Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization

Lisha Li, Kevin Jamieson, Giulia DeSalvo,
Afshin Rostamizadeh, Ameet Talwalkar

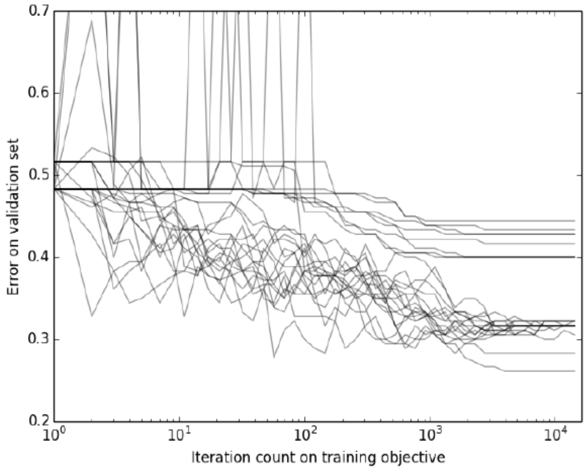
JMLR 2017

Information & Intelligence System Lab.
Sang Heon Lee
lawlee1@naver.com

2018/08/10

Introduction

- Hyperparameter optimization
  - Finding particular hyperparameters that optimizes some evaluation criterion, e.g., loss on a validation set
- High variability in model quality across hyperparameter settings



- Can we **identify and terminate poor-performing hyperparameter settings early** in a principled online fashion to speed up hyperparameter optimization?

Introduction (Cont'd)

- Previous research
  - Evaluate models after they are fully trained to convergence
  - Require explicit forms for the convergence rate behavior of the iterations

- Propose a robust, general-purpose, widely applicable bandit-based solution to hyperparameter optimization
  - without any information about the rate of convergence
  - non-stochastic best arm identification problem

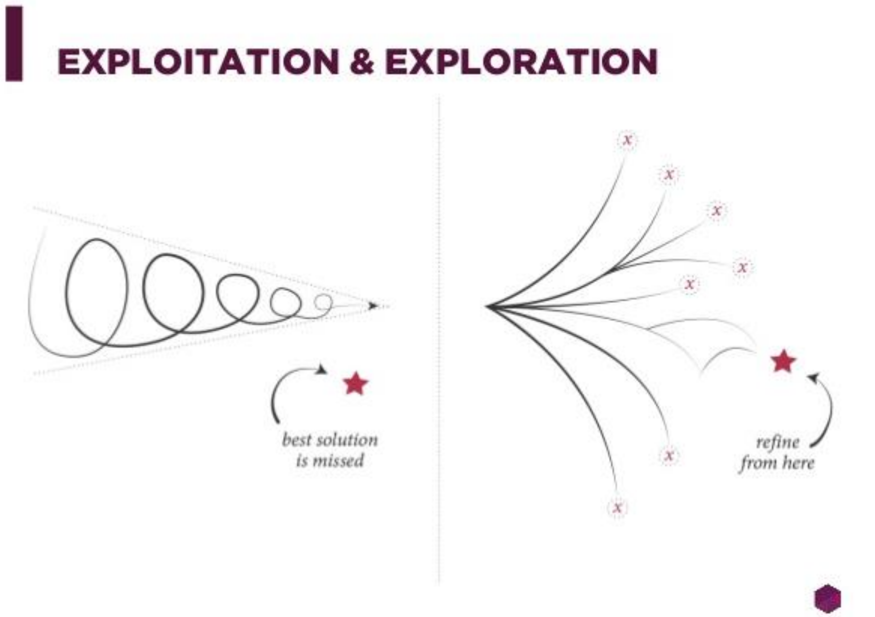
Background

- Multi-armed bandit problem

  - n different slot machines (arms)
  - Each machines have different rewards (regrets)
  - *How to **maximize rewards** (or **minimize regrets**) in limited time?*

Background (Cont'd)

- Exploitation vs. Exploration
  - Exploitation : choose the best arm until now
  - Exploration : choose a new arm to get more information
  - trade-off relations : the proper control of these two behaviors is core



Background (Cont'd)

- Stochastic and non-stochastic best arm identification
  - n arms where $l_{i,k}$ denotes the loss observed on the kth pull of the ith arm

    T_i is the number of times the ith arm was pulled

Stochastic : For all $i \in [n]$, $k \geq 1$, let $\ell_{i,k}$ be an i.i.d. sample from a probability distribution on $[0,1]$ such that $\mathbb{E}[\ell_{i,k}] = \mu_i$. The goal is to identify $\arg\min_i \mu_i$ while minimizing $\sum_{i=1}^{n} T_i$.

Non-stochastic (proposed in this work) : For all $i \in [n]$, $k \geq 1$, let $\ell_{i,k} \in \mathbb{R}$ be generated by an oblivious adversary, i.e., the loss sequences are independent of the algorithm's actions. Further, assume $\nu_i = \lim_{\tau \to \infty} \ell_{i,\tau}$ exists. The goal is to identify $\arg\min_i \nu_i$ while minimizing $\sum_{i=1}^{n} T_i$.

Background (Cont'd)

- Multi-armed bandit problem and hyperparameter optimization
  - best arm identification problem

  - arms = hyperparameter settings
  - # of pulling the arm = number of training iterations
  - regret (output) = intermediate validation loss of model

  - MAB problem in hyperparameter optimization
    : find hyperparameter settings that minimize final loss

Proposed method

- Find the best arm in limited time as soon as possible
  - *"The question is not if the algorithm will identify the best arm, but how fast it does so relative to a baseline method."*
- Non-stochastic best arm identification problem
  - No assumptions on the convergence behavior of the validation losses
- Simple and intuitive algorithm
  - requires no inputs or free-parameters to adjust
  - outperforms the baseline method

Proposed method (Cont'd)

- *Successive Halving* Algorithm
  - Originally introduced by Zohar Karnin (2013)

  1. Set a budget (total time for hyperparameter tuning)
  2. Uniformly allocate the budget to a set of hyperparameter configurations
  3. Evaluate their performance
  4. Throw out the worst half
  5. Repeat until one configuration remains

- The budget as an input is easily removed by the "doubling trick"
  - B ← n, run, B ← 2*B, …

Successive Halving Algorithm

input: Budget B, n arms where $\ell_{i,k}$ denotes the kth loss from the ith arm

Initialize: $S_0 = [n]$.

For $k = 0, 1, \ldots, \lceil \log_2(n) \rceil - 1$

Pull each arm in S_k for $r_k = \lfloor \frac{B}{|S_k| \lceil \log_2(n) \rceil} \rfloor$ additional times and set $R_k = \sum_{j=0}^{k} r_j$.

Let σ_k be a bijection on S_k such that $\ell_{\sigma_k(1),R_k} \leq \ell_{\sigma_k(2),R_k} \leq \cdots \leq \ell_{\sigma_k(|S_k|),R_k}$

$S_{k+1} = \left\{ i \in S_k : \ell_{\sigma_k(i),R_k} \leq \ell_{\sigma_k(\lfloor |S_k|/2 \rfloor),R_k} \right\}$.

output : Singleton element of $S_{\lceil \log_2(n) \rceil}$

Proposed method (Cont'd)

- *Successive Halving* Algorithm

Successive Halving Algorithm

input: Budget B, n arms where $\ell_{i,k}$ denotes the kth loss from the ith arm

Initialize: $S_0 = [n]$.

For $k = 0, 1, \ldots, \lceil \log_2(n) \rceil - 1$

Pull each arm in S_k for $r_k = \lfloor \frac{B}{|S_k| \lceil \log_2(n) \rceil} \rfloor$ additional times and set $R_k = \sum_{j=0}^{k} r_j$.

Let σ_k be a bijection on S_k such that $\ell_{\sigma_k(1),R_k} \leq \ell_{\sigma_k(2),R_k} \leq \cdots \leq \ell_{\sigma_k(|S_k|),R_k}$

$S_{k+1} = \{ i \in S_k : \ell_{\sigma_k(i),R_k} \leq \ell_{\sigma_k(\lfloor |S_k|/2 \rfloor),R_k} \}$.

output : Singleton element of $S_{\lceil \log_2(n) \rceil}$

[Example] B = 64(epochs), n=16

- k = 0, 1, 2, 3

- k=0)
  - S_0 = [16], r_0 = $\left\lfloor \frac{64}{|[16]| \lceil \log_2 16 \rceil} \right\rfloor = 1$
  - iterate 1 epoch for each 16 hyperparameters
  - remove the half worst : S_1 = [8]

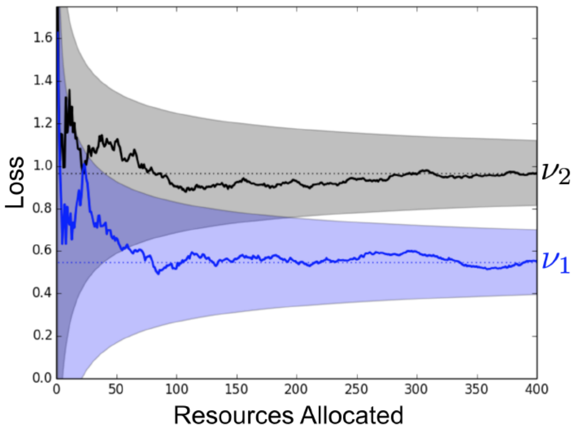
- k=1)
  - S_1 = [8], r_1 = $\left\lfloor \frac{64}{|[8]| \lceil \log_2 16 \rceil} \right\rfloor = 2$
  - iterate 2 epochs for each 8 hyperparameters
  - remove the half worst : S_2 = [4]

…

Proposed method (Cont’d)

- Analysis of Success Halving algorithm
  - Budget B, n arms where $l_{i,k}$ denotes the kth loss from the ith arm
  - $\nu_i = \lim_{\tau \to \infty} \ell_{i,\tau}$ for $i = 1, \ldots, n$ and $\nu_1 < \nu_2 \leq \cdots \leq \nu_n$
  - $\gamma_i(t)$: non-increasing function of t with $|\ell_{i,t} - \nu_i| \leq \gamma_i(t)$
  - $\gamma_i^{-1}(\alpha) = \min\{t \in \mathbb{N} : \gamma_i(t) \leq \alpha\}$
  - if $t_i > \gamma_i^{-1}(\frac{\nu_i - \nu_1}{2})$ and $t_1 > \gamma_1^{-1}(\frac{\nu_i - \nu_1}{2})$, then

$$\begin{aligned}\ell_{i,t_i} - \ell_{1,t_1} &= (\ell_{i,t_i} - \nu_i) + (\nu_1 - \ell_{1,t_1}) + 2\left(\frac{\nu_i - \nu_1}{2}\right) \\ &\geq -\gamma_i(t_i) - \gamma_1(t_1) + 2\left(\frac{\nu_i - \nu_1}{2}\right) > 0\end{aligned}$$



  - so $\ell_{i,t_i} > \ell_{1,t_1}$

    → Comparing immediate loss at t_i and t_1 suffices to determine ordering of final values v_i and v_1

Proposed method (Cont’d)

- Comparing immediate loss at t_i and t_1 suffices to determine ordering of final values v_i and v_1
  - But, we don’t have any information about $\gamma_i(t)$
  - When is t_i and t_1 ?

- Increase pulling time of each arm gradually
  - Remove the worst half at each step
  - Using “doubling trick”
    - B<-n, run, B<-2B, run, …

Proposed method (Cont'd)

- **Theorem 1** *Let* $\nu_i = \lim_{\tau \to \infty} \ell_{i,\tau}$, $\bar{\gamma}(t) = \max_{i=1,\dots,n} \gamma_i(t)$, *and*

$$\begin{aligned} z_{SH} &= 2\lceil \log_2(n) \rceil \max_{i=2,\dots,n} i \left(1 + \bar{\gamma}^{-1}\left(\frac{\nu_i - \nu_1}{2}\right)\right) \\ &\leq 2\lceil \log_2(n) \rceil \left(n + \sum_{i=2,\dots,n} \bar{\gamma}^{-1}\left(\frac{\nu_i - \nu_1}{2}\right)\right). \end{aligned}$$

If the Successive Halving algorithm is run with any budget $B > Z_{SH}$ *then the best arm is guaranteed to be returned.*

- If the algorithm is bootstrapped by the "doubling trick", then this procedure returns the best arm once

Proposed method (Cont'd)

- Drawback of *Successive Halving* Algorithm
  - Requires number of configurations n as an input
  - Given finite budget B, B/n resources are allocated
    → trade-off between n and B/n : exploration and exploitation problem

- *Hyperband* algorithm
  - addresses this "n versus $r(=B/n)$" problem
  - 2 loops

  (1) inner loop invokes *Successive Halving* for fixed values of n and r

  (2) outer loop iterates over different values of n and r

  - Each run of *Successive Halving* algorithm in *Hyperband* = "bracket"

Proposed method (Cont'd)

- *Hyperband* algorithm

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

input : R, η (default $\eta = 3$)
initialization : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$
for $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
  $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $\quad r = R\eta^{-s}$
  // begin SUCCESSIVEHALVING with (n, r) inner loop
  T = get_hyperparameter_configuration(n)
  for $i \in \{0, \ldots, s\}$ **do**
    $n_i = \lfloor n\eta^{-i} \rfloor$
    $r_i = r\eta^i$
    $L = \{$run_then_return_val_loss$(t, r_i) : t \in T\}$
    T = top_k$(T, L, \lfloor n_i/\eta \rfloor)$
  end
end
return *Configuration with the smallest intermediate loss seen so far.*

- R : maximum amount of resource(training time) that can be allocated to a single configuration
- η : percentage of configurations discarded at each step

Proposed method (Cont'd)

- *Hyperband* algorithm

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

input : R, η (default $\eta = 3$)
initialization : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$

for $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ do
    $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r = R\eta^{-s}$
    // begin SUCCESSIVEHALVING with (n, r) inner loop
    T = get_hyperparameter_configuration(n)
    for $i \in \{0, \ldots, s\}$ do
        $n_i = \lfloor n\eta^{-i} \rfloor$
        $r_i = r\eta^i$
        $L = \{$run_then_return_val_loss$(t, r_i) : t \in T\}$
        T = top_k$(T, L, \lfloor n_i/\eta \rfloor)$
    end
end
return *Configuration with the smallest intermediate loss seen so far.*

[Example] R = 81, η =3

- s_{max} = $\lfloor \log_3 81 \rfloor = 4$: 5 brackets (0, 1, 2, 3, 4)
- B = (s_{max}+1)R = 5*81

- s=4)
  - n = $[\frac{5*81}{81}\frac{3\text{\^{}}4}{5}]$ = 81, r $= 81 * 3\text{\^{}}(-4) = 1$
  - get 81 configurations
  - Successive Halving with 1/3 discarded

- s=3)
  - n = $[\frac{5*81}{81}\frac{3\text{\^{}}3}{4}]$ = 34, r $= 81 * 3\text{\^{}}(-3) = 3$
  - get 34 configurations
  - Successive Halving with 1/3 discarded

…

Proposed method (Cont'd)

- *Hyperband* algorithm

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

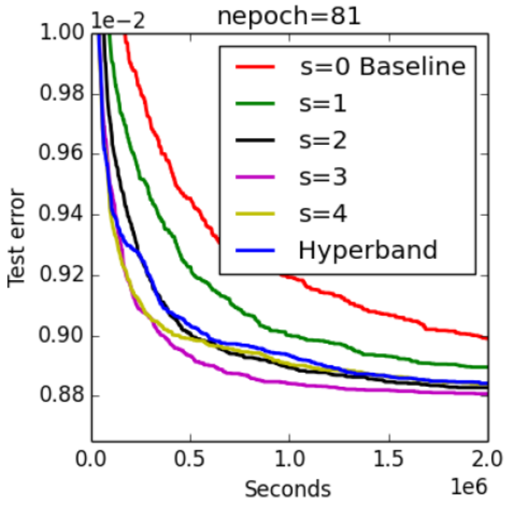
input : R, η (default $\eta = 3$)
initialization : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max}+1)R$

for $s \in \{s_{\max}, s_{\max}-1, \ldots, 0\}$ do
  $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r = R\eta^{-s}$
  // begin SUCCESSIVEHALVING with (n, r) inner loop
  T = get_hyperparameter_configuration(n)
  for $i \in \{0, \ldots, s\}$ do
    $n_i = \lfloor n\eta^{-i} \rfloor$
    $r_i = r\eta^i$
    $L = \{\text{run\_then\_return\_val\_loss}(t, r_i) : t \in T\}$
    T = top_k($T, L, \lfloor n_i/\eta \rfloor$)
  end
end
return *Configuration with the smallest intermediate loss seen so far.*

[Example] R = 81, η =3

- 5 Successive Halving algorithm with different n and r

	$s=4$		$s=3$		$s=2$		$s=1$		$s=0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								



avg error across 70 trials

Proposed method (Cont'd)

- Features of *Hyperband* algorithm
  - Different types of resources
    - Training time
    - Training dataset
    - Feature Subsampling (# of filters in image processing)
  - Determine R and η
    - R as the ratio between max and min resource
    - η as 3 or 4

Experiment

- Hyperparameter Optimization Experiments
  - 3 different resource type : iterations, dataset, feature subsamples

  - Compare with 3 *Bayesian Optimization* algorithms
    - *SMAC, TPE, Spearmint*
    - Variant of *SMAC* using early termination proposed by Domhan et al. (2015)

  - baseline : random search, "random 2X" (B=B*2) : for speedups

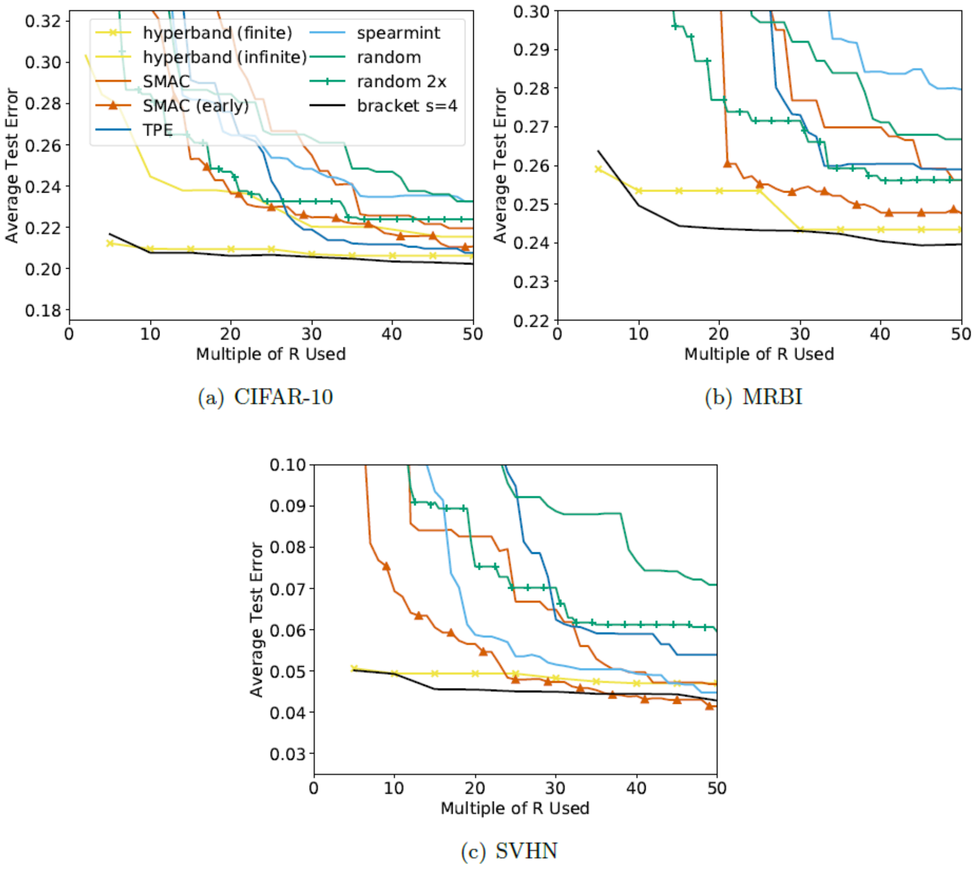
  - getting hyperparameters configuration : uniform sampling

Experiment (Cont'd)

(1) Resource : training iterations

- Tuning CNN
- 8 hyperparameters
  - 6 for SGD, 2 for norm layers
- Dataset
  - CIFAR-10 (R=300), MRBI (R=300), SVHN (R=600)
- batch size = 100
- unit of R : 100 mini-batch iterations, η = 4

- Result
  - 20x faster than random search
  - less variable than other searchers

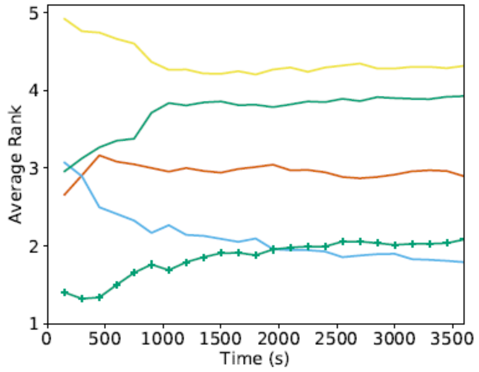


avg error across 10 trials

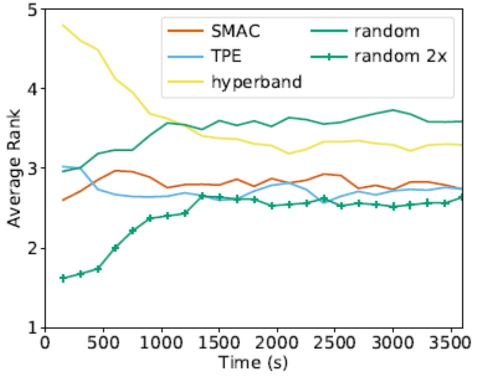
Experiment (Cont'd)

(2-1) Resource : dataset

- Framework introduced by Feurer (2015)
  - explored a structured hyperparameter search space
  - 110 hyperparameters

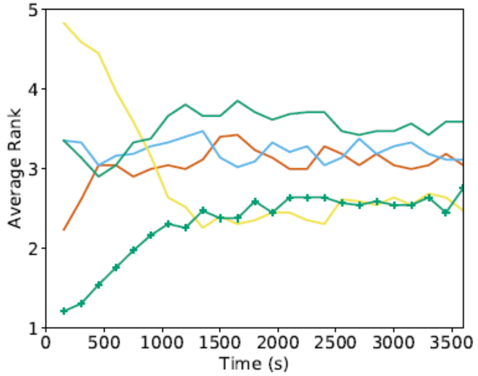


(a) Validation Error on 117 Data Sets



(b) Test Error on 117 Data Sets

- Dataset
  - 117 binary and multiclass classification dataset for OpenML



(c) Test Error on 21 Data Sets

- R = full training set size for each data set

  η = 3

avg error across 20 trials

Experiment (Cont'd)

(2-2) Resource : dataset

- Kernel-based classifier for CIFAR-10
  - multi-class regularized least squares classification model

- Hyperparameter
  - preprocessing method, kernel type, regularization, … : 6 hyperparameters

- R = 400 (unit : 100 datapoints), η = 4

- Result
  - 30x faster than B.O, 70x faster than random

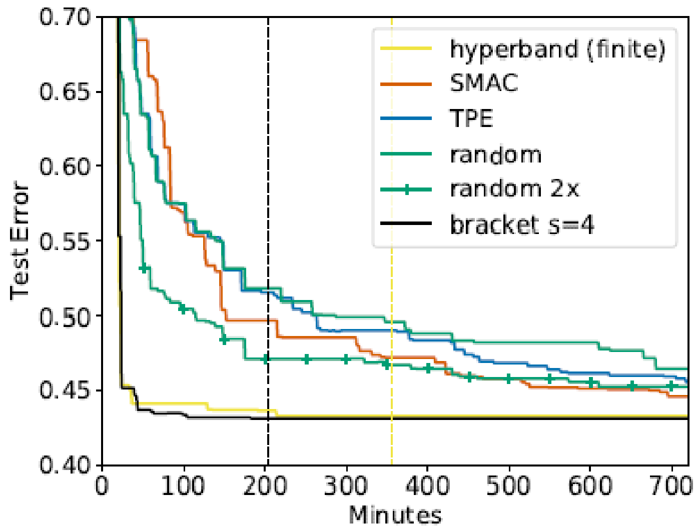


Figure 7: Average test error of the best kernel regularized least square classification model found by each searcher on CIFAR-10. The color coded dashed lines indicate when the last trial of a given searcher finished.

avg error across 10 trials

Experiment (Cont'd)

(3) Resource : feature subsample

- Kernel-based classifier for CIFAR-10
  - multi-class regularized least squares classification model
- Features were randomly generated
  - method described in Rahimi (2007)
- R = 1000 (unit : 100 features), η = 4
- Result
  - 6x faster than B.O and random

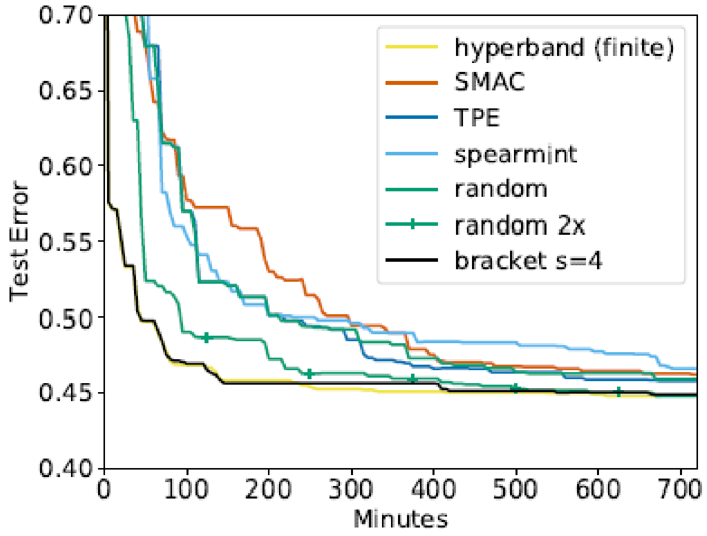


Figure 8: Average test error of the best random features model found by each searcher on CIFAR-10. The test error for HYPERBAND and bracket $s = 4$ are calculated in every evaluation instead of at the end of a bracket.

Conclusion

- Hyperparameter optimization method from non-stochastic multi-armed bandit problem perspective

- Proposed *Hyperband* algorithm
  - Search hyperparameter configuration without any information about convergence of loss
  - Faster than Bayesian optimization, better than baseline

- Combining *Bayesian Optimization* with *Hyperband*