

Learning Nonlinear Loop Invariant with Gated Continuous Logic Networks

Jianan Yao et. al

June 23, 2020

History of the Research

CLN2INV: LEARNING LOOP INVARIANTS WITH CONTINUOUS LOGIC NETWORKS

In this paper, they propose the algorithm of learning loop invariants using CLN.

This paper generalize the model into GCLN, where “G” stands for gated.

Overview

- ▶ Introduction to learning nonlinear invariant.
- ▶ Workflow of the algorithm.
- ▶ Detailed description of the theory and techniques.
- ▶ Experimental evaluation.

Difficulties

- ▶ Large search space with high magnitude terms. e.g. terms like x^2 and x^y grows exponentially.
- ▶ Limited samples. Bounds on the number of loop iterations with integer variables.
- ▶ Distinguishing sufficient inequalities.

Ways to Resolve

- ▶ **Learning with G-CLNs**, where the gated value can be used to turn on or off the terms. Combine dropout.
- ▶ **Fractional sampling**. Relax the semantic of the loop to continuous functions.
- ▶ **Piecewise Biased Quadratic Units(PBQU)**. A way to penalizes loose fits and converges to tight constraints on data.

Background

Definition (Loop Invariant Inference)

Given the precondition P and postcondition Q . Finding an inductive invariant I such that,

$$P \implies I, \{I \wedge LC\} C \{I\}, I \wedge \neg LC \implies Q$$

Loop invariants can be encoded in SMT. The data driven invariant inference is to find SMT formula F s.t.

$$\forall x \in X, F(x) = \text{True}$$

Making Loop Invariant Learnable

Using a differentiable logic: Basic Fuzzy Logic(BL). BL is a relaxation of FOL on continuous truth value on the interval $[0, 1]$.

- ▶ t-norms(\otimes). Consistency: $t \otimes 1 = 1$, $t \otimes 0 = 0$, commutative and monotonic.
- ▶ t-conorms(\oplus), operates as disjunctions for BL and derived from DeMorgan's law with $\neg t = 1 - t$

Semantic Mapping

Principles:

1. Remain the meaning of the logic.
2. Continuous and smooth.
3. Increasing when unsat terms change to sat terms.

We use \mathcal{S} as the mapping function and it is defined recursively.

$\mathcal{S}(F) : X \rightarrow [0, 1]$.

$$\text{Conjunction:} \quad \mathcal{S}(F_1 \wedge F_2) \triangleq \mathcal{S}(F_1) \otimes \mathcal{S}(F_2)$$

$$\text{Disjunction:} \quad \mathcal{S}(F_1 \vee F_2) \triangleq \mathcal{S}(F_1) \oplus \mathcal{S}(F_2)$$

$$\text{Negation:} \quad \mathcal{S}(\neg F) \triangleq 1 - \mathcal{S}(F)$$

$$\text{Greater Than:} \quad \mathcal{S}(x_1 > x_2) \triangleq \frac{1}{1 + e^{-B(x_1 - x_2 - \epsilon)}}$$

$$\text{Greater or Equal to:} \quad \mathcal{S}(x_1 \geq x_2) \triangleq \frac{1}{1 + e^{-B(x_1 - x_2 + \epsilon)}}$$

Example

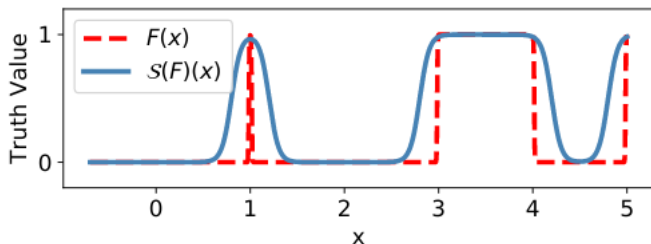
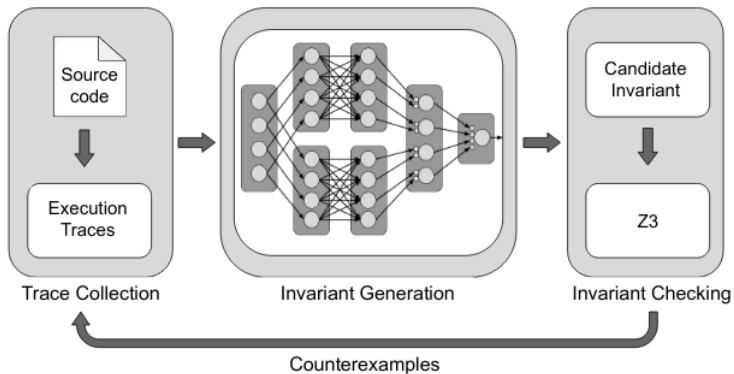


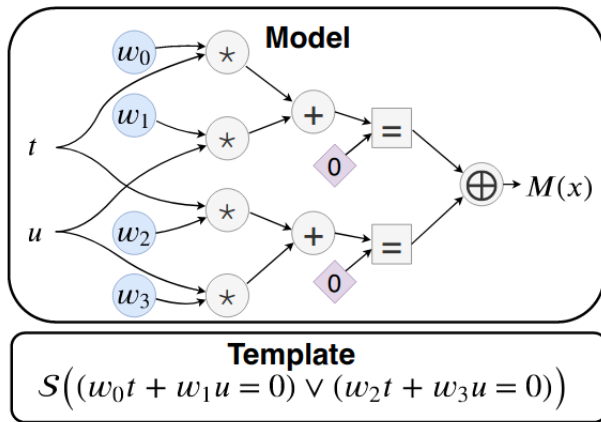
Figure 2. Plot of the formula $F(x) \triangleq (x = 1) \vee (x \geq 5) \vee (x \geq 2 \wedge x \leq 3)$ and its associated CLN $M(x)$.

Workflow



Continuous Logic Network

Example



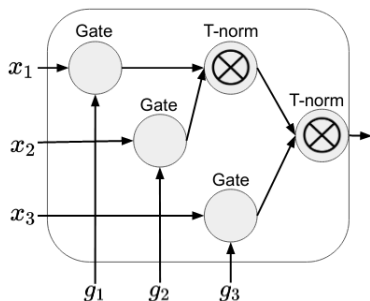
Gated Continuous Logic Network

Previous invariant learning with CLN require a preset template. To let the model fit the data automatically instead of given a template ahead, we use gated CLN.

gated t-norms:

$$T_G(x, y; g_1, g_2) = (1 + g_1(x - 1)) \otimes (1 + g_2(y - 1))$$

Likewise for gated t-conorms.



Current learning parameters: W, g_i

Learning Target

Minimize

$$\mathcal{L}(X; W, G) = \sum_{x \in X} (1 - \mathcal{M}(x; W, G)) + \lambda_1 \sum_{g_1 \in T_G} (1 - g_1) + \lambda_2 \sum_{g_i \in T'_G} g_i$$

where λ_i are normalization parameters.

Formula Extractions

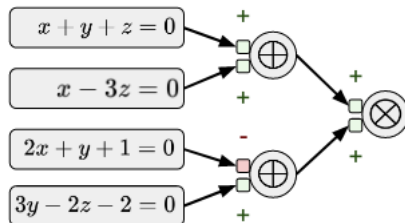


Figure 6. An instance of gated CLN. “+” means activated ($g=1$) and “-” means deactivated ($g=0$). The SMT formula learned is $(3y - 3z - 2 = 0) \wedge ((x - 3z = 0) \vee (x + y + z = 0))$.

Formula Extractions

Algorithm 1 Formula Extraction Algorithm.

Input: A gated CLN model \mathcal{M} , with input nodes $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ and output node p .

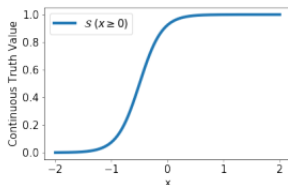
Output: An SMT formula F

Procedure ExtractFormula(\mathcal{M})

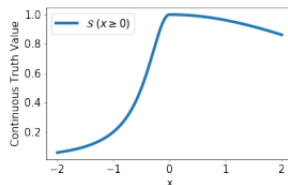
```
1: if  $p = T_G(\mathcal{M}_1, \dots, \mathcal{M}_n; g_1, \dots, g_n)$  then
2:    $F := \text{True}$ 
3:   for  $i := 1$  to  $n$  do
4:     if  $g_i > 0.5$  then
5:        $F := F \wedge \text{ExtractFormula}(\mathcal{M}_i)$ 
6:   else if  $p = T'_G(\mathcal{M}_1, \dots, \mathcal{M}_n; g_1, \dots, g_n)$  then
7:      $F := \text{False}$ 
8:     for  $i := 1$  to  $n$  do
9:       if  $g_i > 0.5$  then
10:         $F := F \vee \text{ExtractFormula}(\mathcal{M}_i)$ 
11:   else if  $p = 1 - \mathcal{M}_1$  then
12:      $F := \neg \text{ExtractFormula}(\mathcal{M}_1)$ 
13:   else
14:      $F := \text{BuildAtomicFormula}(\mathcal{M})$ 
```

Piecewise Construction

$$\mathcal{S}(t \geq u) \triangleq \begin{cases} \frac{c_1^2}{(t-u)^2 + c_1^2} & t < u \\ \frac{c_2^2}{(t-u)^2 + c_2^2} & t \geq u \end{cases}$$



(a) Plot of $\mathcal{S}(x \geq 0)$ with the CLNs' sigmoid construction.



(b) Plot of $\mathcal{S}(x \geq 0)$ with our piecewise construction.

Figure 7. Comparison of the mapping \mathcal{S} on \geq . The hyper-parameters are $B = 5$, $\epsilon = 0.5$, $c_1 = 0.5$, and $c_2 = 5$.

Fractional Sampling

```
//pre: x = y = 0
//      /\ k >= 0
while (y < k) {
    y++;
    x += y * y * y;
}
//post: 4x == k^2
//      * (k + 1)^2
```

(a) The ps4 program in the benchmark.

x	y	y^2	y^3	y^4
0	0	0	0	0
1	1	1	1	1
9	2	4	8	16
36	3	9	27	81
100	4	16	64	256
225	5	25	125	625

(b) Training data generated without Fractional Sampling.

x	y	y^2	y^3	y^4	x_0	y_0	y_0^2	y_0^3	y_0^4
-1	-0.6	0.36	-0.22	0.13	-1	-0.6	0.36	-0.22	0.13
-0.9	0.4	0.16	0.06	0.03	-1	-0.6	0.36	-0.22	0.13
1.8	1.4	1.96	2.74	3.84	-1	-0.6	0.36	-0.22	0.13
0	-1.2	1.44	-1.73	2.07	0	-1.2	1.44	-1.73	2.07
0	-0.2	0.04	-0.01	0.00	0	-1.2	1.44	-1.73	2.07
0.5	0.8	0.64	0.52	0.41	0	-1.2	1.44	-1.73	2.07

(c) Training data generated with fractional sampling.

Optimization

Data Normalization:

Large inputs cause instability and prevent the CLN model from converging. In the implementation, the paper requires the L2-norm equals a set value I .

Weight Regularization

To avoid trivial invariant, we require the L^p -norm of the weight vector equals to a nonzero value.

Term Dropout:

Large number of terms poses difficulties for learning. Random dropout of terms.

Experimental Evaluation: Nonlinear

Problem	Degree	# Vars	PIE	NumInv	G-CLN
divbin	2	5	-	✓	✓
cohendiv	2	6	-	✓	✓
mannadiv	2	5	✗	✓	✓
hard	2	6	-	✓	✓
sqrt1	2	4	-	✓	✓
dijkstra	2	5	-	✓	✓
cohencu	3	5	-	✓	✓
egcd	2	8	-	✓	✓
egcd2	2	11	-	✗	✓
egcd3	2	13	-	✗	✓
prodbin	2	5	-	✓	✓
prod4br	3	6	✗	✓	✓
fermat1	2	5	-	✓	✓
fermat2	2	5	-	✓	✓
freire1	2	3	-	✗	✓
freire2	3	4	-	✗	✓
knuth	3	8	-	✓	✗
lcm1	2	6	-	✓	✓
lcm2	2	6	✗	✓	✓
geo1	2	5	✗	✓	✓
geo2	2	5	✗	✓	✓
geo3	3	6	✗	✓	✓
ps2	2	4	✗	✓	✓
ps3	3	4	✗	✓	✓
ps4	4	4	✗	✓	✓
ps5	5	4	-	✓	✓
ps6	6	4	-	✓	✓

Experimental Evaluation: Nonlinear

Problem	Data Norm.	Weight Reg.	Drop-out	Frac. Sampling	Full Method
divbin	X	X	✓	✓	✓
cohendiv	X	X	X	✓	✓
mannadiv	X	X	✓	✓	✓
hard	X	X	✓	✓	✓
sqrt1	X	X	X	✓	✓
dijkstra	X	X	✓	✓	✓
cohencu	X	X	✓	✓	✓
egcd	X	✓	X	✓	✓
egcd2	X	✓	X	✓	✓
egcd3	X	✓	X	✓	✓
prodbin	X	✓	✓	✓	✓
prod4br	X	✓	✓	✓	✓
fermat1	X	✓	✓	✓	✓
fermat2	X	✓	✓	✓	✓
freire1	X	✓	✓	✓	✓
freire2	X	✓	X	✓	✓
knuth	X	X	X	X	X
lcm1	X	✓	✓	✓	✓
lcm2	X	✓	✓	✓	✓
geo1	X	✓	✓	✓	✓
geo2	X	✓	✓	✓	✓
geo3	X	✓	✓	✓	✓
ps2	✓	X	✓	✓	✓
ps3	X	X	✓	✓	✓
ps4	X	X	✓	✓	✓
ps5	X	X	✓	X	✓
ps6	X	X	✓	X	✓

Experimental Evaluation: Nonlinear

Problem	Convergence Rate of CLN	Convergence Rate of G-CLN
Conj Eq	75%	95%
Disj Eq	50%	100%
Code2Inv 1	55%	90%
Code2Inv 11	70%	100%
ps2	70%	100%
ps3	30%	100%

Experimental Evaluation: Linear

Among 133 linear cases, except 9 unsovable cases, the tool solved all of the rest.