Using Machine Learning Classifiers in SAT Branching [Extended Abstract]

Ruth Helen Bergin [1], Marco Dalla[1,2], Andrea Visentin [1,3], Barry O'Sullivan[1,2,3], Gregory Provan[1]

[1]School of Computer Science & IT, University College Cork, Ireland
[2]SFI Centre for Research Training in AI, University College Cork, Ireland
[3]Insight SFI Research Centre for Data Analytics, University College Cork, Ireland
119401946@umail.ucc.ie,{m.dalla,b.osullivan,gprovan}@cs.ucc.ie, a.visentin@ucc.ie

Abstract

The Boolean Satisfiability Problem (SAT) can be framed as a binary classification task. Recently, numerous machine and deep learning techniques have been successfully deployed to predict whether a CNF has a solution. However, these approaches do not provide a variables assignment when the instance is satisfiable and have not been used as part of SAT solvers. In this work, we investigate the possibility of using a machine-learning SAT/UNSAT classifier to assign a truth value to a variable. A heuristic solver can be created by iteratively assigning one variable to the value that leads to higher predicted satisfiability. We test our approach with and without probing features and compare it to a heuristic assignment based on the variable's purity. We consider as objective the maximisation of the number of literals fixed before making the CNF unsatisfiable. The preliminary results show that this iterative procedure can consistently fix variables without compromising the formula's satisfiability, finding a complete assignment in almost all test instances.

Introduction

The Boolean Satisfiability Problem (SAT) is seminal in computer science. This problem can be treated as a binary classification task. Learning algorithms trained either on statistical features extracted at the instance level or on different representations of the CNF instances have been proved able to successfully predict satisfiability with great accuracy (Devlin and O'Sullivan 2008), (Dalla, Visentin, and O'Sullivan 2021). Some of these are based on statistical features, such as (Xu et al. 2008), also used on different tasks, e.g. category classification, solver selection and algorithm configuration. Machine learning applications in SAT solving have been widely researched in the last few years, with three main areas of research: creating standalone SAT solvers with pure machine learning methods(e.g. (Selsam et al. 2018)), replacing some components of existing conflict-driven clause learning (CDCL) solvers with learning-directed heuristics(e.g. (Vaezipoor et al. 2020), (Liang et al. 2018)) and modifying the local search solvers with learning-aided modules (e.g. (Zhang et al. 2020)).

This paper uses a SAT/UNSAT classification technique to guide the assignment of a literal in a branching solution. This approach iteratively assigns variables by predicting the satisfiability of the reduced formula. This work can be used as a heuristic SAT solver or as a pre-processing approach that reduces the size of the CNF by assigning variables without compromising its satisfiability.

Methodology

Iterative approach. We define a branching heuristic in which the variable fixed during the branch is selected according to a statistical measure of purity, while its truth value is assigned using a machine learning heuristic based on a SAT/UNSAT classifier. This heuristic can be iteratively applied to fix all the variables of a SAT instance.

A summary of the solving pipeline is shown below:

1. Of the free variables, select the one with the highest *purity score*.
2. Create two reduced formulas, one for each possible Boolean assignment. We define as *shadow CNFs* these partially fixed instances.
3. Extract statistical features from these CNFs.
4. Use a random forest to predict the satisfiability of the two *shadow CNFs*.
5. Branch on the assignment that has the higher predicted SAT probability.
6. (Only in training and evaluation) Using Glucose 4 SAT Solver, check if the reduced CNF is still satisfiable; otherwise, stop.
7. Iterate the process until all the variables are fixed or until the partial assignment is shown to be invalid.

Training the Random Forest. The random forest is trained on hand-crafted instances from the CBS dataset from SATlib. The dataset comprises 1000 random-3-SAT CNFs with controlled backbone size, each with 100 variables and 403 clauses. We probed 80 instances in order to generate a labelled dataset with a variety of sizes and balanced distribution between SAT and UNSAT. The probing is similar to the previous pipeline: a variable is selected, two reduced *shadow CNFs* are generated, SATzilla features are computed and stored as training inputs and the output of a SAT solver is stored as the target value. The branching can be directed by the solutions computed by the SAT solver to obtain a labelled dataset with SAT/UNSAT balance. A random forest classifier is then hyperparametrised and trained on this dataset. Finally, the classifier is deployed in step 5. of the iterative approach. We evaluate a model trained in the statistical SATzilla features (**ML** and one that also includes the probing features (**ML+probing**).

Variable Selection

We use as variable selection (point 1. of the iterative approach) an approach that focuses on purity inspired by the Davis-Putnam Pure Literal Rule. While the strict purity of a literal is a binary characteristic, it is possible to implement a new measure that evaluates how close a literal is to purity and its importance for the formula resolution. The goal is to select a variable that, if assigned incorrectly, makes the resulting formula unsatisfiable. Among the unfixed variables, we select the one with the higher score computed with the following formula:

$$score(v) = |f(l) - f(\neg l)| \tag{1}$$

where l is the literal in which the variable v appears positively, and $\neg l$ is the negated literal. Function f is a measure of the literal relevance and purity:

$$f(l) = \frac{number\_appearances(l) * cov(l)}{avg\_clause\_size(l)} \tag{2}$$

We included the average clause size in which the literal appears as an approximation of the literal importance; if a literal is part of smaller clauses, its assignment becomes more important for satisfiability. Another measure of how important a literal is to instance satisfaction is its covariance with the negations of crucial literals. For example, if a literal l shares a binary clause with literal k, but $\neg k$ is in a unit clause, then it's imperative to assign $l = True$ to prevent the contradiction of a unit clause. This concept can be generalised to clauses of any size by distributing the weight of a literal to the covariant literals of its negation. To represent this, let the K be the set of literals which share a clause with l.

$$cov(l) = \sum_{k \in K} \frac{f(\neg k)}{number\_appearances(k)} \tag{3}$$

Results

We consider two different baselines: a random assignment (**Random**) and the assignment that makes most of the literals of that variable true (**Majority**), e.g. we fix the variable to false if the literal mostly appears negated. We run the approaches on 50 randomly selected CBS instances unseen during the random forest training phase. Figure 1 shows the number of variables fixed before making the instance unsatisfiable. At every step, a SAT solver assesses if the last assignment compromises the satisfiability of the formula. The random performs poorly, early making an invalid assignment. A simple **Majority** heuristic strongly outperforms the classifier based on statistical features, assigning to completion 28% of the test instances. The ML approach with probing can solve all the test set to completion, with the exception of an early invalid assignment.

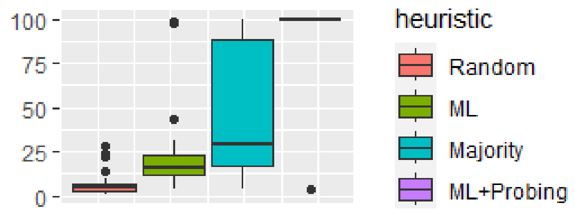

Figure 1: Percentage of variables fixed maintaining the instance satisfiability

Discussion and Further Work

This preliminary work investigates the usage of SAT/UNSAT machine learning classifiers as branching heuristics. Early experiments show that this approach can successfully guide the exploration of the search tree.

Further investigations on alternative variable selection heuristics, classifiers, and extracted features should be conducted to improve the performance. The generalisation capabilities should be assessed by increasing the size and variety of the test set. Further, the pipeline itself could be improved including the implementation of backtracking.

Acknowledgments

This publication has emanated from research conducted with financial support of Science Foundation Ireland under Grants 12/RC/2289-P2 and 18/CRT/6223, which are co-funded under the European Regional Development Fund.

References

Dalla, M.; Visentin, A.; and O'Sullivan, B. 2021. Automated SAT Problem Feature Extraction using Convolutional Autoencoders. In *33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021, Washington, DC, USA, November 1-3, 2021*, 232–239. IEEE.

Devlin, D.; and O'Sullivan, B. 2008. Satisfiability as a Classification Problem. *Proc. of the 19th Irish Conf. on Artificial Intelligence and Cognitive Science*.

Liang, J. H.; Oh, C.; Mathew, M.; Thomas, C.; Li, C.; and Ganesh, V. 2018. Machine Learning-Based Restart Policy for CDCL SAT Solvers. In *International Conference on Theory and Applications of Satisfiability Testing*.

Selsam, D.; Lamm, M.; Bünz, B.; Liang, P.; de Moura, L.; and Dill, D. L. 2018. Learning a SAT Solver from Single-Bit Supervision. *CoRR*, abs/1802.03685.

Vaezipoor, P.; Lederman, G.; Wu, Y.; Grosse, R.; and Bacchus, F. 2020. Learning Clause Deletion Heuristics with Reinforcement Learning. In *Proceedings of the Conference on Artificial Intelligence and Theorem Proving (AITP)*.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. SATzilla: portfolio-based algorithm selection for SAT. *Journal of artificial intelligence research*, 32: 565–606.

Zhang, W.; Sun, Z.; Zhu, Q.; Li, G.; Cai, S.; Xiong, Y.; and Zhang, L. 2020. NLocalSAT: Boosting Local Search with Solution Prediction. In *International Joint Conference on Artificial Intelligence*.