Summary of "Google Vizier: A Service for Black-Box Optimization"

Adam Stautberg (astaut) and Mihir Patel (patilmi)

Problem and Motivation

The main motivation behind this paper was creating a system for black-box optimization. Black-box optimization is necessary for systems that, due to complexity, become easier to experiment with than to understand. Black-box optimization makes minimal assumptions about the problem under consideration, therefore it is used across a broad range of problems. Google Vizier aims to be a scalable black-box optimization system that can handle millions of trials per study, thousands of parallel trial evaluations per study, and billions of studies. The other main goals of Google Vizier are ease of use and ability to experiment with new algorithms.

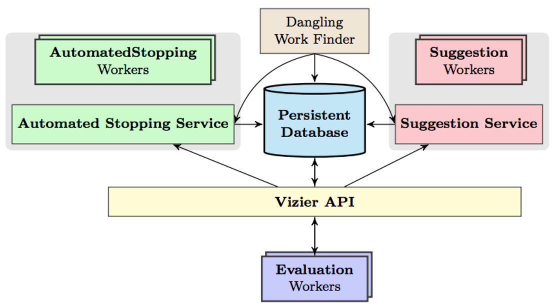
Terminology:
Trial: list of parameter values, x, that will lead to a single evaluation of $f(x)$. If it is completed than the value of $f(x)$ has been assigned to it.
Study: represents a single optimization run over a feasible space. Each Study contains a configuration describing the feasible space, as well as a set of Trials. It is assumed that $f(x)$ does not change in the course of a Study

Hypothesis

Google Vizier uses a distributed suggestion service to provide black-box optimization across billions of studies for Google.

Solution Overview

Google Vizier Architecture:



Implemented as an RCP API, vizier consists of a persistent database with an automated stopping service, suggestion service, and a dangling work finder.

Automated stopping service: Performance curve stopping rule estimates result based on previous trials and intermediate results, terminates unpromising trials. Median stopping rule terminates a trial if its best objective value yet is less than the median value of previous trials at the same step.

Dangling work finder: Will alert engineers if a study is failing too often.

Suggestion service: Does the actual black-box optimization work of coming up with suggestions and running trials.

Limitations and Possible Improvements

This paper does not really touch on limitations or possible improvements.

Summary of Class Discussion

Q: The first sentence of the paper is the following: "Any sufficiently complex system acts as a black box when it becomes easier to experiment with than to understand." It seems to me that black box optimization in general would yield very little information about why certain parameter values are more optimal, rather it would just determine the optimal values. I feel like knowing *why* a certain parameter value is optimal is very important in most use cases, why is this less important in black box optimization?
A: Essentially black-box optimization does not tell you the why. It is used for just finding the optimal input, finding out why a certain combination is optimal is a separate area of research.

Q: Could you explain figure 4? I wasn't able to fully understand it.
A: Essentially each strand is a trial and each vertical line is a dimension representing the value of a parameter for each trial. It is mainly just to show an example of a visualization tool.

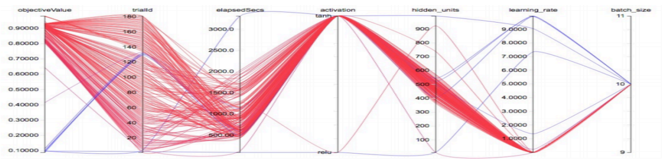


Figure 4: The Parallel Coordinates visualization [18] is used for examining results from different Vizier runs. It has the benefit of scaling to high dimensional spaces (∼15 dimensions) and works with both numerical and categorical parameters. Additionally, it is interactive and allows various modes of slicing and dicing data.

Q: Of the four optimization algorithms currently implemented that were discussed in 4.2, do you know if Vizier has any way of helping a user choose which algorithm to use? Or is that on the user to be familiar with these algorithms and choose one for their use case?
A: By default Vizier will dynamically choose the best algorithm.

HyperBand Summary

Problem and Motivation

-Machine learning models exploding in complexity and expressibility along with concurrent explosion in computational costs.

-Sheer number of tunable parameters makes the problem unwieldy

-Parameters greatly affect model performance but are not well understood therefore brute force methods are common in practice.

Hypothesis

-Evaluate many configurations quickly rather than run models to completion before evaluating.

-Much time is spent on evaluating poor models. Would be better to save computation as soon as we can surmise that it is a poor configuration

Solution Overview

-The Hyperband solution is described as pure exploration adaptive resource allocation. Solution focuses on assessing models with the purpose of dropping them rather than intelligently generating them. Throwing out poor models quicker allows for more resources to be spent on exploring more models.

-HyperBand extends Successive Halving by Tawalkar and Jamieson. Successive Halving throws out the worst n given configurations at every stage.

-HyperBand then grid searches over allocated budgets and top % of models to keep, while running the successive halving algorithm. Each such combination of ratio and budget is called a bracket. Hyperband begins with the most aggressive (lowest ratio of kept vs thrown) bracket first.

Limitations and Possible Improvements

-Doesn’t guarantee convergence.

-Doesn’t seem to really beat SuccessiveHalving

-Async Successive halving.

Summary of Class Discussion

-Asynchronous successive halving was brought up

-Theoretical analysis of various situations involving hyperband are included in the appendix