Deep Learning Proposal - Learning to learn

Ruiqi Zhong, rz2383; Haoyan Min, hm2689; Yilan Ji, yj2425

November 2017

1 Introduction and Goal

In recent years the combination of reinforcement learning and deep learning has led to significant technological breakthroughs. More profoundly, programs starting with zero knowledge can learn from simulated environments and achieve super-human performance[1][2][3]. On the other hand, big data and machine learning are more widely employed by the industry and there is an increasing demand for data analysts; nevertheless, many of their works are routine (e.g. parameter tuning). Can we train a meta-learning program that can automatically and efficiently find an optimal machine learning model?
Previous works have employed Bayesian optimization techniques (with or without labeled meta-data) and achieved respectable performance[4][5][6]. However, they fail to take into account other important meta-data such as time required to train a model, training accuracy, etc while searching for the optimal model. In this project, we attempt to take these factors into account and use reinforcement learning and RNN to generate a meta-learning program composed of scikit-learn primitives that can quickly and efficiently search for a near optimal model m^* on data set D. (see more rigid definition of an "optimal model" "CASH problem" here[7]). Such a program will not only validate experiences and techniques employed by data analysts, but also help us discover new ones.

2 High Level Formulation and Algorithm

We will use deep Q lerning, a state-of-the-art reinforcement learning algorithm to train such a meta-model/program. This task can be formulated as a reinforcement learning problem as follows. Suppose that we want to find an optimal model m^* on data set D. Each action a_t is the t^{th} line of code/t^{th} hyper-parameter to experiment with that is chosen by our program; correspondingly, a model m_t will be trained on the data set D, and its performance statistics is denoted as $P(m_t)$. Each s_t includes all the past actions chosen ($a_{1:t-1}$) and their corresponding performance statistics $P(m_{1:t-1})$(training error, testing error, f-score, etc) on the data set D; the reward r_t is the improvement in accuracy (included in the performance P_{m_t}) "minus" the time τ_t it takes to train m_t. We will use a low dimension continuous real vector h_t to represent s_t, and $s_{t+1} = H(s_t, P(m_t), a_t)$, the value of action a_t is $Q(s_t, a_t)$, and reward $r_t = R(P(m_t, \tau_t))$. After modelling Q, H with neural networks, we employ ϵ-greedy approach to generate a policy based on $Q(s_t, a_t)$, use Monte-Carlo or Temporal Difference[1] to generate training data, and update the network Q. Q can be perfectly modeled by LSTM: the inputs are P_{m_t}, a_{t-1}, output a_t and memory cells representing the state s_t

3 Data Set and Evaluation Criteria

3.1 Data Set

In this project, we will focus on binary classification only. However, we are not limited to only this one task: there are more fine-grained tasks, such as optimizing accuracy/f-score/roc area, etc.
1) Preliminary: Synthetic data set with some distribution assumptions and simulated noise.
2) Advanced[8]: OpenML datasets : https://www.openml.org/s/14/data

3.2 Evaluation Criteria

The comparisons to the baseline would be conducted as follows: given the same amount of time, we compare the best model discovered by our program and our adversary, respectively.
Preliminary: we will first implement the framework of Q-learning and focus on a few simple and quick classification algorithms and a limited set of prepossessing steps (2 or 3). We hope that our model will 1) outperform randomized search 2) Bayesian optimization (if possible). We also hope to see that our program learn by itself some common sense knowledge of machine learning, for example, if training accuracy is significantly better than testing accuracy, it might be better to further regularize the model.
Advanced: include a larger set of algorithms and hyper-parameters and compare our generated program with auto-sklearn

References

[1] Silver, David, et al. "Mastering the game of go without human knowledge." Nature 550.7676 (2017): 354-359.

[2] Andrew, Alex M. "REINFORCEMENT LEARNING: AN INTRODUCTION by Richard S. Sutton and Andrew G. Barto, Adaptive Computation and Machine Learning series, MIT Press (Bradford Book), Cambridge, Mass., 1998, xviii+ 322 pp, ISBN 0-262-19398-1,(hardback,£ 31.95).-." Robotica 17.2 (1999): 229-235.

[3] Hessel, Matteo, et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning." arXiv preprint arXiv: 1710.02298 (2017).

[4] Feurer, Matthias, et al. "Efficient and robust automated machine learning." Advances in Neural Information Processing Systems. 2015.

[5] MLA Brochu, Eric, Vlad M. Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning." arXiv preprint arXiv:1012.2599 (2010).

[6] Swersky, Kevin, Jasper Snoek, and Ryan P. Adams. "Multi-task bayesian optimization." Advances in neural information processing systems. 2013.

[7] C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In Proc. of KDD'13, pages 847–855, 2013.

[8] Vanschoren, Joaquin, et al. "OpenML: networked science in machine learning." ACM SIGKDD Explorations Newsletter 15.2 (2014): 49-60.