A simple dynamic bandit algorithm for hyper-parameter tuning

Xuedong Shang, Emilie Kaufmann, Michal Valko

6th ICML Workshop on Automated Machine Learning (2019)

A simple dynamic bandit algorithm for hyper-parameter tuning

Xuedong Shang XUEDONG.SHANG@INRIA.FR
SequeL team, INRIA Lille - Nord Europe, France

Emilie Kaufmann EMILIE.KAUFMANN@UNIV-LILLE.FR
CNRS & ULille, UMR 9189 (CRIStAL), SequeL team, INRIA Lille - Nord Europe, France

Michal Valko VALKOM@GOOGLE.COM
DeepMind Paris

Abstract

Hyper-parameter tuning is a major part of modern machine learning systems. The tuning itself can be seen as a sequential resource allocation problem. As such, methods for multi-armed bandits have been already applied. In this paper, we view hyper-parameter optimization as an instance of best-arm identification in infinitely many-armed bandits. We propose `D-TTTS`, a new adaptive algorithm inspired by Thompson sampling, which *dynamically* balances between refining the estimate of the quality of hyper-parameter configurations previously explored and adding new hyper-parameter configurations to the pool of candidates. The algorithm is easy to implement and shows competitive performance compared to state-of-the-art algorithms for hyper-parameter tuning.

1. Introduction

Training a machine learning algorithm often requires to specify several parameters. For instance, for neural networks, it is the architecture of the network and also the parameters of the gradient algorithm used or the choice of regularization. These *hyper-parameters* are difficult to learn through the standard training process and are often manually specified.

When it is not feasible to design algorithms with a few hyper-parameters, we opt for *hyper-parameter optimization* (HPO). HPO can be viewed as a *black-box optimization* problem where the evaluation of the objective function is expensive as it is the accuracy of a learning algorithm for a given configuration of hyper-parameters. This vastly limits the number of evaluations that can be carried out, which calls for a design of efficient high-level algorithms that automate the tuning procedure.

Several naive but daily used HPO methods are grid search and random search. More sophisticated methods address HPO as a *sequential resource allocation problem*, by adaptively choosing the next hyper-parameter(s) to explore, based on the result obtained previously. For example, *evolutionary optimization* follows a process inspired by the biological concept of *evolution*, which repeatedly replaces the worst-performing hyper-parameter configurations from a randomly initialized population of solutions; see Loshchilov and Hutter (2016) for an example of using `CMA-ES` for hyper-parameter tuning. A major drawback of evolutionary optimization is its lack of theoretical understanding.

Bayesian optimization (BO) is another approach that leverages the sequential nature of the setting. BO depends on a prior belief for the target function, typically a Gaussian

process. This prior distribution can be updated to a posterior given a sequence of observations. Several algorithms exploiting this posterior distribution to decide where to sample next have been given (see Shahriari et al., 2016, for a survey). Snoek et al. (2012) and Klein et al. (2017) provide Python packages called `Spearmint` and `RoBO` to perform hyper-parameter tuning with BO methods. Similar packages are available for `PyTorch` (`BoTorch`[1]) and `TensorFlow` (`GPflowOpt` by Knudde et al. 2017). Among BO algorithms, `TPE` (Bergstra et al., 2011) and `SMAC` (Hutter et al., 2011) were specifically proposed for HPO. A shortcoming of BO is that most algorithms select where to sample next based on optimizing some *acquisition function* computed from the posterior, e.g., the expected improvement (Jones et al., 1998). This auxiliary task cannot be solved analytically but needs to be performed itself by optimization procedures as `L-BFGS` that make the process slow.

Bandits (see Lattimore and Szepesvári, 2019 for a recent book) are a simple model for sequential resource allocation, and some bandit tools have already been explored for global optimization and HPO: First, in the field of Bayesian optimization, the `GP-UCB` algorithm (Srinivas et al., 2010) is a Gaussian process extension of the classical `UCB` bandit algorithm (Auer et al., 2002). Later, Hoffman et al. (2014) proposed to use *best-arm identification* (BAI) tools—still with a Bayesian flavor—for automated machine learning, where the goal is to smartly try hyper-parameters from a pre-specified *finite* grid.

However, in most cases, the number of hyper-parameter configurations to explore is infinite. In this paper, we investigate the use of bandit tools suited for an *infinite* number of arms. There are two lines of work for tackling a very large or infinite number of configurations (arms). The first combines standard bandit tools with a hierarchical partitioning of the arm space and aims at exploiting the (possibly unknown) smoothness of the black-box function to optimize (Bubeck et al., 2010; Grill et al., 2015; Shang et al., 2019; Bartlett et al., 2019). To the best of our knowledge, these methods have never been investigated for HPO. The second line of work does not assume any smoothness: At each round, the learner may ask for a new arm from a *reservoir distribution* ν_0 (pick randomly a new hyper-parameter configuration) and add it to the current arm pool $\mathcal{A}$, or re-sample one of the previous arms (evaluate configuration already included in $\mathcal{A}$), in order to find an arm with a good mean reward (i.e., a hyper-parameter configuration with a good validation accuracy). The *stochastic infinitely many-armed bandits* (SIAB) is studied by Berry et al. (1997); Wang et al. (2008) for the rewards maximization problem while Carpentier and Valko (2015); Aziz et al. (2018a) study the simple regret problem, which is related to BAI. While most proposed algorithms consist of querying an *adequate* number of arms from the reservoir before running a standard BAI algorithm, Li et al. (2017) propose a more robust approach called `Hyperband` that uses several such phases.

In this paper, we go even further and propose the first *dynamic* algorithm for BAI in SIAB, that at each round, may either query a new arm from the reservoir or re-sample arms previously queried. Our algorithm leverages a Bayesian model and builds on the top-two Thompson sampling (`TTTS`) algorithm by Russo (2016). An extensive numerical study is presented to show the competitiveness of our algorithm with respect to state-of-the-art HPO methods.

1. https://botorch.org/

2. Hyper-parameter optimization

In this paper, we view HPO as a particular *global optimization* setting, for which the target function f is a mapping from a hyper-parameter configuration to some measure of failure for the machine learning algorithm trained with these hyper-parameters. Formally, we aim at solving an optimization problem of the form $f^\star = \min\{f(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in \Omega\}$, where $\boldsymbol{\lambda}$ denotes a configuration of hyper-parameters chosen from a configuration space Ω. A hyper-parameter optimizer is a sequential procedure, that at each round t, selects a configuration $\boldsymbol{\lambda}_t$ to evaluate using some sampling rule, after which a (costly and *noisy*) evaluation of $f(\boldsymbol{\lambda}_t)$ is observed. Besides, a hyper-parameter configuration $\widehat{\boldsymbol{\lambda}}^\star$ is recommended as a guess for a close-to-optimal configuration at the end. The hope is that $f(\widehat{\boldsymbol{\lambda}}^\star)$ is not far from $f^\star$.

We restrict our presentation to hyper-parameter tuning for supervised learning algorithms. Given a training dataset $\mathcal{D}_{\texttt{train}}$ containing n labeled examples in $\mathcal{X} \times \mathcal{Y}$ and a choice of hyper-parameter configuration $\boldsymbol{\lambda}$, a supervised learning algorithm (neural network, `SVM`, gradient boosting, ...) produces a predictor $\widehat{g}_{\boldsymbol{\lambda}}^{(n)} : \mathcal{X} \to \mathcal{Y}$. Note that there can be some randomness in the training process (e.g., if stochastic gradient descent is used) so that $\widehat{g}_{\boldsymbol{\lambda}}^{(n)}$ may still be random for a given training set and hyper-parameters. The goal is to build a predictor that generalizes well. If we had access to the distribution $\mathbf{P}$ that generated the data (i.e., assuming that data points in $\mathcal{D}_{\texttt{train}}$ are i.i.d. from $\mathbf{P}$), this generalization power would be measured by the risk $f(\boldsymbol{\lambda}) \triangleq \mathbb{E}\left[\ell\left(\boldsymbol{Y}, \widehat{g}_{\boldsymbol{\lambda}}^{(n)}(\boldsymbol{X})\right)\right]$, where ℓ is some loss function measuring the distance between two predictions and the expectation is taken on $(\boldsymbol{X}, \boldsymbol{Y}) \sim \mathbf{P}$ and the possible randomness in the training process.

In practice, however, the explicit evaluation of f is impossible, but there are several methods for *noisy evaluations*. We can either compute the validation error of $\widehat{g}_{\boldsymbol{\lambda}}^{(n)}$ on a held-out validation set, $1/|\mathcal{D}_{\texttt{valid}}| \sum_{i=1}^{|\mathcal{D}_{\texttt{valid}}|} \ell(\widehat{g}_{\boldsymbol{\lambda}}^{(n)}(\mathbf{x}_i), \mathbf{y}_i)$, or a cross validation error over the training set as an approximation of the objective.

3. Active Thompson sampling for HPO

HPO can be cast as a BAI in bandits. However, standard algorithms are not directly applicable since the search space is often continuous. We thus turn our attention to BAI an infinitely many-armed bandits; see Appendix A for a detailed discussion.

In this section, we introduce a new algorithm for BAI in an infinite bandit model, that is an adaptation of Thompson sampling (Thompson, 1933). Thompson sampling can be seen as the very first bandit algorithm ever proposed, but has been used for the *rewards maximization* objective, which is quite different from BAI, as explained by Bubeck et al. (2009). Instead of using vanilla Thompson sampling, we build on `TTTS` that is an adaptation of Thompson sampling for BAI in finite-arm bandits. Unlike the state-of-the-art algorithm `SequentialHalving` that requires the knowledge of the total budget to operate, `TTTS` is particularly appealing as it does not need to have it. Such algorithms are referred to as *anytime*. Besides, it is known to be optimal in a Bayesian (asymptotic) sense, as it attains the best possible rate of decay of the posterior probability of the set of wrong models.

As a Bayesian algorithm, `TTTS` uses a prior distribution Π_0 over the vector of means of the K arms, $\boldsymbol{\mu} \triangleq (\mu_1, \ldots, \mu_K)$, which can be updated to a posterior distribution Π_t after t observations. Under the Bernoulli bandit model, arm i produces a reward $Y_{t,i} = 1$

with probability μ_i, and $Y_{t,i} = 0$ with probability $1 - \mu_i$ when sampled at round t. Given independent uniform prior for the mean of each arm, the posterior distribution on $\boldsymbol{\mu}$ is a product of K Beta distributions: $\Pi_t = \bigotimes_{i=1}^{K} \texttt{Beta}(1 + S_{t,i}, N_{t,i} - S_{t,i} + 1)$, where $N_{t,i}$ is the number of selections of arm i until round t and $S_{t,i}$ is the sum of rewards obtained from that arm. At each round t, TTTS chooses one arm from the following two candidates to evaluate: (1) it first samples a parameter $\boldsymbol{\theta}$ from Π_{t-1}, and the first candidate is defined as $I_t^{(1)} \triangleq \arg\max_{i \in \mathcal{A}} \theta_i$, (2) it repeatedly samples new $\boldsymbol{\theta}'$ until $I_t^{(2)} \triangleq \arg\max_{i \in \mathcal{A}} \theta'_i$ is different from $I_t^{(1)}$. TTTS depends on a parameter $\beta \in (0, 1)$. In particular, the algorithm selects $I_t = I_t^{(1)}$ with probability β and $I_t = I_t^{(2)}$ with probability $1 - \beta$.

TTTS can also be used for bandit settings in which the rewards are bounded in $[0, 1]$ by using a binarization trick first proposed by Agrawal and Goyal (2012): When a reward $Y_{t,i} \in [0, 1]$ is observed, the algorithm is updated with a fake reward $Y'_{t,i} \sim \texttt{Ber}(Y_{t,i}) \in \{0, 1\}$. TTTS can thus be used for BAI for a *finite* number of arms that with rewards in $[0, 1]$. We now present a simple way of extending TTTS to deal with an infinite number of arms.

Dynamic TTTS In an infinite bandit algorithm, at each round, we either query a new arm from the reservoir *and sample it*, or re-sample a previous arm. In a Bayesian setting, we can also imagine that at each round, an arm is queried from the reservoir and added with a *uniform prior* to the list of queried arms, *regardless of whether it is sampled or not*. Then, at round t, D-TTTS consists in running TTTS on these t arms, out of which several are endowed with a uniform prior and have never been sampled.

Leveraging the fact the the maximum of k uniform distribution has a $\texttt{Beta}(k, 1)$ distribution and that TTTS only depends on the maxima of posterior samples, we give the following equivalent implementation for D-TTTS (Algorithm 1). Letting $\mathcal{L}_t$ be the list of arms that have been queried from the reservoir and sampled *at least once* before round t, at round t we run TTTS on the set $\mathcal{A}_t \triangleq \mathcal{L}_t \cup \{\mu_0\}$ where μ_0 is a pseudo-arm with posterior distribution $\texttt{Beta}(t - k_t, 1)$, where $k_t \triangleq |\mathcal{L}_t|$.

It remains to decide how to recommend the arm as our best guess. In this paper, we choose the most natural recommendation strategy for Bayesian algorithms that outputs the arm with the largest *posterior probability of being optimal*. Letting Θ_i be the subset of the set Θ of possible mean vectors such that arm i is optimal, $\Theta_i \triangleq \left\{\boldsymbol{\theta} \in \Theta \,|\, \theta_i > \max_{j \neq i} \theta_j\right\}$, the posterior probability that arm i is optimal after round t is defined as $\Pi_t(\Theta_i)$. At any time t, we therefore recommend arm $\widehat{I}_t \triangleq \arg\max_{i \in \mathcal{A}} \Pi_t(\Theta_i)$.

4. Experiments

We benchmark our bandit-based strategy against different types of HPO algorithms, namely, TPE, random search, Hyperband and a simple Thompson sampling variant of Hyperband (called H-TTTS described in Appendix B), for the tuning of classifiers (SVM and MLP) on 4 different classification tasks: *wine*, *breast cancer*, and *adult* datasets from UCI machine learning repository (Dua and Taniskidou, 2017); and the MNIST dataset (LeCun et al., 1998).

For all the methods, a noisy evaluation of the black-box function f (see the terminology introduced in Section 2) for a hyper-parameter configuration $\boldsymbol{\lambda}$ consists in performing a shuffled 3-fold cross-validation on $\mathcal{D}_{\texttt{train}}$. More precisely, given a random partitioning $\cup_{j=1}^{3} \mathcal{D}^j_{\texttt{valid}}$ of $\mathcal{D}_{\text{train}}$, where the folds are of equal size, we train a classifier $\widehat{g}_{\boldsymbol{\lambda}}^{(j)}$

Algorithm 1 Sampling rule of Dynamic `TTTS` (`D-TTTS`)

Input: β; B (total budget); ν_0

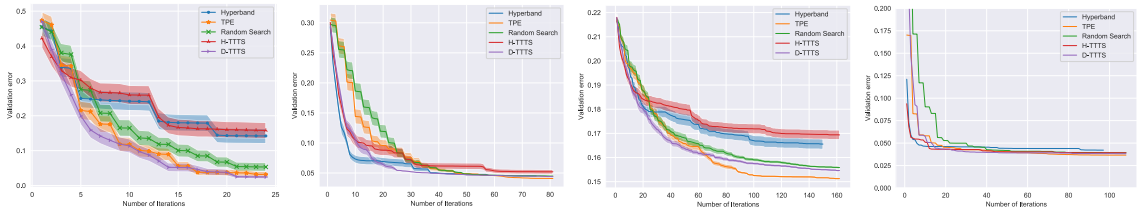
Initialization: $\mu_1 \sim \nu_0$; $t \leftarrow 0$; $\mathcal{A} \leftarrow \{\mu_0, \mu_1\}$; $m \leftarrow 1$; $S_0, N_0 \leftarrow 0$; $S_1 \sim \texttt{Ber}(\mu_1)$, $N_1 \leftarrow 1$

while $t < B$ **do**
  $\forall i = 0, \dots, m$, $\theta_i \sim \texttt{Beta}(S_i + 1, N_i - S_i + 1)$; $U \sim \mathcal{U}([0,1])$
  $I^{(1)} \leftarrow \arg\max_{i=0,\dots,m} \theta_i$
  if $U > \beta$ **then**
    while $I^{(2)} \neq I^{(1)}$ **do**
      $\forall i = 0, \dots, m, \theta'_i \sim \texttt{Beta}(S_i + 1, N_i - S_i + 1)$
      $I^{(2)} \leftarrow \arg\max_{i=0,\dots,m} \theta'_i$
    end while
    $I^{(1)} \leftarrow I^{(2)}$
  end if
  if $I^{(1)} \neq 0$ **then**
    $Y \leftarrow$ `evaluate arm` $I^{(1)}$; $X \sim \texttt{Ber}(Y)$
    $S_{I^{(1)}} \leftarrow S_{I^{(1)}} + X$; $N_{I^{(1)}} \leftarrow N_{I^{(1)}} + 1$; $S_0 \leftarrow S_0 + 1$
  else
    $\mu_{m+1} \sim \nu_0$; $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mu_{m+1}\}$;
    $Y \leftarrow$ `evaluate arm` $m+1$; $X \sim \texttt{Ber}(Y)$
    $S_{m+1} \leftarrow X$; $N_{m+1} \leftarrow 1$; $m \leftarrow m + 1$
  end if
  $t \leftarrow t + 1$
end while

on $\mathcal{D}_{\texttt{train}} \backslash \mathcal{D}^j_{\texttt{valid}}$ for each fold j and compute the average validation error defined as $e \triangleq 1/|\mathcal{D}_{\texttt{train}}| \sum_{j=1}^{3} \sum_{i \in \mathcal{D}^j_{\text{valid}}} \mathbb{1}\{\widehat{g}^{(j)}_{\boldsymbol{\lambda}}(\mathbf{x}_i) \neq \mathbf{y}_i\}$, which we report as a noisy estimate of the risk $f(\boldsymbol{\lambda}) \triangleq \mathbf{P}(\widehat{g}^{(n)}_{\boldsymbol{\lambda}}(\boldsymbol{X}) \neq \boldsymbol{Y})$.

Observe that both the noisy evaluation and the value of f belong to $[0,1]$. Therefore we can introduce an *arm* with rewards in $[0,1]$ for each hyper-parameter $\boldsymbol{\lambda}$. Sampling arm $\boldsymbol{\lambda}$ produces reward $r \triangleq 1 - e \in [0,1]$ with a different random partitioning and random seed for training for each selection. Arm λ is assumed to have mean of $1 - f(\boldsymbol{\lambda})$. In an infinite arm setting, querying a new arm from the reservoir corresponds to selecting a new hyper-parameter at random from the search space. With these two notions (*arm sampling* and *reservoir querying*), our algorithm for infinite BAI applies to HPO.

For the experiments, we adapt the recommendation rule of `D-TTTS` to the HPO applications considered and always recommend the hyper-parameter configuration that has produced the smallest cross-validation error so far (which is also the recommendation rule used by other approaches, e.g., `Hyperband`). For all methods, we report the cross-validation error for the recommended hyper-parameter configuration, as a function of time. We stress again that, unlike in standard bandits, where we could use the simple regret as a performance metric, we do not have access to the ground truth generalization error in real classification tasks. Therefore, we only report a proxy of the true error rate that we are interested in.



(a) wine (b) breast cancer (c) adult (d) MNIST

Results We first benchmark[2] our methods on a few simple UCI datasets using SVM from scikit-learn as the classifier. We optimize over two hyper-parameters: the *penalty parameter* C and the *kernel coefficient* γ[3] for an RBF kernel, for which the pre-defined search bounds are both $\left[10^{-5}, 10^{5}\right]$.

Fig. 1a shows the mean cross-validation error of SVM run on the UCI wine dataset over 24 pulls[4] averaged on 100 runs. The task is to predict the quality score of wine (between 0 and 10) given 11 attributes. Recall that one iteration corresponds to one arm pull. In this experiment, D-TTTS improves over other benchmark algorithms. Fig. 1b is the same experiment run on the UCI breast cancer dataset over 81 pulls. The task is to predict whether a patient has breast cancer based on 32 attributes. We repeat the experiment 100 times. This time, D-TTTS is slightly worse than Hyperband at the beginning, but improves later. Finally, we optimize SVM on a relatively more complicated UCI adult dataset over 162 pulls, for which the result is shown in Fig. 1c. The task is to tell whether the income of an individual is higher than 50k or not given 14 attributes. This experiment is also averaged over 100 runs. D-TTTS is better than other algorithms at the beginning, but is outperformed by TPE towards the end. We see that, although not always the best, D-TTTS shows a consistent, robust, and quite competitive performance in the 3 tasks.

We now carry out the classic MNIST digits classification task using multi-layer perceptron (MLP). We choose to optimize over three hyper-parameters: the *size of hidden layer* (an integer between 5 and 50), the ℓ_2 *penalty parameter* α (between 0 and 0.9) and the *initial learning rate* (bounded in $\left[10^{-5}, 10^{-1}\right]$). Fig. 1d shows the result of MLP run on MNIST over 108 pulls, this time averaged over 20 runs. D-TTTS is slightly worse than Hyperband and H-TTTS in the very beginning, but is performing well afterward.

5. Discussion

We presented a way to use Thompson sampling for BAI for infinitely many-armed bandits and explained how to use it for HPO. We introduced the *first fully dynamic algorithm* for this setting and showed through an empirical study that it is a promising approach for HPO. In the future, we plan to provide experiments on more datasets and establish theoretical guarantees to support the good performance of D-TTTS, with the hope to provide a finite-time upper bound on its probability of error. We also plan to investigate variants of this algorithm for the non-stochastic bandits for which Hyperband can be used, which would allow spending more time on the more promising algorithms.

2. code at http://researchers.lille.inria.fr/~valko/hp/publications/shang2019simple.code.zip
3. γ is the parameter of the RBF kernel defined as $\exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)$
4. the number of pulls here and later is chosen exactly as in the work of Li et al. (2017)

Acknowledgments

The research presented was supported by European CHIST-ERA project DELTA, French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council, Inria and Otto-von-Guericke-Universitt Magdeburg associated-team north-European project Allocate, and French National Research Agency projects BADASS (n.ANR-16-CE40-0002), and BoB (n.ANR-16-CE23-0003).

References

Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Conference on Learning Theory (CoLT)*, pages 1–26, 2012.

Jean-Yves Audibert and Sébastien Bubeck. Best-arm identification in multi-armed bandits. In *Proceedings of the 23rd Conference on Learning Theory (CoLT)*, 2010.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi- armed bandit problem. *Machine Learning Journal*, 47(23):235–256, 2002.

Maryam Aziz, Jesse Anderton, Emilie Kaufmann, and Javed Aslam. Pure exploration in infinitely-armed bandit models with fixed-confidence. In *Proceedings of the 29th International Conference on Algorithmic Learning Theory (ALT)*, 2018a.

Maryam Aziz, Kevin Jamieson, and Javed Aslam. Pure-exploration for infinite-armed bandits with general arm reservoirs. *arXiv preprint arXiv:1811.06149*, 2018b.

Peter L. Bartlett, Victor Gabillon, and Michal Valko. A simple parameter-free and adaptive approach to optimization under a minimal local smoothness assumption. In *Proceedings of the 30th International Conference on Algorithmic Learning Theory (ALT)*, 2019.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 2546–2554, 2011.

Donald A. Berry, Robert W. Chen, Alan Zame, David C. Heath, and Larry A. Shepp. Bandit problems with infinitely many arms. *Annals of Statistics*, 25(5):2103–2116, 1997.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory (ALT)*, pages 23–37, 2009.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvari. $\mathcal{X}$-armed bandits. *Journal of Machine Learning Research*, 12:1587–1627, 2010.

Alexandra Carpentier and Michal Valko. Simple regret for infinitely many armed bandits. In *Proceedings of the 32nd International conference on Machine Learning (ICML)*, pages 1133–1141, 2015.

Dheeru Dua and Karra E. Taniskidou. UCI machine learning repository, 2017.

Eyal Even-dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 162–169, 2003.

Stefan Falkner, Aaron Klein, and Frank Hutter. `BOHB`: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. Best-arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 3212–3220, 2012.

Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 667–675, 2015.

Matthew W. Hoffman, Bobak Shahriari, and Nando de Freitas. On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AIStats)*, pages 365–374, 2014.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION)*, pages 507–523, 2011.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1238–1246, 2013.

Aaron Klein, Stefan Falkner, Numair Mansur, and Frank Hutter. `RoBO`: A flexible and robust Bayesian optimization framework in Python. In *7th Workshop on Bayesian Optimization at Neural Information Processing Systems (NIPS-BayesOpt)*, 2017.

Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. `GPflowOpt`: A Bayesian optimization library using TensorFlow. *arXiv preprint arXiv:1711.03845*, 2017.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2019.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Ameet Talwalkar, and Afshin Rostamizadeh. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Ilya Loshchilov and Frank Hutter. CMA-ES for hyperparameter optimization of deep neural networks. In *Workshop Track of the 4th International Conference on Learning Representations*, 2016.

Daniel Russo. Simple Bayesian algorithms for best arm identification. In *Proceedings of the 29th Conference on Learning Theory (CoLT)*, 2016.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Xuedong Shang, Emilie Kaufmann, and Michal Valko. General parallel optimization without a metric. In *Proceedings of the 30th International Conference on Algorithmic Learning Theory (ALT)*, 2019.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 2951–2959, 2012.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International conference on Machine Learning (ICML)*, pages 1015–1022, 2010.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285, 1933.

Yizao Wang, Jean-Yves Audibert, and Rémi Munos. Algorithms for infinitely many-armed bandits. In *Advances in Neural Information Processing Systems 21 (NIPS)*, pages 1729–1736, 2008.

Appendix A. Best-arm identification for HPO

Hyper-parameter optimization can be modeled as a BAI task in a bandit model. Given a finite set of arms $\mathcal{A} \triangleq \{1, \dots, K\}$, once we select arm i, we get an independent observation from some unknown distribution ν_i with mean μ_i. A BAI algorithm sequentially selects arms in order to identify the arm with the largest mean, $I^\star \triangleq \arg\max_{i \in \mathcal{A}} \mu_i$.[5] In the context of HPO, each arm models the quality of a given hyper-parameter configuration $\boldsymbol{\lambda}$. When the arm is sampled, a noisy evaluation of $f(\boldsymbol{\lambda})$ is received, where $f(\boldsymbol{\lambda})$ is the mean reward of that arm. A BAI algorithm consists of a sequential arm selection strategy, indicating which arm I_t is selected at round t, coupled with *recommendation rule* that selects a candidate best arm $I_t^\star$ at round t. The goal is either to minimize the error probability $\mathbf{P}(\mu_{I_t^\star} \neq \mu_{I^\star})$ (Audibert and Bubeck, 2010; Karnin et al., 2013) or the *simple regret* (Bubeck et al., 2009; Gabillon et al., 2012), defined as $r_t = \mu_{I^\star} - \mu_{I_t^\star}$, possibly after a total budget B, whose knowledge may be used by the algorithm. Note that the BAI problem can also be studied from a fixed-confidence point of view (Even-dar et al., 2003).

Standard BAI algorithms are however not straightforwardly applicable to HPO when the search space is infinite or continuous. To handle these important cases, we rather turn our attention to BAI in infinitely many-armed bandits (Carpentier and Valko, 2015). In this context, there is an infinite pool of arms, whose means are assumed to be drawn from some *reservoir distribution* ν_0. In this setting, a BAI algorithm can maintain a list of arms (hyper-parameter configurations) that have been tried before. At each round, it can either query a new arm from the reservoir (add a new hyper-parameter, selected at random, to the current pool), add it to the list and sample it (evaluate it), or, sample an arm already in the list (re-evaluate a configuration tried before).

A natural way to perform BAI in an infinitely many-armed bandit model consists in first querying a *well-chosen* number of arms from the reservoir and then running a standard BAI algorithm on those arms (Carpentier and Valko, 2015). However, this ideal number may rely on the difficulty of the learning task, which is hardly known in practice. The `Hyperband` algorithm (Li et al., 2017) takes a step further and successively queries several batches of arms from the reservoir, including a decreasing number of arms in each batch, while increasing the budget dedicated to each of them. `SequentialHalving` (`SHA`, Karnin et al., 2013), a state-of-the-art BAI algorithm, is then run on each of these batches of arms. This approach seems more robust in that it trades off between *the number of arms that is needed to capture a good arm* and *how much measurement effort we should allocate to each of them.* However, a numerical study performed by Aziz et al. (2018b) seems to reveal that an infinite bandit algorithm based on `SHA` should always query the maximal number of arms from the reservoir.[6]

All the existing algorithms are still subject to a pre-defined scheduling of how many arms should be queried from the reservoir. Contrary to them, our algorithm (`D-TTTS`) does not need to decide in advance how many arms will be queried, and is therefore fully *dynamic*.

5. In this paper, we present BAI problems in a standard way for which we search for an arm with the largest mean. However, for HPO, it is important to mention that we look for a hyper-parameter configuration that minimizes the *validation error*. One can easily see that it does not change the essence of the problem.
6. The reference we give is a preliminary draft that has been withdrawn due to technical issues in the proof. Yet, we believe the experimental section to be sound.

Remark 1 *`Hyperband` was given specifically for hyper-parameter tuning. Its original philosophy is to adaptively allocate resources to more promising configurations. Resources can be time, the size of a random subset of the dataset, the number of randomly sampled features, etc. In this setting, the classifier is not always trained into completion given a parameter configuration, but is rather stopped early if it is empirically not performing well so that we can allocate more resources to other configurations. In such case, different evaluations of a single configuration are not i.i.d. anymore. Thus, HPO is stated as a non-stochastic infinitely-armed bandit problem. The idea of early stopping we further combined with Bayesian optimization (Falkner et al., 2018).*

Appendix B. Hyper-TTTS

Algorithm 2 Sampling rule of Hyper `TTTS` (`H-TTTS`)

Input: β; γ; B; $s_{\max}$; ν_0
Initialization: $T \leftarrow \lfloor B/s_{\max} \rfloor$

for $s \leftarrow s_{\max}$ to 0 **do**
  $K \leftarrow \left\lceil \frac{s_{\max}+1}{s+1}\gamma^s \right\rceil$
  $\mathcal{A} \leftarrow \{i = 1, \ldots, K : \mu_i \sim \nu_0\}$; $t \leftarrow 0$
  while $t < T$ **do**
    `sample` $\boldsymbol{\theta} \sim \Pi_t$
    $I^{(1)} \leftarrow \arg\max_{i \in \mathcal{A}} \theta_i$
    `sample` $b \sim$ `Ber`(β)
    if $b = 1$ **then**
      $Y \leftarrow$ `evaluate arm` $I^{(1)}$
    else
      while $I^{(2)} \neq I^{(1)}$ **do**
        $\forall i \in \mathcal{A}, \theta_i' \sim$ `Beta`$(S_i + 1, N_i - S_i + 1)$
        $I^{(2)} \leftarrow \arg\max_{i \in \mathcal{A}} \theta_i'$
      end while
      $I^{(1)} \leftarrow I^{(2)}$
      $Y \leftarrow$ `evaluate arm` $I^{(1)}$
    end if
    $X \sim$ `Ber`(Y)
    $S_{I^{(1)}} \leftarrow S_{I^{(1)}} + X$; $N_{I^{(1)}} \leftarrow N_{I^{(1)}} + 1$
    $t \leftarrow t + 1$
  end while
end for

We also show another way of extending `TTTS` to deal with an infinite number of arms and call it `Hyper-TTTS` or `H-TTTS`, a variant of `Hyperband` in which `SHA` is replaced by `TTTS`. `H-TTTS`, formally stated as Algorithm 2, runs $s_{\max}$ batches of `TTTS` with different number of arms and each batch with a same budget $T \triangleq \lceil B/s_{\max} \rceil$ with B the total budget. The number of arms within each bracket is decreasing by a factor of γ. One inconvenience of

H-TTTS is that $s_{\max}$ and γ still need to be tuned. In our experiments, we use the same tuning as used by Hyperband (Li et al., 2017).

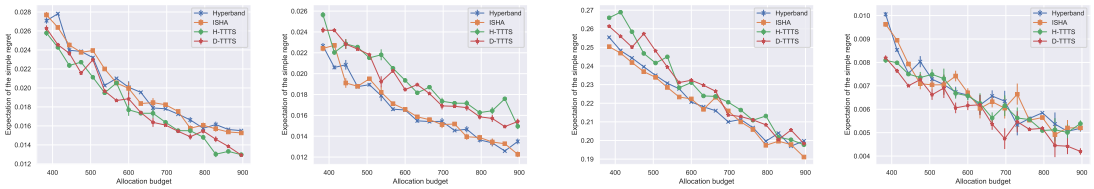
Appendix C. More numerical simulations

In this part, we give results comparing D-TTTS to Hyperband and to ISHA (infinite sequential halving, Aziz et al. 2018b), an adaption of sequential halving to the infinite bandit setting. In these experiments, the arms are Bernoulli distributed and the reservoir distribution ν_0 is fixed to some Beta(a, b) distribution.

ISHA consists in running SHA on a fixed number of arms drawn from the reservoir. Observe that for a total budget B, there exists a maximum number of arms $K^\star$ that can be processed by SHA, which satisfies $B \triangleq \lceil K^\star \log_2(K^\star) \rceil$. Following Aziz et al. (2018b), we run ISHA with $K^\star$ arms drawn from the reservoir.

In Fig. 2, we report the simple regret as a function of time for different algorithms for four Beta reservoir distributions. H-TTTS and D-TTTS are run with $\beta = 1/2$ which is known to be a robust choice (Russo, 2016). Each point represents the expected simple regret $\mathbb{E}[1 - \mu_{I_n^*}]$ estimated with 1000 runs for an algorithm run with budget n. D-TTTS is very competitive on 3 reservoirs and H-TTTS is sometimes better, sometimes worse than Hyperband. We also tried the SiRI algorithm (Carpentier and Valko, 2015) with b as the tail parameter where $\nu_0 =$ Beta(a, b), but obtained worse performance and therefore we do not report its results.

Note that in the implementation of Hyperband for this *stochastic* infinite bandit setting we modified the algorithm so that the elimination phase of the underlying SHA algorithm is carried out according to the average loss of previous samples, as samples from an arm are i.i.d. in this setting and not a converging sequence.



(a) Beta(1,1) (b) Beta(3,1) (c) Beta(1,3) (d) Beta(0.5,0.5)

Figure 2: Simple regret as a function of the number of arms evaluations