LightGBM Hyperparameter Optimization for Clause Classification in Theorem Proving *

Zarathustra Goertzel[1], Jan Jakubův[1,2], Mikoláš Janota[1], and Cezary Kaliszyk[2]

[1] Czech Technical University in Prague, Prague, Czech Republic
[2] University of Innsbruck, Innsbruck, Austria

1 Motivation: Machine Learning in Theorem Proving

Applications of machine learning (ML) in automated theorem proving (ATP) often involve training of models from large datasets. This training is usually performed by some machine learning framework, and there are many frameworks to choose from. However, many machine learning methods are parametrized and quite often using the right hyperparameter values is essential to achieve good prediction results. In our experiments with the ML theorem prover ENIGMA [6, 7, 3, 5], we have selected hyperparameter values based on our experience, or we've performed a grid search over some set of possible values. Additionally, as a part of our recent experiments with ENIGMA on Isabelle [4], we have implemented an automated hyperparameter selection tool `lgbtune`.[1] While this tool has been successfully used during the experiments, its performance has never been evaluated in detail, and this is the main topic of this work.

ENIGMA is a machine learning guidance system for automated theorem prover E [9]. E's input is a first-order logic problem consisting of axioms and a conjecture to be proved. This problem is translated to first order *clauses*, and a proof search based on a *given clause loop* is launched in the space of clauses. In each step of this proof search, one unprocessed clause, called *given*, is selected for processing. The given clause selection is one of the most important choice points performed during the proof search, and this is where ENIGMA guides the prover.

ENIGMA experiments are done using the *training/evaluation loop*. Given a set of benchmark problems $\mathcal{P}$, we run E over all problems $\mathcal{P}$. We analyze *successful* proof searches, and clauses selected during the search are classified as *useful* and *useless*. The clauses that participate in the final proof are considered useful, while the others are useless because their processing might have been avoided. On thusly labeled clauses, we train a machine learning model $\mathcal{M}$ to distinguish useful clauses from useless ones. Model $\mathcal{M}$ is then used to guide next E search over problems $\mathcal{P}$. This results in new successful proof searches used to construct new training data for the next iteration of the training/evaluation loop.

As the underlying machine learning method, ENIGMA can use decision trees or graph neural networks. In this work we concentrate on hyperparameter setting for training of decision tree models. ENIGMA support two decision tree frameworks: XGBoost [2] and LightGBM [8]. Recently, we favor LightGBM because it is faster and more stable on large training data. The data input for the LightGBM trainer are labeled clauses (as useful/useless) represented by numeric feature vectors. The training data we encounter with ENIGMA are quite specific and consist of a large amount (millions of clauses) of long (single vector length over 60k) but sparse vectors (around 1% of non-zero values). Apart from the training vectors, the LightGBM trainer take other *hyperparameters* as an input. We target this topic in the rest of this work.

*Supported by the ERC Project *SMART* Starting Grant no. 714034, and by the Czech MEYS under the ERC CZ project *POSTMAN* no. LL1902.

[1] https://github.com/ai4reason/enigmatic/blob/master/enigmatic/lgbtune.py

2 LightGBM Hyperparameter Tuning

LightGBM supports few dozens of hyperparameters that influence the model training process. While many of them might be safely used with their default values, setting of some hyperparameters is crucial for success and to prevent overfitting. While there exist basic guidelines for setting of hyperparameters, setting them in practice often requires experience and understanding of the training process. There are, however, automated tools to search for suitable parameter values like Optuna [1] or FLAML [10] which support LightGBM. For our experiments with ENIGMA, we have developed our tool `lgbtune` to search for suitable values of LightGBM hyperparameters targeted to our specific ATP needs.

Our tool `lgbtune` is implemented in Optuna. Given the training data, it keeps a small amount of the training data (5%) for a later independent evaluation, and it trains the models only on the rest. Then it proceeds in phases when it tries to search optimal values for selected hyperparameters, different ones in each phase. In each phase, several values of tuned hyperparameters are tried, and the resulting models are evaluated on the part of input data not used for training. The best hyperparameter values are then fixed in the phases to follow.

Some of the hyperparameters are dependent, that is, the optimal value of one parameter might depend on the value of another one. For example, the optimal number of tree leaves might depend on the maximal tree depth. We tune dependent parameters together in one phase. In phase (1) we tune probably the most influential parameter, that is, the number of leaves (`leaves`) in a decision tree. We set the tree depth (`max_depth`) to unlimited and thus we eliminate one dependent parameter. In other phases we then tune (2) randomized feature sampling (`bagging_fraction`, `bagging_freq`), (3) the minimal number of data in leaves (`min_data`), and (4) L1/L2 regularization terms (`lambda_l1`, `lambda_l2`). Value selection mechanism and distribution are derived from Optuna.

3 Machine Learning and ATP Evaluation

In `lgbtune`, the quality of a model is estimated based on its accuracy on the shelved training data. This *ML evaluation* should correlate with the actual performance of the prover guided by the model, that is, with *ATP evaluation*. However, this is not always the case, because the training data are typically unbalanced, and more than 90% are typically negative training samples (clauses classified as *useless*). Hence it is crucial to compute separately accuracies on positive and negative testing samples. Moreover, it seems that the positive accuracy is even more important for ATP evaluation. This can be explained by the behavior of E, where accidental processing of a single useless clause does not need to do much harm, while postponing of the processing of a useful clause can effectively block any path to success. Hence we measure the quality of a model as $2\mathit{pos} + \mathit{neg}$, where *pos* and *neg* are positive and negative accuracies.

We have successfully used `lgbtune` during our recent ENIGMA experiments [4], where it helped us to increase the accuracy of models by more than 5%. In this presentation, we would like to present our tool, and, additionally, perform an extended evaluation and testing of our tool (the extended evaluation is currently in progress). We will evaluate the ATP performance of trained models and test our assumptions about the correlation of ML and ATP performance. Furthermore, we would like to evaluate the impact of a different phase orders during the tuning, and the importance of tuned parameters. This could help us to decide which phases should be given more time, and which phases should be removed. Finally, `lgbtune` is motivated by LightGBM plugins from Optuna and FLAML, and we would like to compare directly with their performance. All the experiments will be targeted to data from our ATP experiments.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM, 2016.

[3] Karel Chvalovský, Jan Jakubův, Martin Suda, and Josef Urban. ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. In *CADE*, volume 11716 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2019.

[4] Zarathustra Amadeus Goertzel, Jan Jakubuv, Cezary Kaliszyk, Miroslav Olsák, Jelle Piepenbrock, and Josef Urban. The isabelle ENIGMA. *CoRR*, abs/2205.01981, 2022.

[5] Jan Jakubův, Karel Chvalovský, Miroslav Olsák, Bartosz Piotrowski, Martin Suda, and Josef Urban. ENIGMA anonymous: Symbol-independent inference guiding machine (system description). In *IJCAR (2)*, volume 12167 of *Lecture Notes in Computer Science*, pages 448–463. Springer, 2020.

[6] Jan Jakubův and Josef Urban. ENIGMA: efficient learning-based inference guiding machine. In *CICM*, volume 10383 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 2017.

[7] Jan Jakubův and Josef Urban. Enhancing ENIGMA given clause guidance. In *CICM*, volume 11006 of *Lecture Notes in Computer Science*, pages 118–124. Springer, 2018.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3146–3154, 2017.

[9] Stephan Schulz. E - A Brainiac Theorem Prover. *AI Commun.*, 15(2-3):111–126, 2002.

[10] Chi Wang, Qingyun Wu, Markus Weimer, and Erkang Zhu. Flaml: A fast and lightweight automl library. In *MLSys*, 2021.