design a finite measure on trajectories?

   prefer simple trajectories with limited variation

A kernel for STL

Computing kernels in three steps:

integration w.r.t. a base measure μ_0

$$k'(\varphi, \psi) = \int_{r \in \mathcal{T}} \varphi(r)\psi(r) d\mu_0(r)$$

normalisation

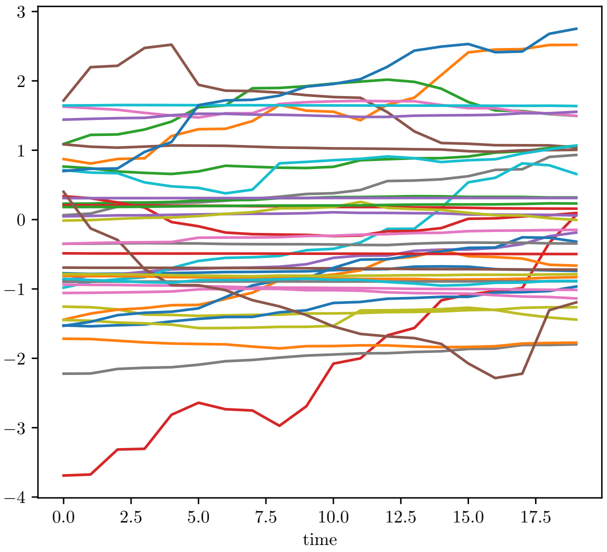
$$k_0(\varphi, \psi) = \frac{k'(\varphi, \psi)}{\sqrt{k'(\varphi, \varphi)k'(\psi, \psi)}}$$

exponentiation

$$k(\varphi, \psi) = \exp\left(-\frac{1 - 2k_0(\varphi, \psi)}{\sigma^2}\right)$$

The base measure μ_0

Compute integral by Montecarlo sampling of μ_0: $$k'(\varphi, \psi) \approx \frac{1}{M} \sum_{i=1}^{M} \varphi(r_i)\psi(r_i)$$



μ_0 is defined via its sampling algorithm:

- fixed time step Δ up to a final time T
- Bounded total variation (sampled from squared Gaussian)
- Limited change of sign of derivative

"Learning" model checking

Equipped with the previous definitions, we can try to solve the following problem:

Given $p(\psi_j \mid M)$ for **randomly** chosen formulae $\psi_1, \ldots, \psi_n$

can we predict $p(\varphi \mid M)$?

without knowing or executing the system M

Learning with STL kernels

Different kinds of prediction tasks:

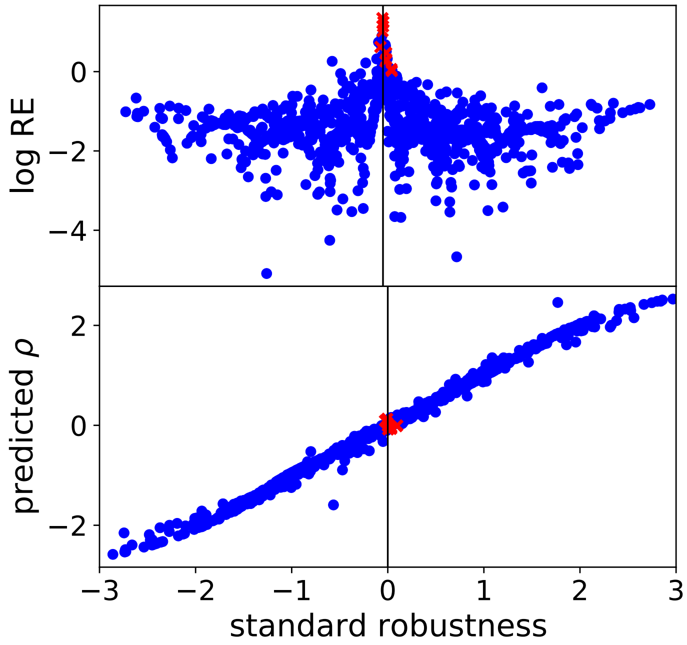
- Boolean truth and robustness for individual trajectories
- average robustness (w.r.t. μ_0 or a generic process μ)
- satisfaction probability (w.r.t. μ_0 or a generic process μ)

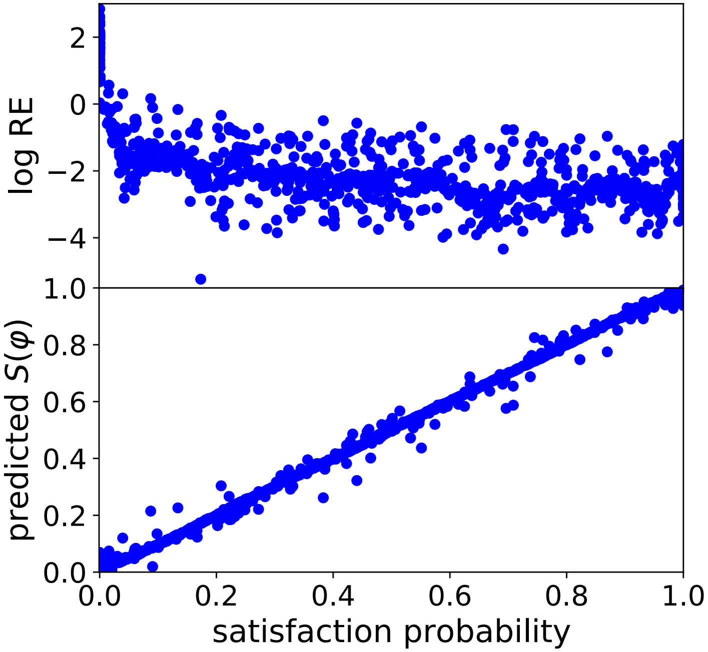
Data distribution over STL formulae φ: prefer simple formulae over complex ones

Training set: $\{(\psi_j, y_j)\}_{j=1\ldots,n}$

Learning algorithm: kernel ridge regression (with cross-validation)

Experimental Results

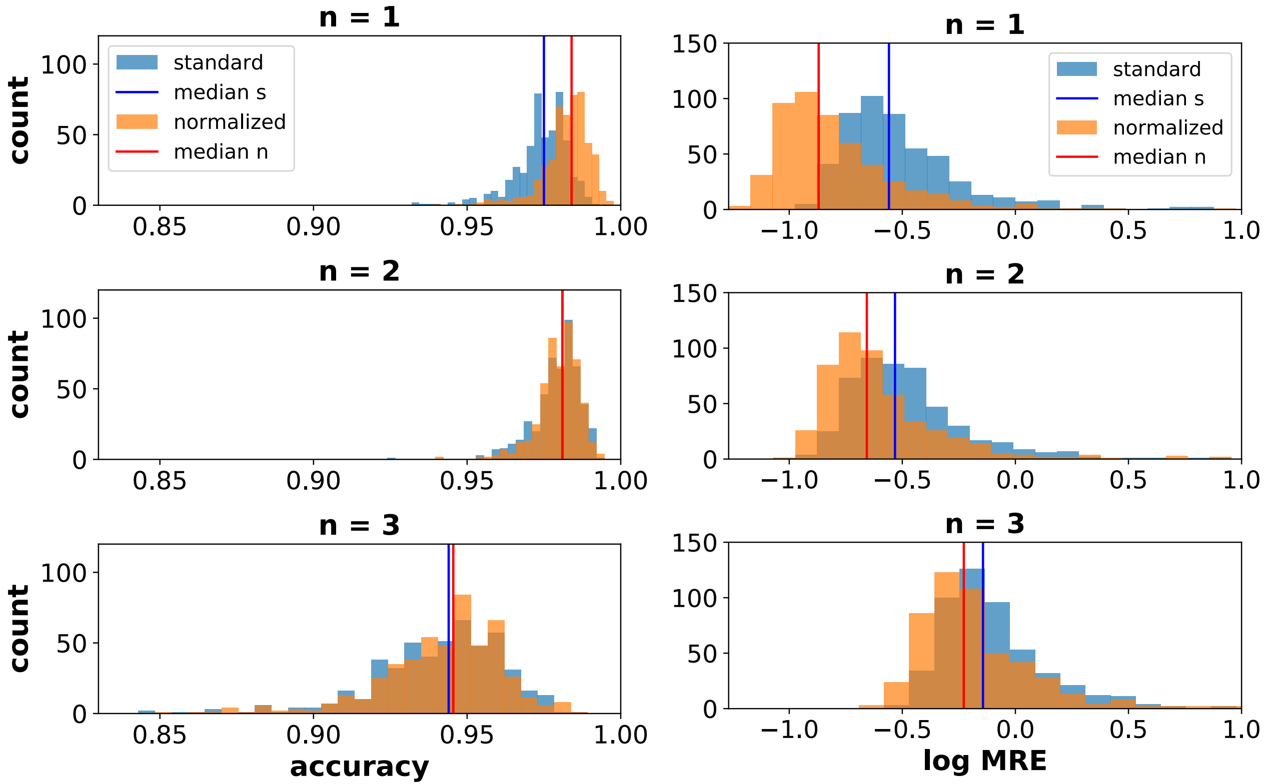




(left) Robustness on single trajectories and (right) satisfaction probability (μ_0)

Good generalisation on out-of-distribution formulae

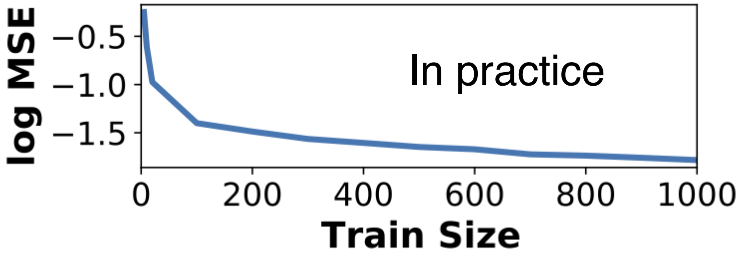
Experimental Results on the stochastic models



(left) Accuracy of satisfiability prediction and (right) MRE of robustness prediction

Immigration (1d)
Isomerization (2d)
Transcription (3d)

How many input points we need?



PAC bounds for 0-1 loss:

$$L(h) \leq \hat{L}_D(h) + \frac{\Lambda}{\sqrt{m}} + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}.$$

Λ: maximum norm of regression functions; δ: error probability; m: dataset size;

$$L(h) = \mathbb{E}_{\varphi \sim p_{data}}\left[\mathbb{I}\big(h(\varphi) \neq y(\varphi)\big)\right]; \quad \hat{L}_D(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}\big(h(\varphi_i) \neq y(\varphi_i)\big)$$

Dessert: ongoing work

How to make embeddings explicit (i.e. in $\mathbb{R}^k$)?

kernel PCA

Can we replace quantitative with Boolean semantics?

Boolean kernel

How to use these embeddings for STL requirement mining?

invert the embeddings using GNN

From implicit to explicit embeddings

input data → representation in $\mathbb{R}^\infty$ → representation in $\mathbb{R}^k$ → prediction task

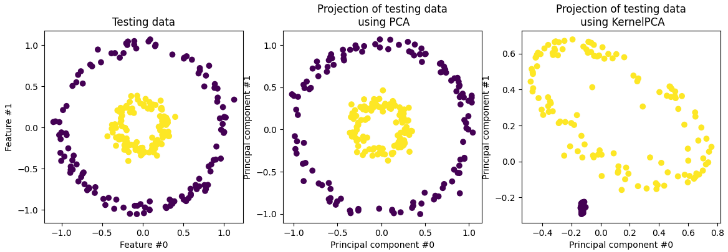
STL kernel

dimensionality reduction $k << n < \infty$

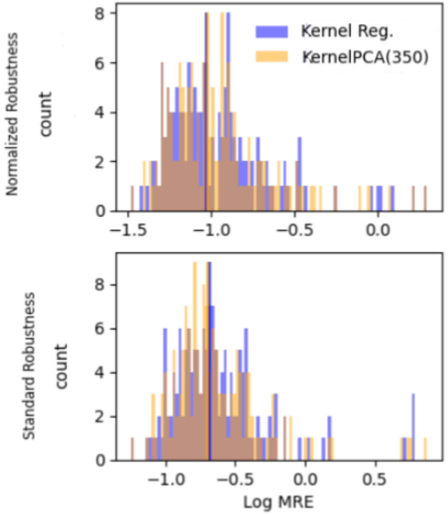
Goal: reduce the dimensionality of the embeddings using Kernel-PCA

Kernel-PCA

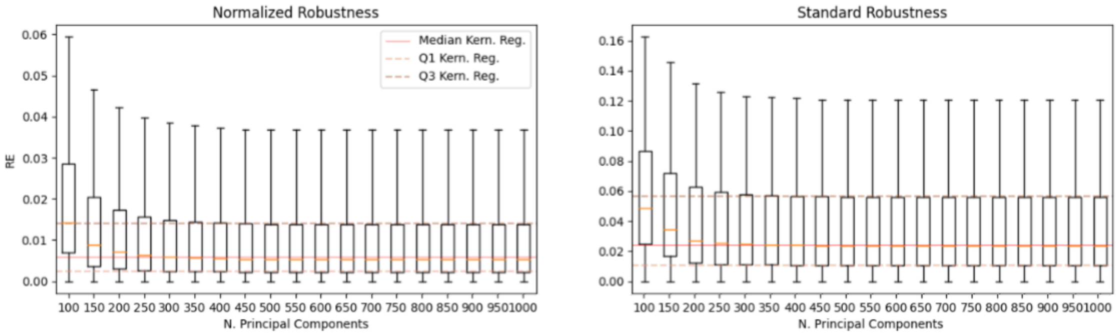
Project input data on a high-dimensional continuous space $\mathbb{R}^n$ using a kernel, then perform dimensionality reduction using PCA to project the embeddings in $\mathbb{R}^k$, where downstream tasks are performed.



Kernel-PCA: experimental results



MRE Comparison of STL Kernel Regression with $n = 1000$ and Kernel PCA + linear regression with $k = 350$.



After ~ 350 principal components, the performance of Kernel PCA stabilises to errors comparable to that of STL Regression.

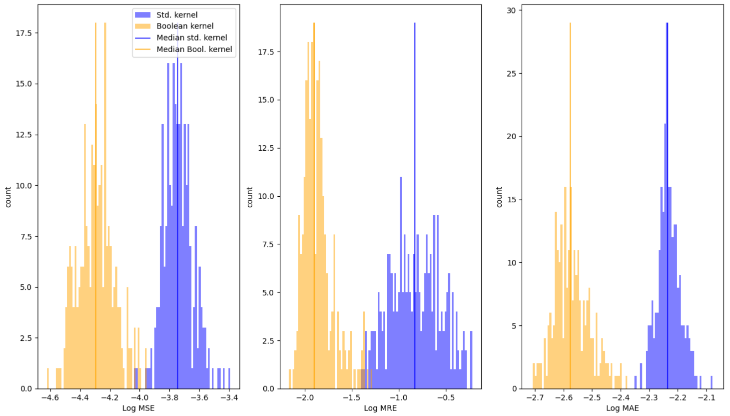
Intuition: many of the formulae in the training set bring the same contribution to the final predictions, without adding a significant amount of information. Reducing the dimension of the embeddings saves computational time without hurting the predictive performance.

A STL-kernel leveraging qualitative satisfaction

Adapt the definition of the STL Kernel to rely on the qualitative/Boolean semantics of STL

$$k'_b(\varphi, \psi) = \int_{r \in \mathcal{T}} \bar{\varphi}(r)\bar{\psi}(r)\mathrm{d}\mu_0(r)$$

i.e. integral of the product of the satisfiability value of input formulae w.r.t. measure μ_0.



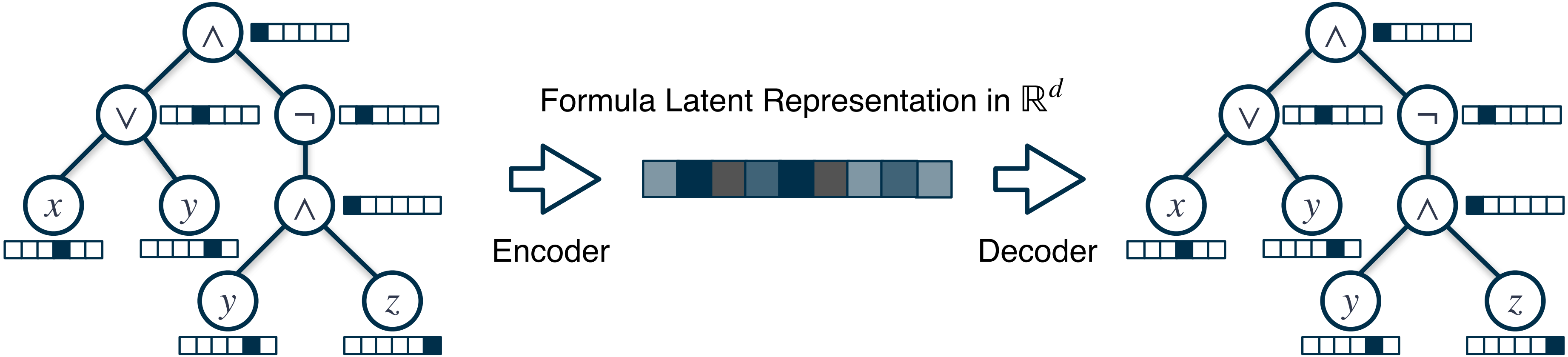
Satisfaction probability

Advantages:

- the Boolean kernel preserves semantic equivalence
- the Boolean kernel outperforms the standard one on the task of satisfaction probability;
- interpretable measure of similarity between STL formulae (allowing to sample formulae as diverse as possible).

Inverting the embedding

Problem with kernel embeddings: non-invertibility → **encoding-decoding architecture**



Learn invertible encodings using Graph Neural Networks (GNN):

- Encode parse tree of the formula into the latent space
- Decode latent vectors to syntactic trees, ideally with the same semantic meaning of the input formula

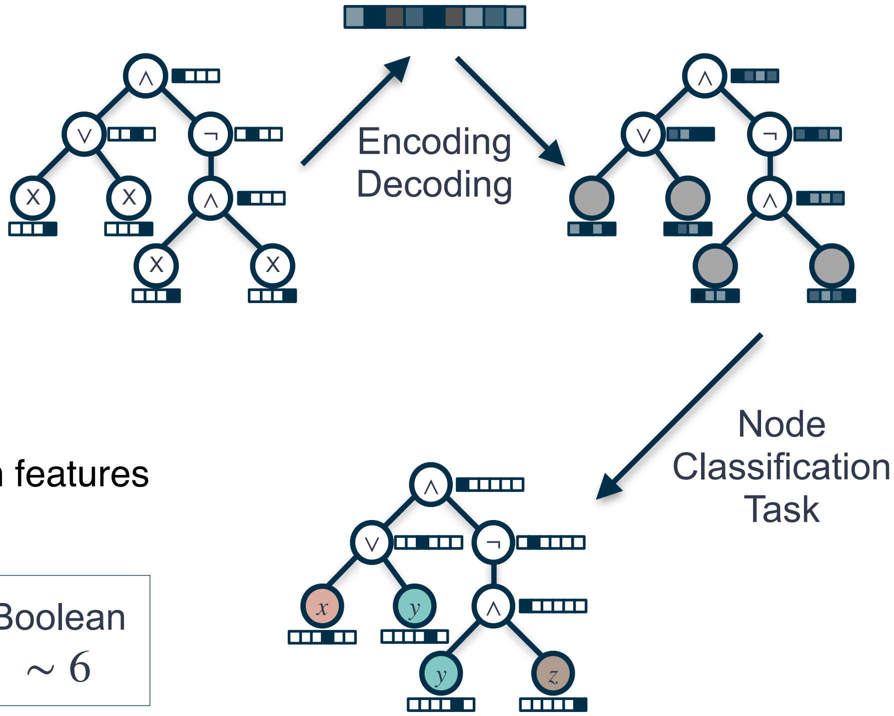
A simpler setting: boolean formulae

Problems with GNN encoding-decoding architectures:

- Scalability to deeper parse trees
- Learning temporal/threshold parameters of operators

Current solutions/attempts:

- Boolean logic setting (i.e. non-parametric formulae)
- Hierarchical approach: first learn adjacency matrix, then features



Currently $92 \pm 3\ \%$ average reconstruction accuracy on Boolean formulae with 5 variables and parse trees having depth ~ 6

Conclusions

- Using kernels + kernel PCA, we can construct finite dimensional embeddings which are effective in solving the “learning” model checking problem.
- Leveraging GNN deep learning models we are trying to build syntax based invertible embeddings.
- Idea: combine syntax and semantic based embeddings to get invertible mappings from formulae to real vector spaces
- use the framework for STL requirement mining, formula translation, sanitisation and simplification, game-based synthesis, …

Acknowledgements and References

Jan Kretinski

Laura Nenzi

Gaia Saveri

Houssam Abbas

Bortolussi, L., Gallo, G. M., Křetínský, J., & Nenzi, L. *Learning model checking and the kernel trick for signal temporal logic on stochastic processes.* In: TACAS, 2022.