AutoML challenge: system description of Lisheng Sun

Lisheng Sun
CECILE829@GMAIL.COM
1-16-10-303, Shimizu, Suginami-ku, Tokyo, 167-0033, Japan

Abstract

Human efforts play an essential role in many successful machine learning projects, especially in the phase of selecting the right algorithms, designing good workflows and hyperparameter tuning. Usually these are done by hand, which might make the final solution specific to the problem, and also increases the difficulty for non-experts to use machine learning. The AutoML Challenge organized by ChaLearn aims to gather participants to provide their own solutions to this problem by studying various real world datasets. I proposed a general, case-separating and improved on-the-fly system, which, with a post submission, won the third place in the final phase (AutoML5) of the challenge. In this article, I will give a detailed description of this system (referred as AutoML-LSUN system).

Keywords: AutoML Challenge, machine learning, hyperparameter tunning

1. Description of AutoML-LSUN System

I constructed my AutoML-LSUN system using case separation based on task types: regression, classification (multi-class, multi-label). The workflow of the system is illustrated in fig. 1. It can be divided into 3 steps:

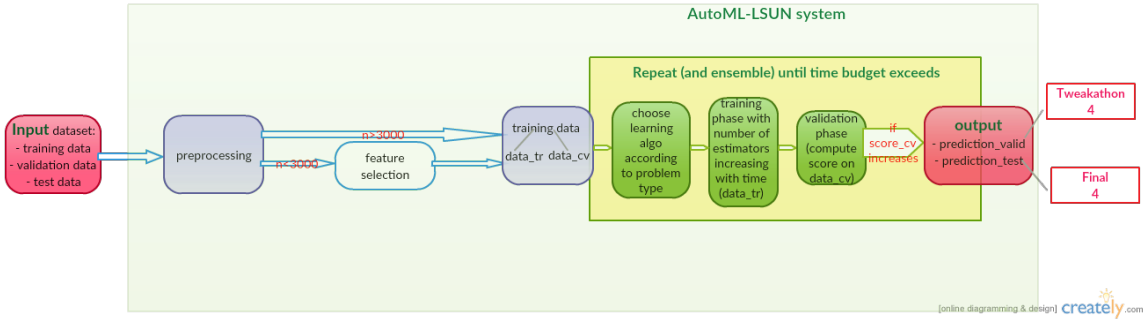

Figure 1: Workflow of AutoML-LSUN system. The system is case separating and improved on-the-fly.

1. **Step 1: Preprocessing**
   In this step, I address in particular the problem of missing values by replacing them with the mean along the axis.

2. **Step 2: Feature selection**
   The feature selection is applied only when the number of features is modest ($p < 3,000$) and the task type is neither binary classification nor multi-label classification (these 2 cases will be treated differently as described below). Any feature selection method I tried (variances-based, dispersion-based (4), regularization-based) slows down dramatically the whole system when the dimension is too high. I have experimented with different threshold values p_{th} and found $p_{th} = 3,000$ is appropriate.

   When applied,

   - For sparse data: I use Lasso to select features via L1-norm regularization (3).
   - For non sparse data: I first remove constant features, then select the remaining features according to a percentile of the highest F-value scores (3). The percentage is determined by a grid search.

3. **Step 3: Actual training and prediction**
   In this step, I do a case separation based on different task types. After Step 2, now the number of features is either $p > 3,000$ (no feature selection has been applied) or $p << 3,000$ (after feature selection). The training dataset is split into 2 subsets, one for training, the other for cross validation. As sparse data are usually of high dimension, I treat both similarly using linear models. In order to make full use of the time budget, I create a timer for each task with Python's multiprocessing (5). The system is trained on training subset, the number of estimators considered is increased as a function of time. The system performance is then evaluated on cross validation subset by computing for each dataset its task-specific score. An output is generated / updated if this score increases. Ensemble methods are also used in regression for a final result improvement.

   After many tests, I opted the Freeze-Thaw method (6) to deal with binary classification tasks. A Python implementation was developed by James Lloyd for the phase of AutoML1 (7). I modified this implementation and tested it on binary classification tasks of previous round. Since it works always well, I merged it into AutoML-LSUN system. This method is different from other parts of my system, in the sense that it treats the task as a whole problem of hyperparameter tuning without breaking it into feature selection phase and training / prediction phase. Based on bayesian optimization approche, the Freeze-Thaw method proposes to study the partial information provided by iterative training procedure and decide whether to pause ('freeze') that procedure and start a new one or to resume ('thaw') a partially-completed one.

For regression, I use Bagging with few estimators to get a quick output and improve it progressively, a final ensemble is performed just before the time budget to reduce loss. The base estimator of Bagging is chosen in function of the number of features: linear regression for high dimension and decision trees for lower dimension. For multi-class classification, instead of the final ensemble, I validate each iteration with the task-specific scoring function, and update the output predictions if the score increases. I also modified the MyAutoML system provided by the Starting-kit of the challenge and merged it into my system to treat multi-label classification problem, as well as any uncovered cases.

2. Discussion

Although the AutoML-LSUN system is far from being a fully automated machine, this competition taught me a lot. In particular, I learned how to quickly generate predictions with few base estimators when the task is constrained by a time budget. Also, I learned how to improve these predictions by increasing progressively the number of base estimators. More improvements on the system could be made, for example:

- Categorical variables: They are not treated in AutoML-LSUN system. One Hot Encoding can be used to get right information from these special variables.
- Warm start: I didn't use warm starter in AutoML-LSUN system, because it is not supported by the scikit-learn version used in this challenge. But this can avoid repeated calculation, which will release computation time for more learning iterations and is very likely to improve the result.
- More ensemble learning: More different learning models can be used to create separate output files. We can then generate a final result by ensembling all output files. Different ensemble methods can be tested: voting, averaging, or ensembling only the low-correlated predictions.

References

[1] Guyon, Isabelle, et al. "Design of the 2015 ChaLearn AutoML challenge." Neural Networks (IJCNN), 2015 International Joint Conference on. IEEE, 2015.

[2] ChaLearn Automatic Machine Learning Challenge (AutoML), 2016. https://competitions.codalab.org/competitions/2321#results

[3] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

[4] Ferreira, Artur J., and Mrio AT Figueiredo. "Unsupervised feature selection for sparse data." ESANN. 2011.

[5] G. van Rossum and F.L. Drake (eds), Python Reference Manual, PythonLabs, Virginia, USA, 2001. http://www.python.org

[6] Swersky, Kevin, Jasper Snoek, and Ryan Prescott Adams. "Freeze-thaw Bayesian optimization." arXiv preprint arXiv:1406.3896 (2014).

[7] Lloyd, James. "automl-phase-1." 2015. https://github.com/jamesrobertlloyd/automl-phase-1