Advanced Topics in Distributed Systems

Task 4: Paper Review Report

Moritz Meister
March 29th 2020

The purpose of this report is to give a critical review of three papers in the context of distributed systems with respect to their contributions, solutions, significance, and technical/experimental quality.

Justification

The following papers were chosen:

1. Jaderberg, M. et al., 2017. *Population based training of neural networks.* (DeepMind)
2. Falkner, S. et al., 2018. *BOHB: Robust and Efficient Hyperparameter Optimization at Scale.* (Published at ICML 2018)
3. Li, L. et al., 2020. *Massively Parallel Hyperparameter Tuning.* (Published at MLSys 2020)

With the following justification:

> In order to adequately explore the large hyperparameter spaces of modern learning models, it is necessary to evaluate a large number of configurations. Given the growing costs of model training, we would ideally like to perform this search as efficiently as possible and in parallel. In this session, we will discuss and compare three asynchronous search algorithms that can be deployed in parallel. Asynchronous algorithms have the advantage that they are robust against stragglers, avoid idle workers and can be combined with extensive early stopping, making them more efficient. However, these meta-level search algorithms introduce additional hyperparameters, for example the number of resources allocated to the training of each configuration, leading to a trade-off between exploration and exploitation. We want to see how the authors of the three papers design their experiments to guarantee comparability of the algorithms.

Background

Modern machine learning (ML) models are computationally expensive and come with a great number of hyperparameters, which influence the final performance of the model. This has resulted in a resurgence of research around hyperparameter optimization (HPO) and more generally in automated machine learning (AutoML). It is a fruitful research topic because it promises to

1. Reduce the amount of human effort for using ML
2. Improve the performance of ML algorithms
3. Improve reproducibility and fairness of scientific studies. (automated search is more reproducible than manual search)

Also commercially, due to the vast amount of resource savings possible, HPO has created a number of cloud services (such as SigOpt or h2o.ai) aiming to provide solutions for the problem.

HPO is in fact a black-box optimization problem and therefore has a long history of research, the novelty added to it recently is the amount of available compute resources. Black box optimization is hard in practice and has the following properties:

1. The function evaluation, in this case the training of a ML model, is extremely expensive.
2. The search space is high dimensional and complex, possibly conditional.
3. There is no gradient information over the search space of hyperparameters available.
4. The feedback generated by the model is only an approximation to the true generalization performance as training datasets cover only a part of reality and are limited in size.

The three papers selected are trying to tackle this problem from different angles, with undirected search, directed search and multifidelity methods, while at the same time keeping scalability has a main goal.

PBT: Population based training of neural networks

Motivation

The success of a neural network in a particular application is too often determined by a series of choices made at the start of the research, such as what type of network to use or the data and method used to train it. Furthermore, current HPO methods require wall clock times greater than that of a single optimization process.

Contributions

The solution proposed in this paper has two contributions: The first contribution is PBT, a simple algorithm to **jointly** optimize a population of models **and** their hyperparameters to maximize their performance, returning the best model of the population, fully trained. This can be done on a fixed computational budget and in the wall clock time of training a single model as hyperparameters are selected automatically during training. The second contribution lies in the rather novel approach of learning a complex schedule for hyperparameters instead of optimizing for a static hyperparameter configuration, which is generally suboptimal.

Solution

The authors distinguish between two common HPO techniques: parallel search and sequential optimization. They assume that in parallel search, configurations are trained independently and sampled undirectedly, while in sequential optimization the information gained by prior trials is used to improve performance gradually. As a remark, one should note that this conception of parallel search is not considered state-of-the-art. There exist methods that are able to run in parallel and in a directed fashion nowadays, which will be seen in the discussion of the BOHB paper.

PBT combines parallel search with sequential directed search with inspiration taken from genetic algorithms. PBT starts by training many neural networks in parallel with random hyperparameter configurations. Throughout the process of training, however, it utilizes the information of the entire population to refine the parameters and allocate computational resources to the promising models. This is done through two alternating steps of exploration and exploitation. To explore, a worker can swap its hyperparameters randomly with new ones throughout training, while not restarting the training process, if it notes that it’s performing worse than the population. For exploitation, it can swap the learned weights, i.e. its entire model with another better performing worker. Through these two steps, PBT can quickly exploit good hyperparameters, dedicate more time to promising models and, crucially, adapt the hyperparameters throughout training, leading to automatic learning of the best configurations.

Technical/experimental quality

The experiments conducted show the effectiveness of PBT across a wide range of tasks, focusing on reinforcement learning, machine translation and generative adversarial networks. They chose these areas because they were shown to be inherently non-stationary and become very unstable with a suboptimal setting of hyperparameters. In the case of GANs, the generators collapse or diverge easily. The same holds for RL, where policies easily end up in suboptimal local optima or collapse. Different methods for the exploitation step are being tested, that is, binary tournaments and truncation selection. Truncation selection works well for GAN training, however, they observe that overall truncation selection is the most robust and useful choice. Instead of comparing their method to other HPO methods, the authors opted to compare their results only to previous hand-tuned state-of-the-art results. This makes a fair comparison, as they do not have to take into account the tuning of the hyperparameters of the HPO method itself, which often limits the comparability. Most notable, PBT stabilizes quickly, and delivered performances that beat the previous state-of-the-art baselines on all the selected problems. All of this is being done in the wall clock time of a single training run. However, one has to keep in mind that even though it's a single training run, they used up to 32 GPUs in parallel.

A positive aspect of their evaluation is the discussion of the additional introduced hyperparameters, such as population sizes, exploitation types, PBT targets (using only exploitation or only exploration), hyperparameter adaptivity or lineage. Surprisingly, they find that PBT deploys also a non-monotonic dropout rate regime, which is usually not used by experts. This hints at the fact that PBT might also discover novel approaches to be considered by experts.

One weak point of the paper is the vagueness of the authors when it comes to the amount of resources used for the different tasks. While they mention the number of GPUs used for the machine translation task, they only speak about the size of the population for the other tasks, where each member is trained on a separate worker. They do not specify the resources of such a worker.

BOHB: Robust and Efficient Hyperparameter Optimization at Scale

Motivation

The authors of the BOHB paper argue for the importance of research in this area with the sensitivity of ML models to many hyperparameters, computational infeasibility of vanilla Bayesian optimization and the goal to automate HPO.

Contributions

The contribution of this publication lays in the practicality of the proposed method and its wide applicability to problem and model types. The authors show state-of-the-art results on models such as high-dimensional toy functions, support vector machines, feed-forward neural networks, Bayesian neural networks, deep reinforcement learning and convolutional neural networks. To define their contributions, the authors introduce five desiderata, that a practical HPO solution should provide:

1. Strong Anytime Performance: meaning given a low computational budget the method should find a good configuration.
2. Strong Final Performance: meaning given a larger budget, the method should be able to converge to a good configuration.
3. Effective Use of Parallel Resources.
4. Scalability in terms of the number of hyperparameters and search space dimensions
5. Robustness and Flexibility. Depending on the problem at hand, the hyperparameters might be of differing importance and types, a practical HPO method should be robust enough to cover most of them.

While their method covers all these aspects, this classification for HPO methods can also be seen as a contribution and should find consideration in future research.

Solution

Hyperband (HB) is a multi-fidelity method based on a multi-armed bandit strategy, which approximates the objective function (generalization performance of the model) with cheap evaluations of the function on smaller budgets. HB repeatedly calls Successive Halving (SH) to identify the best samples out of *n* randomly sampled configurations. In a nutshell, SH evaluates n configurations with a small budget, and promotes the best half while doubling their budget until only one configuration is left.

The authors propose with their solution to replace the undirected random sampling with Bayesian sampling in order to achieve better final performance. Bayesian optimization uses a probabilistic model for the objective function based on previously observed realisations. With an acquisition function one can then trade off exploration vs exploitation to draw new samples. The authors use a Tree Parzen Estimator (TPE) as their method of choice, in contrast to previous solutions, which used Gaussian Processes (GP). They argue for the TPE, as the better solution, as it supports mixed continuous and discrete search spaces and the model computation scales linearly in the number of observations, compared to cubic-time for GPs.

Technical/experimental quality

As mentioned before, a wide range of experiments are performed on different model types and state-of-the-art performance is shown in all of them (at the time of publication). There are a few things to note which make the experimental quality overall high. First of all, it is noted that the overall goal is to optimize on the largest budget, and hence the model should always be computed on the largest budget for which enough observations are available. This is important because the authors take the same approach with the other methods, which might not use multiple fidelities, that they compare to. For example all models in random search are also trained on the largest budget. This way a fair comparison is guaranteed. Other authors often omit the detail of budget allocated to non fidelity methods. HyperBand has some strong theoretical guarantees, and with the experimental evaluation it is shown that they can be kept for BOHB by sampling a fraction of configurations at random, while only being slower by a constant factor, without sacrificing final performance. Furthermore, what’s positive is that evaluations are performed with respect to the additional hyperparameters introduced by the HPO algorithm itself. Experimentally it is shown that BOHB is insensitive to its own hyperparameters. In comparison to other paper which often do not state how certain configurations of the meta-hyperparameters are found and how much resources were used to train these.

One final remark is that the authors opted to use a surrogate model to predict the performance of models given a configuration. This way the method becomes more practical as not every model needs to be trained to completion. This surely introduces some inaccuracy, but the final performances reported in the paper are always the true metrics after trained to completion. Hence, the method still converges to good optima, outperforming previous state-of-the-art.

The main weakness of the approach is the brute-force nature of looping over different budget brackets and using SH multiple times. Other publications find that the most aggressive stopping bracket usually performs best. The authors of BOHB do not mention this possibility. Hence, empirically SH with Bayesian optimization might perform just as well as BOHB but with less resources required.

ASHA: Massively Parallel Hyperparameter Tuning

Motivation

The motivation for the paper is presented in three recent trends in modern machine learning, based on which the authors argue for new approaches to hyperparameter tuning:

1. High-dimensional hyperparameter spaces
2. Increasing training times
3. Rise of parallel computing

Summarizing these, they define the large-scale regime for HPO, for which the goal is to "*evaluate orders of magnitude more hyperparameter configurations than available parallel workers in a small multiple of the wall-clock time needed to train a single model.*"

Contributions

The paper itself names five contributions for the large-scale regime for HPO. However, really there are two main contributions, while the other three are more results of the first two and therefore supporting the contributions.

1. Asynchronous Successive Halving Algorithm (ASHA), a simple and practical method to exploit parallelism and aggressive early-stopping.
2. Experimental results, showing that the most aggressive budget bracket of BOHB usually performs best, therefore motivating the use of SHA only, but extending it to the parallel setting.

Solution

Let's revisit SHA briefly to set the context. The successive halving algorithm samples a number of candidate configurations in the so called base rung and proceeds with the following steps:

1. Given a total budget R (e.g. iterations, compute resources), allocate the budget uniformly to the set of configurations in a given rung.
2. Evaluate all candidates in the rung.

3. Promote the top half of the candidates to the next rung based on some performance metric such as accuracy or loss.
4. By doubling the budget per configuration for the next rung the budget is allocated uniformly again. Now repeat the steps until only one configuration remains.

Simply implementing this in an embarrassingly parallel fashion is not applicable to the large scale regime, as a new rung can only begin once all trials in the previous rung finished.

Instead, the authors remove this bottleneck of synchronous promotions, by allowing the algorithm to grow from the base rung up and promote configurations whenever possible. ASHA starts by letting workers evaluate randomly drawn configurations for the base rung. When a worker finishes and requests a new trial, the algorithm looks at the runs from top to bottom to see if there are trials in the top ½ (or ⅓, this is actually a new hyperparameter) of each rung that can be promoted to the next rung. If not the worker will continue adding configurations to grow the base rung until eventually configurations can be promoted.

Technical/experimental quality

ASHA is being empirically evaluated on four different tasks. The first task is to compare ASHA to SHA, PBT, HyperBand and BOHB in the sequential setting, in order to justify the modifications to SHA. The notable observation for this experiment is that SHA performs at least as good as BOHB and PBT and outperforms simple HyperBand. This leads to the conclusion of the authors that the budget brackets with the most aggressive early stopping rates usually perform best. Hence, as HyperBand and BOHB mostly loop over different brackets in a brute force fashion, they incorporate an inefficiency in this point. The second observation is that adding asynchrony does not penalise final performance, and hence is applicable for the large scale regime.

The second experiment is a limited scale distributed neural architecture search (NAS) and HPO task. Thirdly, ASHA is being evaluated in a large scale setting with 500 workers for a machine translation task against only HyperBand and a Google Service called Vizier for HPO. Lastly, the authors compare ASHA to PBT on a modern LSTM benchmark based on the Penn Tree Bank dataset. Also in these tasks, ASHA performs at least as good as its competitors. The strength of ASHA lies in tasks where a large fraction of configurations is performing very badly, which is given for NAS. Given the fact that ASHA is a simple algorithm, the results are promising.

The authors make three interesting observations in the analysis of the algorithm, which are also the strong points of the algorithm, additionally to its simplicity. 1. They can provide an upper bound on the time to return a configuration trained on completion, which is two times the time to train on the max budget (*2*time(R)*) and can possibly be reduced to *time(R)* if training of promoted trials can be resumed instead of restarted from scratch. 2. The bottleneck of synchronous promotions can be removed at the cost of a small number of incorrect promotions. 3. The algorithm is applicable in an infinite horizon setting, as experiments can simply be continued with more rungs as more resources (compute time) become available.

The set up of the experiments shows some weaknesses. The authors opted to compare only to a selected of other methods in some of the experiments without mentioning the reasons for that. Furthermore, the additional hyperparameters introduced by ASHA, such as stopping rates and budget allocations , as well as their effects on the results are not discussed in great depth. In order to guarantee a fair comparison, the same effort for tuning the HPO algorithm itself should be used, but it is not clear how the settings for the different algorithms were chosen.

Final Remarks

All three papers present some interesting aspect to HPO. As per the knowledge of the reviewer, PBT is the first publication to propose an optimization method to produce non-stationary hyperparameter schedules. BOHB presents a theoretically sound method with wide applicability, shown through thorough experimentation with a surrogate model to make experimentation affordable. On the other hand, ASHA presents itself as a simple and practical approach, which does not require maintaining a big state for the model itself and can possibly be extended by a directed sampling method such as Bayesian optimization to replace the random sampling part.