Bandit Limited Discrepancy Search and Application to Machine Learning Pipeline Optimization

Akihiro Kishimoto, Djallel Bouneffouf, Radu Marinescu, Parikshit Ram, Ambrish Rawat, Martin Wistuba[a], Paulito Palmes, and Adi Botea[b]

IBM Research

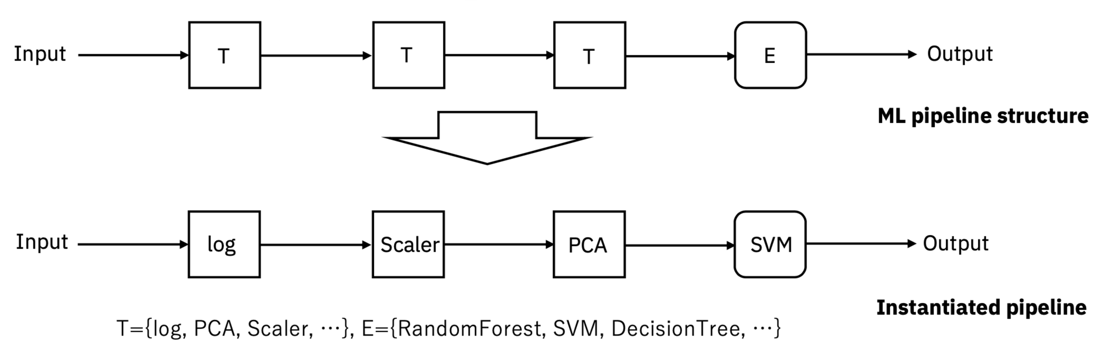
IBM Research

[a] Currently at Amazon Research, Germany   [b] Currently at Eaton, Ireland

Research Motivations

- AutoML has become acute due to the recent explosion in ML applications.
- We focus on a scenario where AutoML optimizes a pipeline with a *fixed* structure for a given *black-box* objective function.
  - *Combined Algorithm Selection and Hyper-parameter optimization (CASH)*
    - Instantiating ML methods for pipeline stages and performing hyper-parameter optimization (HPO) for these selected ML methods.
    - Implying an optimization problem with a fixed number of decision variables.
- We focus on the *algorithm selection* problem in the fixed pipeline structure and introduce Bandit Limited Discrepancy Search (BLDS).
  - Effective algorithm selection methods are incorporated into the alternating direction method of multipliers (ADMM) (Liu *et al.*, 2020).
  - ADMM splits CASH into the algorithm selection phase and the HPO solved separately in an iterative manner.

Example of Algorithm Selection Task

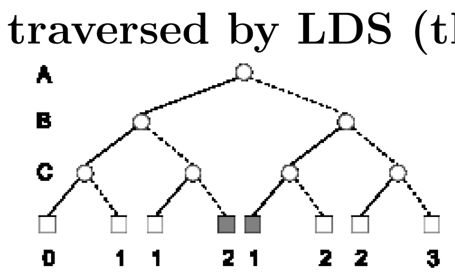


Bandit Limited Discrepancy Search (BLDS)

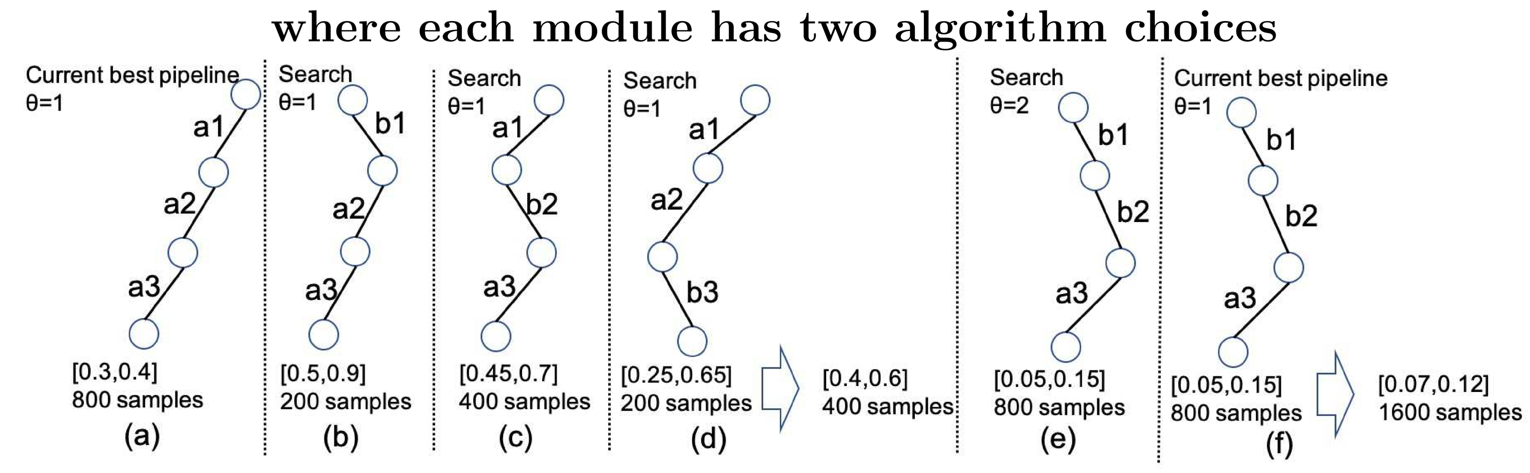
- BLDS combines three ideas:
  - Limited Discrepancy Search (LDS) (Harvey and Ginsberg, 1995)
  - Multi-fidelity optimization (Sabharwal *et al.*, 2016)
  - Multi-armed bandit algorithm (Auer *et al.*, 2002)

- BLDS starts with a randomly initial pipeline repeatedly refined by LDS.
  - Assuming that a better pipeline tends to be instantiated in a similar fashion to the current best pipeline.
  - Examining a limited search space where similar pipelines are located.
  - Re-initializing the pipeline when a better solution is not found.
- BLDS starts with a small subset of training data and increases the size of the subset.
  - The size of training data: $b\eta^{j-1}$ for the j-th evaluation of a pipeline
- BLDS assumes the real objective value for pipeline p to be in $[LCB, UCB]$ and decides:
  - Whether or not p is promising.
  - Whether or not p should be trained with a larger training subset.
  - $LCB = v - \sqrt{\frac{\log\frac{cLD_k^2}{\delta}}{D_k}}$, $UCB = v + \sqrt{\frac{\log\frac{cLD_k^2}{\delta}}{D_k}}$.
    - v: objective value for the k-th evaluation with validation set V
    - $c\ (>4), \delta$: constants, $D_k = \sum_{j=1}^{k} b\eta^{(j-1)}$
    - L = the number of possible pipelines

Search space traversed by LDS (the height is 3)



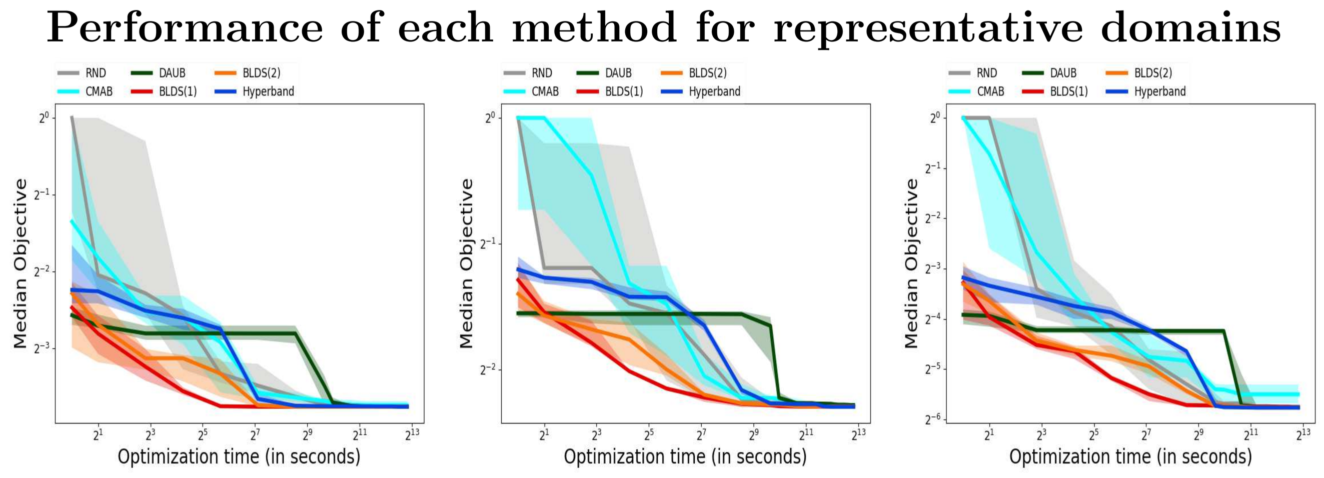
BLDS ($\eta = 2$) for a three-step pipeline structure where each module has two algorithm choices



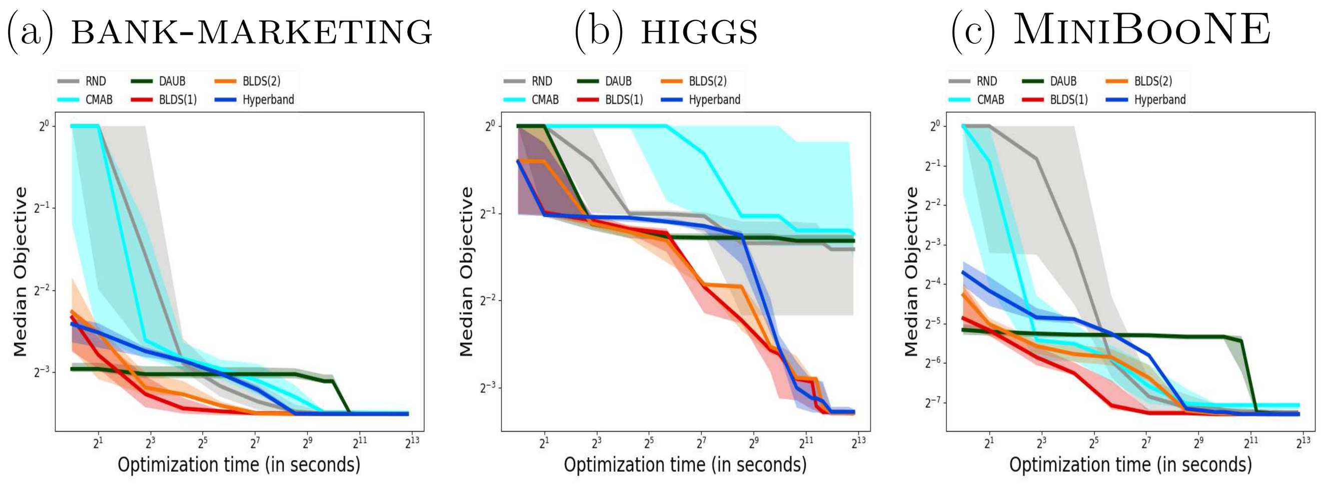
Experimental Results

- Machine: Intel Xeon CPU E5-2667 processor at 3.3GHz (one core in use)
- Algorithms: BLDS(1), BLDS(2), CMAB, DAUB, Hyperband and RND
  - BLDS, DAUB and Hyperband use the same multi-fidelity optimization strategy
- 10 benchmarks from OpenML repositories (binary classification)
  - (1.0 – AUROC) as the black-box objective function
  - 70-30% train-validation split and 10 runs
  - Time limit of two hours per algorithm per run
- 4-stage pipeline structure with 3072 possible pipelines
  - Three data preprocessing/transforming steps and one estimation step

Performance of each method for representative domains



(a) BANK-MARKETING (b) HIGGS (c) MINIBOONE



(d) ADULT (e) GUILLERMO (f) NOMAO

Summary of Experimental Results

- The multi-fidelity optimization tends to have has an advantage.
- BLDS(1) tends to achieve better objective values much more quickly than the others.
- DAUB suffers from a significant overhead in its bootstrapping due to more configurations than in (Sabharwal *et al.*, 2016)
  - 3072 ML pipelines versus only 41 ML classifiers.
- BLDS(2) under-performs BLDS(1) possibly caused by a much larger number of pipelines needed to be re-trained and evaluated within a discrepancy threshold θ.
  - 26 pipelines within $\theta = 1$ versus 272 pipelines within $\theta = 2$.

Conclusions

Summary

- Introduced BLDS to address algorithm selection in AutoML.
- Demonstrated that BLDS empirically performs well and tends to converge more quickly than the other competing algorithms.

Future Work

- Combine BLDS with HPO under AutoML ADMM.
- Further enhance search to be able to deal with large-scale training data and more complicated pipeline structures.
  - Combination with an approach for selecting candidate pipelines with meta-learning
  - Introduction of a better discrepancy value allowing for a more granular control of the local search space
  - Parallelization of BLDS
- Apply BLDS to other tasks, e.g., HPO.
- Have a better theoretical understanding to our MAB strategy.