Hypersearch: A Parallel Training Approach For Improving Neural Networks Performance

Anonymous Author
Anonymous Institution
Anonymous Region
anon.email@domain.com

Anonymous Author
Anonymous Institution
Anonymous Region
anon.email@domain.com

Abstract

Neural Networks is one of the most vastly used AI (machine learning) techniques. Unfortunately it relies greatly on high processing power to achieve great results within reasonable time frame. This paper proposes a novel parallel algorithm, Hypersearch, that optimizes their performance by taking advantage of both worlds of hyperparameter optimization and parallel programming. Our results show that Hypersearch not only performs well, but is also competitive against state-of-the-art Hyperparameter optimization alternatives. More specifically we show that it performs better than Hyperopt-TPE (Tree-structured Parzen Estimator).

1 Introduction

Hyperparameter Optimization refers to any procedure which finds hyperparameters that maximizes performance (Bergstra et al., 2011). This is usually done manually (hand tuning) by someone actually trying different hyperparameter choices and using the performance due to each to guess the next better choice to try. The obvious problem with this is that it takes too long (depending on neural network and data size) and usually requires expert knowledge of neural networks and the problem domain to make sensible guesses. Hence a lot of research has gone into automatic methods, some of the state-of-the-art ones being: HyperOpt which uses Bayesian Optimization with TPE (Bergstra et al., 2013), Spearmint which uses Bayesian Optimization with GP (Eggensperger et al., 2013), SMAC which uses Bayesian Optimization with a custom modeling function (Thornton et al., 2013), and Hyperband which uses a Random Search Bandit-Based Aproach (Li et al., 2017).

This work introduces an easily parallelizable algorithm called Hypersearch that attempts to improve neural networks performance faster than current state-of-the-art hyperparameter optimization methods, by simultaneously optimizing the parameters and hyperparameters for a given neural network architecture. We only evaluate a sequential implementation of Hypersearch, leaving the parallel or distributed implementation for future work.

2 Hypersearch Algorithm

Given a neural network N, a hyperparameter space H, a maximum number of epochs M, a number of sampled hyperparameters per session S and a number of sessions m, the algorithm works as follows:

1. Samples S hyperparameters from H.
2. Parallely train N with the S hyperparameters for $\lfloor \frac{M}{m} \rfloor$ epochs (where $m \leq M$). Let h be the hyperparameter of the neural network with best convergence.

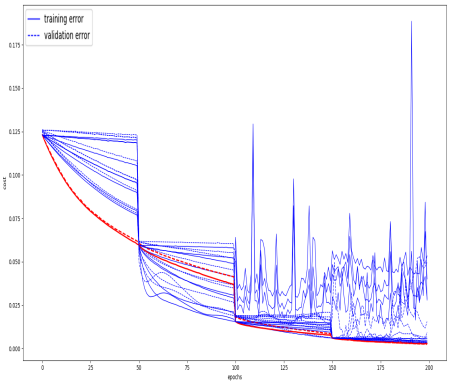
Black in AI workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

3. Samples $S-1$ hyperparameters in the neighbourhood of h.
4. Discards the $S-1$ under-performing neural networks and replaces them with $S-1$ new neural networks initialized with the best neural networks parameters and respectively assigned the sampled $S-1$ hyperparameters.
5. Repeat steps 2 to 4 $m-1$ times.
6. Return best neural network(s).

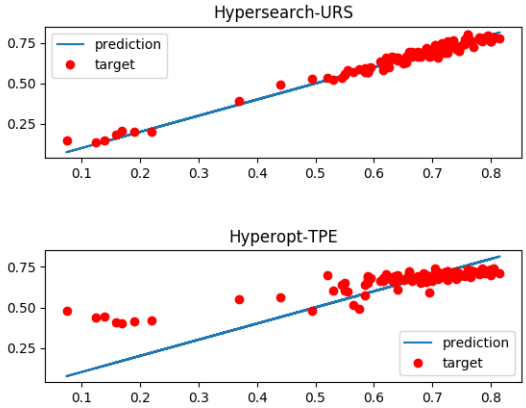
Assuming constant time for choosing best network and selecting next hyperparameters (true if using least training error and random sampling respectively), the time complexity of Hypersearch is hence $\mathcal{O}(S*M)$. This is the same as that of normal hyperparameter optimization where a neural network is trained S times with different hyperparameters for M epochs.

3 Results

Figure 1a shows the behaviour of the training and validation losses when using Hypersearch to train neural networks on a given regression problem. Hypersearch achieves extremely low losses quickly thanks to its simultaneous optimization of parameters and hyperparameters. This demonstrates that Hypersearch is a useful method in practice for automatically finding the best performing neural network of a given architecture. Hypersearch and Hyperopt are then compared on a regression problem by running both methods with the same initial conditions (network architecture, parameters initialization, hyperparameters space, optimizer, etc). Figure 1b shows the test data predictions of the best network produced by each method, and is representative of multiple runs of the experiment. We observe that Hypersearch outperforms Hyperopt.



(a) Performance of neural networks: $S=10$, $m=4$



(b) Predictions of best neural network on test data

Figure 1: Hypersearch with uniform random sampling (URS) of next hyperparameters and comparison with Hyperopt-TPE

4 Conclusion

Hypersearch performs better than Hyperopt-TPE, with a secondary advantage being that at the end of training it has numerous neural networks with similar performance. Hence the popular technique of averaging the predictions of multiple networks can be applied here possibly providing better results. It is also worth noting that the goal of Hypersearch is to find the best neural network of a given architecture, not just the best hyperparameters. Hence its final best hyperparameters may not always be optimal (their good performance during training is mainly as a result of the best parameters initialization in each Hypersearch session).

Other future work could include more thorough benchmarking (with other state of the art hyperparameter optimizers) and an actual parallel or distributed implementation of Hypersearch that takes full advantage of its high parallelism.

References

James Bergstra, Dan Yamins, and David D Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*. Citeseer, 13–20.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*. 2546–2554.

Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. 2013. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, Vol. 10. 3.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* 18, 1 (2017), 6765–6816.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 847–855.