Note: Learning to Generate SAT Formulas

Yu-Zhe Shi

April 8, 2020

1 Introduction

- Motivation: To extend existing SAT benchmarks.
- Intuition: G2SAT is a model to predict the distribution of literals and clauses in LCGs. It splits LCGs into trees in training phase and merge the trees into larger LCGs in inference phase. Hence, G2SAT has the capability to generate LCGs which preserves the properties of training LCGs, making it possible to generate LCGs with specific desired properties.

2 Author Profiles

- Jiaxuan You, 3rd year PhD, CS, Stanford; `https://cs.stanford.edu/~jiaxuan/`
- Haoze Wu, 2nd year PhD, CS, Stanford; `https://anwu1219.github.io/`
- Clark Barrett, Associate Prof, CS, Stanford; `http://theory.stanford.edu/~barrett/`
- Raghuram Ramanujan, Assistant Prof, MATH-CS, Davidson; `https://www.davidson.edu/people/raghu-ramanujan`
- Jure Leskovec, Associate Prof, CS, Stanford. `https://cs.stanford.edu/people/jure/`

3 Prerequisites

3.1 SAT

- SATisfiability problem is the first NP-Complete problem proved.
- A SAT formula ϕ is a composition of Boolean variables x_i connected with logical operators $\vee$, $\wedge$, $\neg$.

$$\exists x_i,\ \phi(x_i) = 1, \Rightarrow \phi\ is\ satisfiable. \tag{1}$$

- CNF(Conjunctive Normal Form): $(x_1 \vee x_2 \vee \dots) \wedge (x_c \vee x_2 \vee \dots)$

3.2 LCG

- LCG(Literal-Clause Graph): Node$\leftarrow$Literal x_i, Clauses $x_i \wedge x_j$. Edge$\leftarrow$ $x_i \Leftrightarrow x_i \vee x_j$. LCGs can be presented in Bipartite graphs where the vertex set can be splitted into a vertex set of literals and a vertex set of clauses.

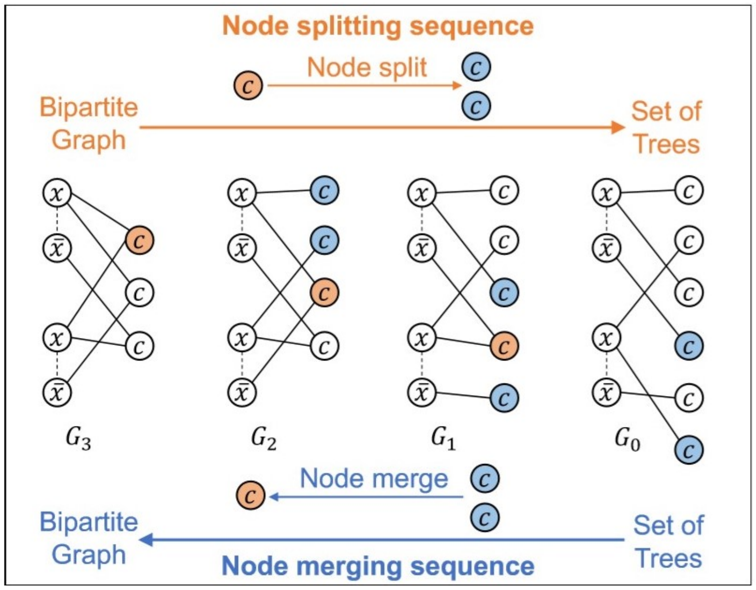
$$\begin{aligned} G = (\mathcal{V}, \mathcal{E}), \Rightarrow \mathcal{V} = \mathcal{V}_2 \cup \mathcal{V}_1 \\ \mathcal{V}_1 = \{l_1, \dots, l_n\}, \mathcal{V}_2 = \{c_1, \dots, c_m\} \end{aligned} \tag{2}$$

where there are n literals and m clauses in the LCG.

3.3 Graph Splitting and Merging

- $$\begin{aligned} NodeSplit(u,G) &\Rightarrow (u,v,G') \\ NodeMerge(u,v,G) &\Rightarrow (u,G') \end{aligned} \tag{3}$$

  Hence, the split-merge operation is symmetric.



4 Generate Bipartite Graph Iteratively: Split

- Objective: distribution over graph $p(G)$
- Use Markov property $p(G_i|G_{i-1}) = p(G_i|G_1,\ldots,G_{i-1})$ to predict $p(G)$ iteratively

  $$p(G) = \prod_{i=1}^{n} p(G_i|G_1,\ldots,G_{i-1}) \tag{4}$$

  where G_{i-1} denotes intermediate results.
- Under a sequence of split operations, a bipartite graph can be transformed into a set of trees.

5 Generate Bipartite Graph Iteratively: Merge

- $$\begin{aligned} p(G_i|G_{i-1}) &= p(NodeMerge(u,v,G_{i-1})|G_{i-1}) \\ &= Multinomial(h_u^T h_v/Z|\forall u,v \in \mathcal{V}_2^{G_{i-1}}) \end{aligned} \tag{5}$$

  where Z is a normalization term.
- Under a sequence of merge operations, a bipartite can be generated from a set of trees.

6 Embedding via GraphSAGE

- GraphSAGE is a graph convolutional network with inductive learning capability across different graphs and embedding unseen data.
- 3 node types to be embedded: positive literals, negative literals, clauses.
- embedding of node u at n-layer of GraphSAGE

$$\begin{aligned} n_u^l &= MeanPooling(ReLU(Q^l h_v^l + q^l | v \in Neighbor(u))) \\ h_u^{l+1} &= ReLU(W^l CONCAT(h_u^l, n_u^l)) \end{aligned} \tag{6}$$

7 Scalable G2SAT with Two-phase Generation Scheme

- Millions of candidate pairs, thus infeasible to compute Z.
- Introducing random variable u, v as random nodes with binary conditions (to merge or not to merge), relaxing $p(G_i|G_{i-1})$ to a joint distribution

$$\begin{aligned} &p(G_i, u, v|G_{i-1}) = p(u, v|G_{i-1})p(G_i|G_{i-1}, u, v) \\ &= p(u, v|G_{i-1})p(NodeMerge(u, v, G_{i-1})|G_{i-1}, u, v) \\ &= Uniform(\{(u, v)|\forall u, v \in \mathcal{V}_2^{G_{i-1}}\})Bernoulli(\sigma(h_u^T h_v)|u, v) \end{aligned} \tag{7}$$

- We needn't perform normalization since assignment of bernoulli distribution is only 0 or 1.

8 Training G2SAT

- The essence of training stage is performing node splitting.
- Sampling: Choose a random node s with $Deg(s) > 1$ to split into (u^+, v^+), which is regarded as positive samples. Randomly select another node $v^- \in \mathcal{V}_2^{G_{i-1}}$ $\{u^+, v^+\}$ and (u^+, v^-) forms negative samples.
- Cross-Entropy Loss for classification: to merge or not to merge

$$\mathcal{L} = -\mathbb{E}_{u^+,v^+}[\log(\sigma(h_u^T h_v))] - \mathbb{E}_{u^+,v^-}[\log(1 - \sigma(h_u^T h_v))] \tag{8}$$

- Training iterations terminate till all the remaining clause node $Deg(s) = 1$, or a clause is connected with a single literal in another word.
- The set of trees is saved as G_0, which serves as initialization for inference phase.

9 G2SAT at Inference Phase

- The essence of inference stage is performing node merging.
- Inintialized by G_0 and holds the same iteration with training phase.

10 Experiment

- This paper evaluates if G2SAT preserves the properties of training data (from SATLIB and past SAT competetion) by graph statistics and SAT solver performance.
- Three properties for Graph statistics:
  1. Scale-free structure parameters α_c and α_v:

  - The arity of nodes seems to follow a power-law distribution. Let k be the occurences of a variable

$$f^{pow}(k) = \frac{\alpha - 1}{k_{min}}(\frac{k}{k_min})^{-\alpha} \tag{9}$$

    There is another assumption that the arity of nodes follows an exponential distribution.

$$f^{exp}(k) = (1 - e^{-\beta})e^{-\beta(k-k_{min})} \tag{10}$$

  - Since real LCGs doesn't actually follow any of these distributions, distributions of some families of LCGs have high similarity to them. CMU, IBM follows power-law distribution both in consider of clauses $\alpha_c$and number of occurences α_v. 3-CNF formulas fit exponential distribution better. Hence, we can measure the similarity between the structures of LCGs from these families via α and β respectively.

2. Modularity is the difference between edges falling into a group and the expected number in an equivalent network with edges placed randomly. Let A_{ij} denote the number of edges between vertices i, j with degrees k_i, k_j, s_i is 1 when i belongs to a group of the bipartite graph while -1 when i belongs to another group, while the network contains m edges and n vertices. Modularity Q is

$$Q = \frac{1}{4m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})s_i s_j \tag{11}$$

   Let $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$, we can rewrite Q as

$$Q = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s} \tag{12}$$

   where $\mathbf{B}$ is a real asymmetric matrix and $\mathbf{s}$ is column vector whose elements are s_i. Then Q can be

$$Q = \frac{1}{4m}\sum_{i=1}^{n}(\mathbf{u}_i^T\mathbf{s})^2\beta_i \tag{13}$$

   where β_i is the eigenvalue corresponding to eigenvector $\mathbf{u}_i$ of $\mathbf{B}$.
   When it comes to dividing a graph into multiple groups g of size n_g, we add an additional ΔQ

$$\Delta Q = \frac{1}{4m}(\sum_{ij} B_{ij}s_i s_j - \sum_{ij} B_{ij}) \tag{14}$$

3. Clustering Coefficient is a measure of the degree to which nodes in a graph tend to cluster together, taking temporal domain into consideration. Let $P_m(t)$ be the possibility of two vertices connected by an edge added at time t have m mutual neighbors and the graph contains $N(t)$ vertices,

$$P_m(t) = \frac{n_m(t)}{\frac{1}{2}N(t)[N(t) - 1]}R_m \tag{15}$$

   where $n_m(t)$ denotes the number of vertex pairs with m mutual neighbors just before the addition of the edge at time t. R_m is the measure of clustering, namely the relative probability of two vertices connected by the new edge, which is termporal independent.

$$R_m = A - Be^{-m/m_0} \tag{16}$$

   where A, B and m_0 are constants. R_m increases linearly when m is relatively small and the clustering of the network increases monotonically with R_m.

  Hence, we can measure the how much does the generated graph maintain the properties of training data.

- By comparing the performance of SAT solvers both on real data and generated data, we can measure the how much does the generated graph maintain the properties of training data as well.
- Training deep SAT solvers on generated data boosts their performance on real data.

Table 1: Graph statistics of generated formulas (mean ± std. (relative error to training formulas)).

Method	VIG		VCG			LCG
	Clustering	Modularity	Variable α_v	Clause α_c	Modularity	Modularity
Training	0.50±0.07	0.58±0.09	3.57±1.08	4.53±1.09	0.74±0.06	0.63±0.05
CA	0.33±0.08(34%)	0.48±0.10(17%)	6.30±1.53(76%)	N/A	0.65±0.08(12%)	0.53±0.05(16%)
PS(T=0)	0.82±0.04(64%)	0.72±0.13(24%)	3.25±0.89(9%)	**4.70±1.59(4%)**	0.86±0.05(16%)	**0.64±0.05(2%)**
PS(T=1.5)	0.30±0.10(40%)	0.14±0.03(76%)	4.19±1.10(17%)	6.86±1.65(51%)	0.40±0.05(46%)	0.41±0.05(35%)
G2SAT	**0.41±0.09(18%)**	**0.54±0.11(7%)**	**3.57±1.08(0%)**	4.79±2.80(6%)	**0.68±0.07(8%)**	0.67±0.03(6%)

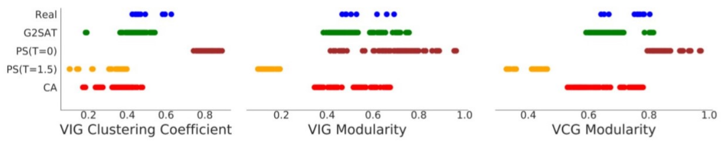


Figure 2: Scatter plots of distributions of selected properties of the generated formulas.

Table 2: Relative SAT Solver Performance on training as well as synthetic SAT formulas.

Method	Solver ranking	Accuracy
Training	$I_2, I_3, I_1, R_2, R_3, R_1$	100%
CA	$I_2, I_3, I_1, R_2, R_3, R_1$	**100%**
PS(T=0)	$R_3, I_3, R_2, I_2, I_1, R_1$	33%
PS(T=1.5)	$R_3, R_2, I_3, I_1, I_2, R_1$	33%
G2SAT	$I_1, I_2, I_3, R_2, R_3, R_1$	**100%**

Table 3: Performance gain when using generated SAT formulas to tune SAT solvers.

Method	Best parameters	Runtime(s)	Gain
Training	(0.95, 0.9)	2679	N/A
CA	(0.75, 0.99)	2617	2.31%
PS(T=0)	(0.75, 0.999)	2668	0.41%
PS(T=1.5)	(0.95, 0.9)	2677	0.07%
G2SAT	(0.95, 0.99)	**2190**	**18.25%**

11 Summary

- The training objective of the model is to decide ”what node to merge”. In essence, the model learns the intra-clause pattern and inter-literal pattern from training data.
- Training phase is node splitting while inference phase is node merging.
- To prune hypothesis space, G2SAT relaxes $p(G_i|G_{i-1})$ to $p(G_i, u, v|G_{i-1})$, thus transforming multinomial distribution to binary distribution.

12 Inspiration

- Decomposition: When the objective is hard to learn, we can solve the problem iteratively via intermediate objectives.
- Hybrid Model: We can use neural networks to obtain appropriate feature embeddings. This work doesn’t exploit the reciprocation between statistical learning models and logical learning models.
- Reduce multinomial distribution to binary distribution.
- Reduce enumerative traversal in the heposythesis space to random selection when the operation is commutative. (Maybe similar to changing sliding window to anchor boxes in CV?)