Bridging LTL$_f$ Inference to GNN Inference for Learning LTL$_f$ Formulae

Weilin Luo[1], Pingjia Liang [1], Jianfeng Du [2,3,*], Hai Wan [1,*],
Bo Peng [1] Delong Zhang [1]

[1] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2] Guangzhou Key Laboratory of Multilingual Intelligent Processing,
Guangdong University of Foreign Studies, Guangzhou, China
[3] Pazhou Lab, Guangzhou, China

Content

Content

Definition of Problem

Learning formulae to characterize the high-level behavior of a system from observation traces.

- focus on the *linear temporal logic on finite traces* (LTL_f) formula
- arbitrary form
- noisy data

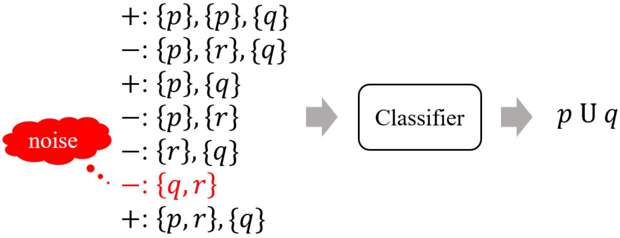

Figure 1: Learning LTL_f formulae from noisy data.

Significance and Challenge

Wide applications:

1. verification of system properties [3]
2. behavior classification [1]
3. explainable models [4]

Significance and Challenge

Wide applications:

1. verification of system properties [3]
2. behavior classification [1]
3. explainable models [4]

Challenging task:

1. *huge search space* of the target formula in arbitrary form
2. *wrong search bias* resulting from noisy data

State-of-the-art Approach

State-of-the-art (SOTA) approach to learn LTL_f formulae:

- SAT-based[1,5]
- based on bayesian inference[4]

They either assume a noise-free environment or restrict the hypothesis space by LTL_f templates.

State-of-the-art Approach

State-of-the-art (SOTA) approach to learn LTL_f formulae:

- SAT-based [1,5]
- based on bayesian inference [4]

They either assume a noise-free environment or restrict the hypothesis space by LTL_f templates.

Gaglione et al. (2021) [2]:

- MaxSAT-based approach

The scalability of them is limited in calling the MaxSAT solver.

State-of-the-art Approach

State-of-the-art (SOTA) approach to learn LTL_f formulae:

- SAT-based [1,5]
- based on bayesian inference [4]

They either assume a noise-free environment or restrict the hypothesis space by LTL_f templates.

Gaglione et al. (2021) [2]:

- MaxSAT-based approach

The scalability of them is limited in calling the MaxSAT solver.

Developing new approaches based on *neural networks* to learn arbitrary LTL_f formulae from noisy data.

Content

LTL$_f$ Graph

Definition 1 (LTL$_f$ Graph (simplified))

Let ϕ be an LTL$_f$ formula. Its *LTL$_f$ graph* G_ϕ is a four-tuple $(V_\phi, E_\phi, \mathsf{W}_\phi, \mathsf{b}_\phi)$ defined as follows,

- if unfold$(\phi_i) = p$, then $V_\phi = V_\phi \cup \{v_p\}$ and $\mathsf{b}_\phi(v_p) = 0$;
- if unfold$(\phi_i) = \neg\phi_j$, then $V_\phi = V_\phi \cup \{v_{\phi_j}\}$, $E_\phi = E_\phi \cup \{\langle v_{\phi_j}, v_{\phi_i}\rangle\}$, $\mathsf{W}_\phi(\langle v_{\phi_j}, v_{\phi_i}\rangle) = -1$, and $\mathsf{b}_\phi(v_{\phi_i}) = 1$;
- if unfold$(\phi_i) = \phi_j \wedge \phi_k$, then $V_\phi = V_\phi \cup \{v_{\phi_j}, v_{\phi_k}\}$, $E_\phi = E_\phi \cup \{\langle v_{\phi_j}, v_{\phi_i}\rangle, \langle v_{\phi_k}, v_{\phi_i}\rangle\}$, $\mathsf{W}_\phi(\langle v_{\phi_j}, v_{\phi_i}\rangle) = 1$, $\mathsf{W}_\phi(\langle v_{\phi_k}, v_{\phi_i}\rangle) = 1$, and $\mathsf{b}_\phi(v_{\phi_i}) = -1$;
- if unfold$(\phi_i) = \mathsf{X}\phi_j$, then $V_\phi = V_\phi \cup \{v_{\mathsf{X}\phi_j}\}$, $E_\phi = E_\phi \cup \{\langle v_{\mathsf{X}\phi_j}, v_{\phi_i}\rangle\}$, $\mathsf{W}_\phi(\langle v_{\mathsf{X}\phi_j}, v_{\phi_i}\rangle) = 1$, and $\mathsf{b}_\phi(v_{\phi_i}) = \mathsf{b}_\phi(v_{\mathsf{X}\phi_j}) = 0$;
- if unfold$(\phi_i) = \phi_k \vee (\phi_j \wedge \mathsf{X}\phi_i)$, then $V_\phi = V_\phi \cup \{v_{\phi_k}, v_{\phi_j}, v_{\mathsf{X}\phi_i}\}$, $E_\phi = E_\phi \cup \{\langle v_{\phi_k}, v_{\phi_i}\rangle, \langle v_{\phi_j}, v_{\phi_i}\rangle, \langle v_{\mathsf{X}\phi_i}, v_{\phi_i}\rangle\}$, $\mathsf{W}_\phi(\langle v_{\phi_k}, v_{\phi_i}\rangle) = 2$, $\mathsf{W}_\phi(\langle v_{\phi_j}, v_{\phi_i}\rangle) = 1$, $\mathsf{W}_\phi(\langle v_{\mathsf{X}\phi_i}, v_{\phi_i}\rangle) = 1$, $\mathsf{b}_\phi(v_{\phi_i}) = -1$, and $\mathsf{b}_\phi(v_{\mathsf{X}\phi_i}) = 0$,

where $p \in \mathbb{P} \cup \{\top, \bot\}$ and ϕ_j, ϕ_k are LTL$_f$ formulae.

LTL_f Graph

Example 1

Let $\phi = p\mathsf{U}(\mathsf{X}\neg q)$ be an LTL_f formula. $\mathsf{sub}(\phi) = \{p, q, \phi_3, \phi_4, \phi_5\}$, where $\phi_5 = p\mathsf{U}(\mathsf{X}\neg q)$, $\phi_4 = \mathsf{X}\neg q$, and $\phi_3 = \neg q$. The LTL_f graph of ϕ is illustrated as follows.

- $\mathsf{W}_\phi(\langle v_{\mathsf{X}\phi_5}, v_{\phi_5}\rangle) = 1$
- $\mathsf{W}_\phi(\langle v_{\phi_4}, v_{\phi_5}\rangle) = 2$
- $\mathsf{W}_\phi(\langle v_p, v_{\phi_5}\rangle) = 1$
- $\mathsf{W}_\phi(\langle v_{\mathsf{X}\phi_3}, v_{\phi_4}\rangle) = 1$
- $\mathsf{W}_\phi(\langle v_q, v_{\phi_3}\rangle) = -1$
- $\mathsf{b}_\phi(v_{\phi_5}) = -1$
- $\mathsf{b}_\phi(v_{\phi_3}) = 1$
- $\mathsf{b}_\phi(v_{\phi_4}) = \mathsf{b}_\phi(v_p) = \mathsf{b}_\phi(v_q) = \mathsf{b}_\phi(v_{\mathsf{X}\phi_3}) = \mathsf{b}_\phi(v_{\mathsf{X}\phi_5}) = 0$

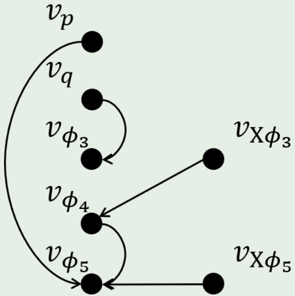

Figure 2: The LTL_f graph of ϕ.

LTL$_f$ Graph

Example 2 (Example 1 cont.)

Let $\phi = p\mathsf{U}(\mathsf{X}\neg q)$ be an LTL$_f$ formula. sub$(\phi) = \{p, q, \phi_3, \phi_4, \phi_5\}$, where $\phi_5 = p\mathsf{U}(\mathsf{X}\neg q)$, $\phi_4 = \mathsf{X}\neg q$, and $\phi_3 = \neg q$.

- $\phi_5 = p\mathsf{U}\phi_4 = \phi_4 \vee (p \wedge \mathsf{X}\phi_5)$.
- Relation of satisfaction of the sub-formulae at the current state (p and ϕ_4) and the next state ($\mathsf{X}\phi_5$):
  - if p is false, ϕ_4 is false, and $\mathsf{X}\phi_5$ is false, then ϕ_5 is false;
  - if p is false, ϕ_4 is false, and $\mathsf{X}\phi_5$ is true, then ϕ_5 is false;
  - if p is false, ϕ_4 is true, and $\mathsf{X}\phi_5$ is false, then ϕ_5 is true;
  - ...
- **Boolean** $\rightarrow$ **real**: false $\rightarrow \leq 0$, true $\rightarrow \geq 1$.
- $\mathsf{W}_\phi(\langle v_{\mathsf{X}\phi_5}, v_{\phi_5}\rangle) = 1$, $\mathsf{W}_\phi(\langle v_{\phi_4}, v_{\phi_5}\rangle) = 2$, $\mathsf{W}_\phi(\langle v_p, v_{\phi_5}\rangle) = 1$, and $\mathsf{b}_\phi(v_{\phi_5}) = -1$.

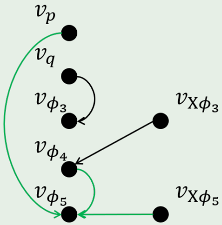

Figure 3: The LTL$_f$ graph of ϕ.

State Classifier

Definition 2 (State Classifier (simplified))

Let ϕ be an LTL$_f$ formula such that $|\mathsf{sub}(\phi)| = L$, $(V_\phi, E_\phi, \mathsf{W}_\phi, \mathsf{b}_\phi)$ an LTL$_f$ graph of ϕ, and $\pi = s_0, s_1, \ldots, s_n$ a trace. $\mathbf{x}_{s_i}^{(0)}$ is defined such that for all $1 \leq j \leq L$, $(\mathbf{x}_{s_i}^{(0)})_j = 1$ if $\phi_j \in s_i$ or $(\mathbf{x}_{s_i}^{(0)})_j = 0$ otherwise. $\mathbf{x}_{s_i}^{(t)}$ is defined recursively as follows:

$$\mathbf{x}_{s_i}^{(t)} = \sigma(\mathbf{C}_\phi \mathbf{x}_{s_i}^{(t-1)} + \mathbf{A}_\phi \mathbf{x}_{s_{i+1}} + \mathbf{b}_\phi), \tag{1}$$

$$(\mathbf{C}_\phi)_{ij} = \begin{cases} \mathsf{W}_\phi(\langle v_{\phi_j}, v_{\phi_i} \rangle), & \text{if } \langle v_{\phi_j}, v_{\phi_i} \rangle \in E_\phi \text{ and} \\ & \phi_j \in \mathsf{sub}(\phi) \\ 0, & \text{otherwise,} \end{cases}$$

$$(\mathbf{A}_\phi)_{ij} = \begin{cases} \mathsf{W}_\phi(\langle v_{\phi_j}, v_{\phi_i} \rangle), & \text{if } \langle v_{\phi_j}, v_{\phi_i} \rangle \in E_\phi \text{ and} \\ & \phi_j \in \{\mathsf{X}\phi_k | \phi_k \in \mathsf{sub}(\phi)\} \\ 0, & \text{otherwise,} \end{cases}$$

$$(\mathbf{b}_\phi)_i = \mathsf{b}_\phi(v_{\phi_i}), \quad \text{for all } v_{\phi_i} \in V_\phi \text{ and } \phi_i \in \mathsf{sub}(\phi).$$

By $\mathcal{S}_\phi$ we denote the *state classifier* $\mathcal{S}_\phi$, *i.e.*, $\mathbf{x}_{s_i}^{(T)} = \mathcal{S}_\phi(\mathbf{x}_{s_i}^{(0)}, \mathbf{x}_{s_{i+1}}, T)$.

Trace Classifier and Relationship

Definition 3 (Trace Classifier)

Let ϕ be an LTL$_f$ formula such that $|\text{sub}(\phi)| = L$, and $\pi = s_0, s_1, \ldots, s_n$ a trace. By $\mathcal{T}_\phi$ we denote the *trace classifier* which takes a vector $\mathbf{x}_{s_i}^{(0)}$ and a number of iterations $T \in \mathbb{N}$ as input and $\mathbf{x}_{s_i}^{(T)}$ as output; *i.e.*, $\mathbf{x}_{s_i}^{(T)} = \mathcal{T}_\phi(\mathbf{x}_{s_i}^{(0)}, T)$, where

$$\mathcal{T}_\phi(\mathbf{x}_{s_i}^{(0)}, T) = \begin{cases} \mathcal{S}_\phi(\mathbf{x}_{s_i}^{(0)}, \mathcal{T}_\phi(\mathbf{x}_{s_{i+1}}^{(0)}, T), T), & 0 \leq i < n \\ \mathcal{S}_\phi(\mathbf{x}_{s_i}^{(0)}, \mathbf{0}, T), & i = n \end{cases} \tag{2}$$

Trace Classifier and Relationship

Definition 3 (Trace Classifier)

Let ϕ be an LTL$_f$ formula such that $|\text{sub}(\phi)| = L$, and $\pi = s_0, s_1, \ldots, s_n$ a trace. By $\mathcal{T}_\phi$ we denote the *trace classifier* which takes a vector $\mathbf{x}_{s_i}^{(0)}$ and a number of iterations $T \in \mathbb{N}$ as input and $\mathbf{x}_{s_i}^{(T)}$ as output; *i.e.*, $\mathbf{x}_{s_i}^{(T)} = \mathcal{T}_\phi(\mathbf{x}_{s_i}^{(0)}, T)$, where

$$\mathcal{T}_\phi(\mathbf{x}_{s_i}^{(0)}, T) = \begin{cases} \mathcal{S}_\phi(\mathbf{x}_{s_i}^{(0)}, \mathcal{T}_\phi(\mathbf{x}_{s_{i+1}}^{(0)}, T), T), & 0 \le i < n \\ \mathcal{S}_\phi(\mathbf{x}_{s_i}^{(0)}, \mathbf{0}, T), & i = n \end{cases} \tag{2}$$

Theorem 1 (Relationship between GNNs and LTL$_f$)

Let ϕ be an LTL$_f$ formula such that $|\text{sub}(\phi)| = L$. For every trace $\pi = s_0, s_1, \ldots, s_n$, $(\mathcal{T}_\phi(\mathbf{x}_{s_0}^{(0)}, L))_L = 1$ if and only if $\pi \models \phi$.

Framework of GLTLf

The framework of GLTLf is summarized as follows.

1. In the first phase, we train a simple homogeneous AC-GNN model $\mathcal{S}$ to distinguish positive and negative traces.
2. In the second phase, we interpret the parameters of $\mathcal{S}$ to obtain an LTL_f formula ϕ_A.

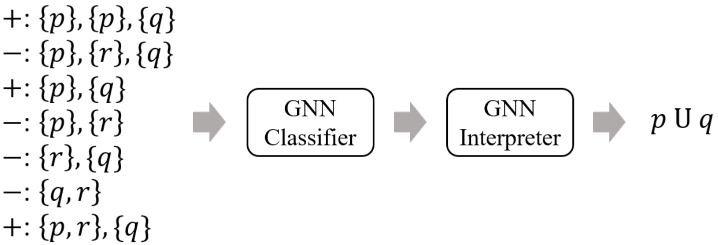

Figure 4: The framework of GLTLf.

Converting Traces to Graphs

Let $\pi = s_0, s_1, \ldots, s_n$ be a trace. We convert π to a *directed path graph* $G_\pi = (V_\pi, E_\pi)$, where V_π is the set of vertices and E_π is the set of edges.

- Each vertex v_i in V_π corresponds to a state s_i in π.
- For each pair of adjacent states s_i, s_{i+1} in π, there is an edge $\langle v_{i+1}, v_i \rangle$ in E_π.

Example 3

Let $\pi = s_0, s_1, s_2$ be a trace, where $s_0 = \{p, \neg q\}$, $s_1 = \{\neg p, q\}$, and $s_2 = \{\neg p, \neg q\}$. The directed path graph G_π of π is illustrated as follows.

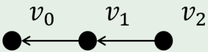

Figure 5: The directed path graph of π.

Training GNNs as Classifiers

Apply GNN on the directed path graph G_π.

- Each vertex $v_i \in V_\pi$ has a feature vector $\mathbf{x}_{v_i}$ $([x_1, \ldots, x_{|\mathbb{P}|}, x_\top, x_{|\mathbb{P}|+2}, \ldots, x_L])$.
- The state classifier $\mathcal{S}$ follows Definition 2, but $\mathbf{C}$, $\mathbf{A}$, and $\mathbf{b}$ need to be trained.
  - **Initializing.** For all $1 \leq j \leq L$, $(\mathbf{x}_{v_i}^{(0)})_j = 1$ if $\phi_j \in s_i$ or $(\mathbf{x}_{v_i}^{(0)})_j = 0$ otherwise.
  - **Updating.**

$$\begin{aligned}\mathbf{x}_{v_i}^{(t)} &= \mathsf{COM}(\mathbf{x}_{v_i}^{(t-1)}, \mathsf{AGG}(\{\{\mathbf{x}_u^{(t-1)} | u \in \mathcal{N}(v_i)\}\})) \\ &= \sigma(\mathbf{C}\mathbf{x}_{v_i}^{(t-1)} + \mathbf{A}\mathbf{x}_{v_{i+1}} + \mathbf{b}).\end{aligned} \tag{3}$$

where σ is a variant of leaky ReLU defined by:

$$\sigma(x) = \begin{cases} \alpha x, & x < 0, \\ x, & 0 \leq x \leq 1, \\ \alpha x + 1 - \alpha, & x > 1. \end{cases} \tag{4}$$

Training GNNs as Classifiers

Apply GNN on the directed path graph G_π.

- The trace classifier $\mathcal{T}$ is defined by Definition 3.
  - $(\mathcal{T}(\mathbf{x}_{v_i}^{(0)}, L))_j$ indicates whether $\pi_i \models \phi_j$, where $\phi_j \in \mathsf{sub}(\phi)$.
- We train $\mathcal{S}$ to minimize the joint objective:

$$\zeta = \zeta_1 + \beta\zeta_2 + \gamma(\zeta_3 + \zeta_4). \tag{5}$$

- $\zeta_1 = ((\mathcal{T}(\mathbf{x}_{v_0}^{(0)}, L))_L - \phi_T(\pi))^2$
- $\zeta_2 = \sum_{i=1}^{L}\sum_{j=1}^{L} |(\mathbf{C})_{ij}| + \sum_{i=1}^{L}\sum_{j=1}^{L} |(\mathbf{A})_{ij}|$
- $\zeta_3 = \sum_{i=1}^{L}\sum_{j=1}^{L}(\mathsf{Relu}((\mathbf{C})_{ij} - 2) + \mathsf{Relu}(-(\mathbf{C})_{ij} - 1))$
- $\zeta_4 = \sum_{i=1}^{L}\sum_{j=1}^{L}(\mathsf{Relu}((\mathbf{A})_{ij} - 1) + \mathsf{Relu}(-(\mathbf{A})_{ij}))$

Interpreting LTL$_f$ Formulae from GNNs

For any $1 \leq i \leq L$, interpret a sub-formula from $(\mathbf{C})_i$, $(\mathbf{A})_i$, and $(\mathbf{b})_i$.

- First recommend a set of candidate formulae based on a *cheap metric*.
- Then select the best formula based on a *expensive metric*, *i.e.*, the discrimination effect for the traces.

Interpreting LTL$_f$ Formulae from GNNs

For any $1 \leq i \leq L$, interpret a sub-formula from $(\mathbf{C})_i$, $(\mathbf{A})_i$, and $(\mathbf{b})_i$.

- First recommend a set of candidate formulae based on a *cheap metric*.
- Then select the best formula based on a *expensive metric*, *i.e.*, the discrimination effect for the traces.

Definition 4 (Cheap Metric: Interpretation Similarity)

Let ϕ_i be an LTL$_f$ formula and $\mathbf{C}, \mathbf{A}, \mathbf{b}$ parameters. The *interpretation similarity* between ϕ_i and the interpretation of $(\mathbf{C})_i, (\mathbf{A})_i, (\mathbf{b})_i$ is defined as follows:

$$\mathsf{sim}(\phi_i, (\mathbf{A})_i, (\mathbf{C})_i, (\mathbf{b})_i) =$$
$$\begin{cases} \frac{1}{1+\mathsf{dis}([(\mathbf{C})_{ij},(\mathbf{b})_i],[-1,1])}, & \phi_i = \neg\phi_j, \\ \frac{1}{1+\mathsf{dis}([(\mathbf{A})_{ij}],[1])}, & \phi_i = \mathsf{X}\phi_j, \\ \frac{1}{1+\mathsf{dis}([(\mathbf{C})_{ij},(\mathbf{C})_{ik},(\mathbf{b})_i],[1,1,-1])}, & \phi_i = \phi_j \wedge \phi_k, \\ \frac{1}{1+\mathsf{dis}([(\mathbf{C})_{ij},(\mathbf{C})_{ik},(\mathbf{A})_{ii},(\mathbf{b})_i],[1,2,1,-1])}, & \phi_i = \phi_j \mathsf{U} \phi_k \end{cases}$$

where $\mathsf{dis}(v_1, v_2)$ is the euclidean distance between the vector v_1 and the vector v_2.

Content

Setting

Benchmarks (noise-free):

- 5 domains for $k_f \in \{3, 6, 9, 12, 15\}$
- For each domain, 50 datasets
- For each dataset,
  - randomly target formula ϕ_A of which has k_f sub-formulae of non-atomic propositions
  - randomly $250/250$ positive/negative traces of ϕ_A as the training set
  - randomly $500/500$ positive/negative traces of ϕ_A as the test set

Setting

Benchmarks (noise-free):

- 5 domains for $k_f \in \{3, 6, 9, 12, 15\}$
- For each domain, 50 datasets
- For each dataset,
  - randomly target formula ϕ_A of which has k_f sub-formulae of non-atomic propositions
  - randomly $250/250$ positive/negative traces of ϕ_A as the training set
  - randomly $500/500$ positive/negative traces of ϕ_A as the test set

Benchmarks (noise):

- randomly chose some traces from the noise-free benchmarks and give them wrong labels
- noise rate δ

Setting

Competitors:

Table 1: Details about SOTA approaches.

approach	noisy data	arbitrary formulae
C.&M. [1]	×	✓
BayesLTL [4]	✓	×
MaxSAT-DT [2]	✓	✓
GLTLf (Ours)	✓	✓

Setting

Competitors:

Table 1: Details about SOTA approaches.

approach	noisy data	arbitrary formulae
C.&M. [1]	×	✓
BayesLTL [4]	✓	×
MaxSAT-DT [2]	✓	✓
GLTLf (Ours)	✓	✓

Tasks:

- All approaches first learn an LTL_f formula from the training set and then are compared by evaluating the classification effect of the learned formulae on the test set.

Comparisons on Noise-free Data

Table 2: Experiment results at $k_g = 15$ on noise-free data across different approaches. N_s stands for the number of successfully solved formulae (50 total).

	$k_f = 3$			$k_f = 6$			$k_f = 9$			$k_f = 12$			$k_f = 15$		
	Acc(%)	F_1(%)	N_s	Acc(%)	F_1(%)	N_s	Acc(%)	F_1(%)	N_s	Acc(%)	F_1(%)	N_s	Acc(%)	F_1(%)	N_s
MaxSAT-DT	100	100	49	100	100	19	100	100	8	100	100	5	100	100	5
C.&M.	99.77	99.77	50	97.93	96.79	47	97.14	95.55	35	95.10	91.91	20	93.74	87.49	8
BayesLTL	85.19	85.96	50	77.94	76.78	50	74.08	75.73	50	72.77	73.47	50	74.85	77.32	50
GLTLf	93.50	93.09	50	86.53	86.28	50	78.09	78.66	50	77.69	78.63	50	78.53	79.34	50

- GLTLf significantly surpasses BayesLTL.
- Although MaxSAT-DT and C.&M. are in the lead, they cannot solve long formulae.

Comparisons on Noisy Data

- GLTLf is proven to be more noise-tolerated then other approaches.
- GLTLf surpasses BayesLTL notably.
- MaxSAT-DT and C.&M. also fail to solve all formulae when the training data was noisy.

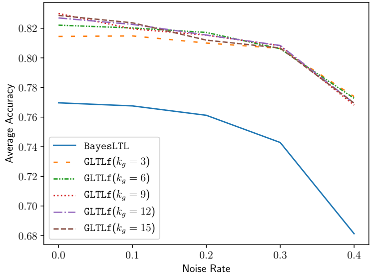

Figure 6: Accuracies among different noise rates. The results are the average results of the 5 accuracies when $k_f \in \{3, 6, 9, 12, 15\}$.

Performance of Interpreting

- There is a significant gap between net accuracies and interpreting accuracies.
- The gap suggests that there is a lot of potential for the interpreting method to evolve.

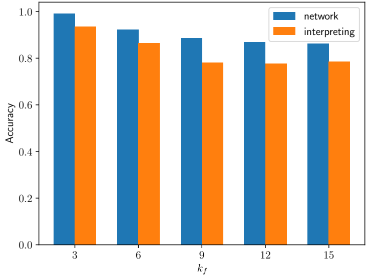

Figure 7: Net accuracies and interpreting accuracies. There results are from $k_g = 15$ setting and noise-free datasets.

Scalability and Robustness

- GLTLf can handle long formulae with higher performance compared to other approaches.
- GLTLf is able to solve formulae in various sizes and is rather robust.
  - Larger networks achieve better accuracies, even when solving short formulae.
  - Small networks can also get good performance on long formulae.

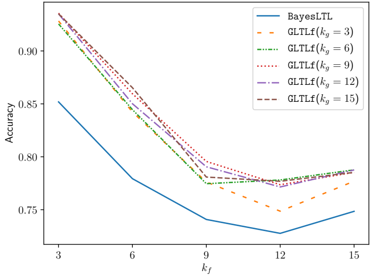

Figure 8: Accuracies among different k_g settings. There results are from noise-free datasets.

Content

Conclusion and Future Work

Conclusion:

1. We theoretically bridge LTL_f inference to GNN inference, which provides a new method for learning arbitrary LTL_f formulae from noisy data.
2. Based on the theoretical result, we design a GNN-based approach, named as `GLTLf`.
3. Experimental results demonstrate that our approach is stronger robustness for noisy data and better scalability in data size.

Future work:

1. extend our approach to learning LTL_f with uncertainty.
2. explore interpretable GNN learning.

References I

[1] Camacho, A.; and McIlraith, S. A. 2019. Learning Interpretable Models Expressed in Linear Temporal Logic. In *ICAPS*, 621–630.

[2] Gaglione, J.; Neider, D.; Roy, R.; Topcu, U.; and Xu, Z. 2021. Learning Linear Temporal Properties from Noisy Data: A MaxSAT Approach. *CoRR*, abs/2104.15083.

[3] Kasenberg, D.; and Scheutz, M. 2017. Interpretable apprenticeship learning with temporal logic specifications. In *CDC*, 4914–4921.

[4] Kim, J.; Muise, C.; Shah, A.; Agarwal, S.; and Shah, J. 2019. Bayesian Inference of Linear Temporal Logic Specifications for Contrastive Explanations. In *IJCAI*, 5591–5598.

[5] Neider, D.; and Gavran, I. 2018. Learning Linear Temporal Properties. In *FMCAD*, 1–10.

Thank you for your listening!