A resource-efficient method for repeated HPO and NAS problems
Giovanni Zappella, David Salinas, Cédric Archambeau
Amazon Web Services

Repeated HNAS

Data scientists do not create machine learning models in a single shot. They try different features, transformations, target variables, etc. During this procedure they often avoid using running HPO and NAS procedures due to their high cost and long execution time. At the same time, hyperparameter configurations often perform similarly if the learning tasks on which they are used are similar. However, defining the similarity between learning tasks is far from trivial and it is often impossible for non-expert users.

Can we speedup HNAS when it is performed on a sequence of related tasks?

Setting

We formalize the problem as a sequence of “Best arm identification” (BAI) problems. Each hyperparameter configuration (or architecture) is an arm in the bandit problem and the goal of the learner is to identify the one with the highest reward.

Every BAI problem has an optimal arm and we assume that the arms previously identified as optimal will be quickly outperformed by other configurations if they are not the best arm for the problem at hand.

Our Solution: RUSH

Input: $\eta$ (halving hyper-parameter), $B$ (budget)
$s = 0$

while a new task is available do

$A_{s+1}^n \leftarrow$ set of new arms
$n \leftarrow |A_{s+1}^n|

for $k = 0, \ldots, \lfloor \log_2 n \rfloor - 1$ do

forall $a \in A_{s+1}^n$ pull it $\max(1, \lfloor n^{k+1}/2 \rfloor)$ times

$r_a = \min(r(e), a \in A_{s+1}^n)$

end for

$\hat{a} \leftarrow$ best arm from $A_{s+1}^n$ such that

if $\hat{a} \notin A_s^n$ then

$A_s^n \leftarrow A_s^n \cup \{\hat{a}\}

end if

$s \leftarrow s + 1

end while

Guarantees

Theorem 1 If the budget $B$ provided to the algorithm for each step of the sequence $1, \ldots, S$ is larger than $\lfloor \log n \rfloor \max\left(2n + \sum_{e=2}^{\lfloor \log_2 n \rfloor - 1}(\Delta_e/2), zn\right)$ then RUSH will correctly identify the best arm.

We can guarantee that when the budget is “large enough”, RUSH will identify the best arm. (see the paper for all the details).

Experimental Results: Predictive Performance

As first step, we would like to verify that our algorithm can identify configurations leading to models with competitive predictive performance.

Table 1: Average prediction performance obtained by different optimizers. For BANK and CXDEFAULT we report 1-F1, for COTYPE we report 1-F1micro, for FCNET and NAS201 the prediction error. In all cases lower is better.

Experimental Results: Number of Evaluations

Since RUSH performs an aggressive pruning by leveraging the optimal configurations identified over the sequence, it save resources by performing less evaluations at higher resource levels.

Experimental Results: Time Gained

Our algorithm does not explicitly account for the different cost associated to different hyperparameters configurations. However, the total time necessary for the tuning is related to the waiting time and often, especially in cloud environments, to cost.

In the following table we report the time gained (in percentage) by using RUSH instead of Successive Halving and Hyperband.

References